INTERNATIONAL STANDARD

ISO 10303-215

First edition
2004-05-15

Industrial automation systems and integration — Product data representation and exchange —

Part 215: Application protocol: Ship arrangement

Systèmes d'automatisation industrielle et intégration — Représentation et échange de données de produits —

Partie 215: Protocole d'application: Aménagement des navires

Reference number
ISO 10303-215:2004(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents Page

Figures

Tables

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75% of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO 10303 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

This part of ISO 10303 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC 4, *Industrial data.*

ISO 10303 is organized as a series of parts, each published separately. The structure of ISO 10303 is described in ISO 10303-1.

Each part of ISO 10303 is member of one of the following series: description methods, implementation methods, conformance testing methodology and framework, integrated generic resources, integrated application resources, application protocols, abstract test suites, application interpreted constructs, and application modules. This part is a member of the application protocols series.

A complete list of parts of ISO 10303 is available from the Internet:

<http://www.tc184-sc4.org/titles/>

Introduction

ISO 10303 is an International Standard for the computer-interpretable representation of product information and for the exchange of product data. The objective is to provide a neutral mechanism capable of describing products throughout their life cycle. This mechanism is suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases, and as a basis for archiving.

This part of ISO 10303 is a member of the application protocol series. This part of ISO 10303 specifies an application protocol (AP) for the exchange of product data representing a ship's internal subdivision information between different organizations with a need for that data. Such organizations include ship owners, design agents, fabricators, and classification societies.

This part of ISO 10303 is one of a series of shipping industry application protocols that together aim to provide an integrated computer interpretable product model for ships.

The series of shipping industry application protocols assumes that the ship product model can be divided into separate ship systems that each cover a key element of the ship for its whole life cycle. These key elements are: ship moulded forms, ship arrangement, ship distribution systems, ship structures, ship mechanical systems, ship outfit and furnishings, and ship mission systems. Each separate system is described by one or more different application protocols. The full series of ship application protocols is shown in Figure 1. Those aspects of the ship product model that are common to each shipbuilding application protocol are described consistently and identically in each application protocol. Annex K provides additional information on the shipbuilding application protocols and their elements. It also contains information on data common to the shipbuilding application protocols.

This application protocol has been developed to support the shipbuilding activities and computer applications associated with the functional design, detail design, production engineering, and operations life cycle phases for commercial or military ships. The types of design activities and computer applications supported include the arrangement of spaces within a ship, definition of the intended function of the compartments and zones, detail design of the compartments and zones, geometric representation of compartments and zones, compartment property requirements, compartment property as-designed and as-built values, identification of the outfitting items contained in a compartment, definition of cargoes, association of cargoes to a compartment, definition of design and operating loading conditions, and damage stability analysis. Figure 2 illustrates the major types of data supported by this application protocol. This planning model was created from the in-scope data from the activities of the application activity model (AAM) and grouped into logical units of functionality. This planning model is used as a guide in developing the application reference model (ARM).

This application protocol defines the context, scope, and information requirements for the exchange of ship arrangement definitions, geometric representations of compartments and zones, compartment properties, cargoes, cargo assignments, loading conditions, and damage stability information, and specifies the integrated resources necessary to satisfy these requirements.

Application protocols provide the basis for developing implementations of ISO 10303 and abstract test suites for the conformance testing of AP implementations.

Clause 1 defines the scope of the application protocol and summarises the functionality and data covered by the AP. Clause 3 lists the words defined in this part of ISO 10303 and gives pointers towards defined elsewhere. An application activity model, that is the basis for the definition of the

scope, is provided by annex F. The information requirements of the application are specified in clause 4 using terminology appropriate to the application. A graphical representation of the information requirements, referred to as the application reference model, is given in annex G.

Resource constructs are interpreted to meet the information requirements. This interpretation produces the application interpreted model (AIM). This interpretation, given in 5.1, shows the correspondence between the information requirements and the AIM. The short listing of the AIM specifies the interface to the integrated resources and is given in 5.2. Note that the definitions and EXPRESS provided in the integrated resources for constructs used in the AIM may include select list items and subtypes which are not imported into the AIM. The expanded listing given in annex A contains the complete EXPRESS for the AIM without annotation. A graphical representation of the AIM is given in annex H. Additional requirements for specific implementation methods are given in annex C.

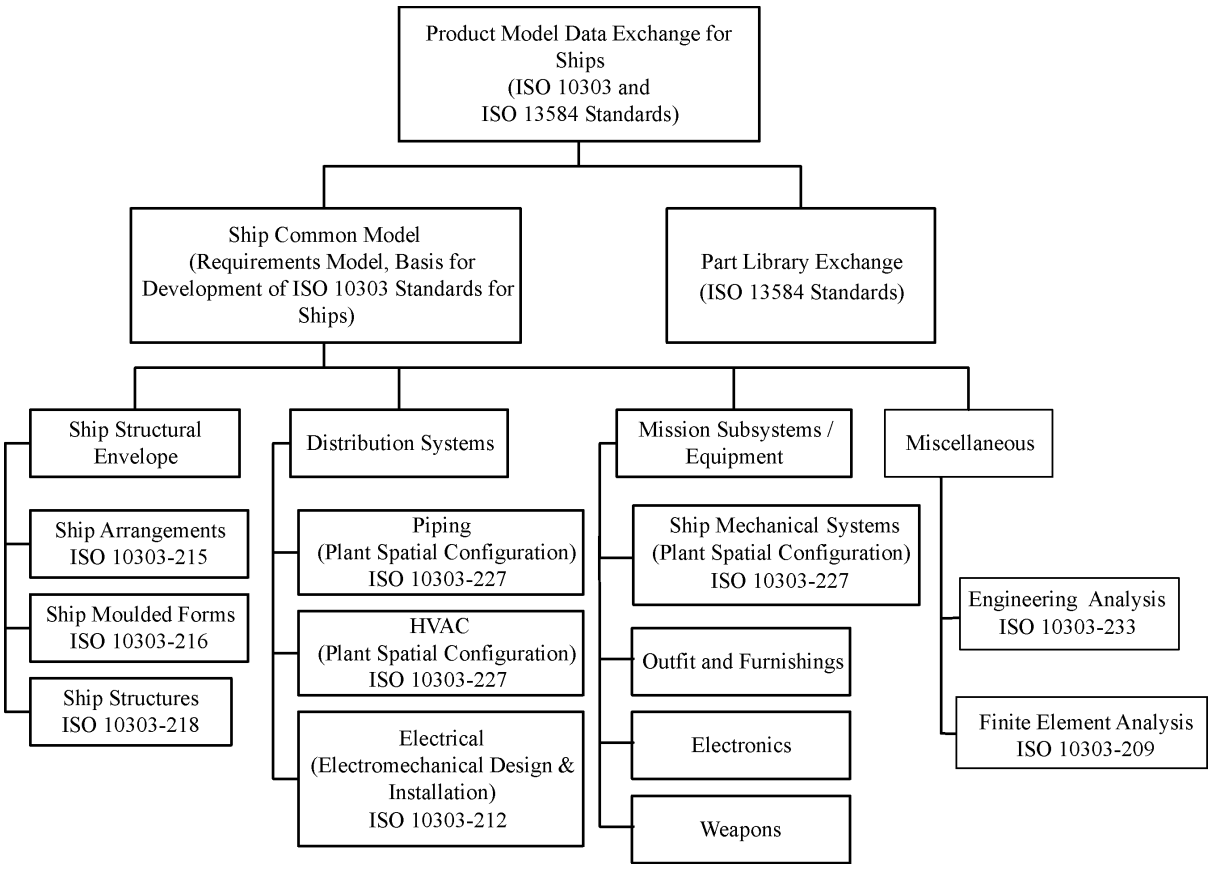

Figure 1 — The full series of ship application protocols

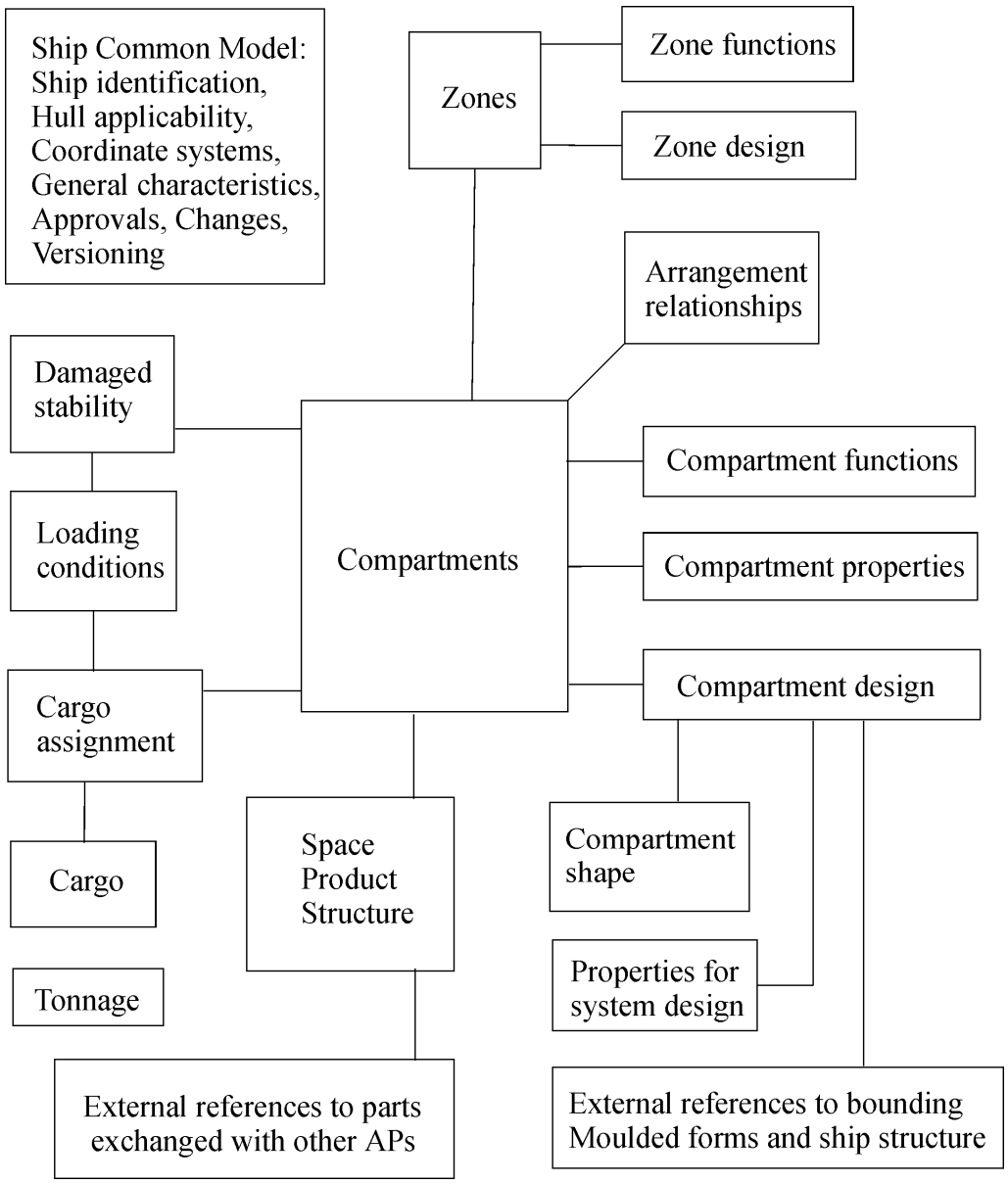

Figure 2 — Data planning model

Industrial automation systems and integration — Product data representation and exchange — Part 215: Application protocol: Ship arrangement

1 Scope

This part of ISO 10303 specifies the use of the integrated resources necessary for the scope and information requirements for the exchange of three-dimensional product definition data and its configuration status information for naval and commercial ship arrangements. Configuration in this context pertains to data specific to revision tracking and change history of selected ship spatial entities within the product model. The term exchange is used to narrow the scope to only those data that are transferred between enterprise systems. This is to distinguish it from a data model supporting distributed, multi-user database applications.

NOTE 1 The application activity model in annex F provides a graphical representation of the processes and information flows that are the basis for the definition of the scope of this part of ISO 10303.

The following are within the scope of this part of ISO 10303:

— data describing the general subdivision of a ship into spatially bounded regions;

— data identifying physical boundaries partitioning the ship into compartments suitable for the stowage of cargo, operation of machinery, and occupancy by crew and passengers;

— data identifying logical boundaries subdividing the ship into zones for the purpose of controlling access, designating design authority, or applying specific design requirements;

— data required for the definition of spatial boundaries based on references to moulded form regions or geometric surfaces;

— configuration management data for identification of versions of compartment designs and for management of changes to the design during the design life cycle phase;

— data identifying the intended functions of compartments and zones;

— data required for recording the volumetric capacities of cargo compartments at various combinations of vessel heel and vessel trim;

— data required for calculation of the magnitude and location of loads acting upon the structural systems of a ship due to the weight of cargoes contained in compartments;

— data required for the determination of adjacency of compartments;

— data identifying spaces related by common functional purpose, position within the ship, or connection by engineering systems;

EXAMPLE Port and starboard wing tank pairs are spaces related by position.

— data identifying dimensional aspects of spaces;

— data identifying the product structuring of engineering parts and structural parts contained within a space;

— data identifying the product structuring of compartments in an area of the ship;

— data required for the definition of design requirements placed on a space by systems within the ship;

— data required for the identification of cargoes, stores and consumables and allocation of those items to compartments and tanks for design analysis or on specific voyages during the operation of the ship;

— definition of loading conditions for analysis of the floating position of the ship under different cargo loading scenarios;

— data required for the analysis of stability of the ship after damage;

— data applicable to a single ship, or to multiple ships in a hull class.

NOTE 2 Annex K provides additional information pertaining to the industrial use of this part of ISO 10303.

The following are outside the scope of this part of ISO 10303:

— data defining the representation of moulded surfaces of structural or non-structural bulkheads;

NOTE 3 Moulded forms are referenced by external instance references to ISO 10303-216.

— data defining the representation of structural systems and parts;

NOTE 4 Structural systems and parts are referenced by external instance references to ISO 10303-218.

— data defining the location, orientation, or geometry of engineering parts and structural parts within a space.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 8501-1, *Preparation of steel substrates before application of paints and related products — Visual assessment of surface cleanliness — Part 1: Rust grades and preparation grades of uncoated steel substrates and of steel substrates after overall removal of previous coatings*

ISO/IEC 8824-1, *Information Technology — Open Systems Interconnection — Abstract Syntax Notation One (ASN.1) — Part 1: Specification of Basic notation*

ISO 10303-1, *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles*

ISO 10303-11, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual*

ISO 10303-21, *Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation methods: Clear text encoding of the exchange structure*

ISO 10303-22, *Industrial automation systems and integration — Product data representation and exchange — Part 22: Implementation methods: Standard data access interface*

ISO/TS 10303-28, *Industrial automation systems and integration — Product data representation and exchange — Part 28: Implementation methods: XML representations of EXPRESS schemas and data*

ISO 10303-31, *Industrial automation systems and integration — Product data representation and exchange — Part 31: Conformance testing methodology and framework: General concepts*

ISO 10303-41, *Industrial automation systems and integration — Product data representation and exchange — Part 41: Integrated generic resource: Fundamentals of product description and support*

ISO 10303-42, *Industrial automation systems and integration — Product data representation and exchange — Part 42: Integrated generic resource: Geometric and topological representation*

ISO 10303-43, *Industrial automation systems and integration — Product data representation and exchange — Part 43: Integrated generic resource: Representation structures*

ISO 10303-45, *Industrial automation systems and integration — Product data representation and exchange — Part 45: Integrated generic resource: Materials*

ISO 10303-216, *Industrial automation systems and integration — Product data representation and exchange — Part 216: Application Protocol: Ship moulded forms*

ISO 10303-218, *Industrial automation systems and integration — Product data representation and exchange — Part 218: Application Protocol: Ship structures*

ISO 10303-508, *Industrial automation systems and integration — Product data representation and exchange — Part 508: Application interpreted construct: Non-manifold surface*

ISO 10303-511, *Industrial automation systems and integration — Product data representation and exchange — Part 511: Application interpreted construct: Topologically bounded surface*

IMO IC110E, *International Convention for the Safety of Life at Sea (SOLAS), Consolidated edition*

IMO ID200E, *International Maritime Dangerous Goods Code (IMDG Code)*

IMO IB520E, *International Convention for the Prevention of Pollution from Ships (MARPOL 73/78), Consolidated edition*

IMO IA701E, *International Convention on Loadlines*

IMO I713E, *International Convention on Tonnage Measurement of Ships*

IMO IA772E, *Code for the Construction and Equipment of Ships Carrying Dangerous Chemicals in Bulk (BCH Code)*

3 Terms, definitions and abbreviations

3.1 Terms defined in ISO 10303-1

For the purposes of this document, the following terms defined in ISO 10303-1 apply:

— application;

— application activity model (AAM);

— application context;

— application interpreted model (AIM);

— application object;

— application protocol (AP);

— application reference model (ARM);

— computer aided design (CAD);

— computer aided manufacture (CAM);

— conformance class;

— conformance requirement;

— data;

— data exchange;

— implementation method;

— information;

— integrated resource;

— interpretation;

— PICS proforma;

— product;

— product data;

— unit of functionality (UoF).

3.2 Terms defined in ISO 10303-21

For the purposes of this document, the following term defined in ISO 10303-21 applies:

— implementation methods.

3.3 Terms defined in ISO 10303-31

For the purposes of this document, the following terms defined in ISO 10303-31 apply:

— conformance testing;

— postprocessor;

— preprocessor.

3.4 Terms defined in ISO 10303-42

For the purposes of this document, the following terms defined in ISO 10303-42 apply:

— axis2_placement_3d;

— bounded curve;

— non-manifold.

3.5 Terms defined in ISO 10303-216

For the purposes of this document, the following terms defined in ISO 10303-216 apply:

— aft perpendicular;

— after perpendicular;

— amidships;

— baseline;

— breadth;

— bulkhead;

— buttock line;

— centreline;

— centroid;

— classification;

— classification society;

— configuration management;

— coordinate system;

— deck;

— deck house;

— depth;

— design waterline;

— forward perpendicular;

— frame;

— furnishings;

— general arrangements;

— hullform;

— hydrostatics;

— hydrostatic property;

— length between perpendiculars;

— longitudinal;

— main deck;

— main dimensions;

— midship;

— moulded form;

— outfitting;

— ship structure;

— station;

— summer load waterline;

— superstructure;

— transom;

— transverse;

— vertical;

— waterline;

— weather deck.

3.6 Terms defined in ISO 10303-218

For the purposes of this document, the following terms defined in ISO 10303-218 apply:

— builder;

— builder hull number;

— combat systems;

— deadweight;

— definition geometry;

— design change process;

— design load;

— design phase;

— engineering parts;

— freeboard;

— hull number;

— lightship;

— main machinery;

— maintenance phase;

— plate;

— predesign phase;

— production engineering;

— ship class;

— ship range;

— stiffener;

— tanks;

— weight of steel.

3.7 Other terms and definitions

For the purposes of this document, the following terms and definitions apply:

3.7.1
collective-protective system zone
a region of a ship completely isolated from the outside environment by maintaining a positive air pressure with respect to non-protected regions for the purpose of protecting the crew from breathing toxic or germ infected air

3.7.2
compartment
a subdivision of a ship, equivalent to a room in a building, used for crew or passenger living space, ship operations, stowage of cargo or supplies, or stowage of fuel and other liquids for the operation of the ship

3.7.3
design zone
a subdivision of a ship product model that may or may not correspond to a compartment subdivision that is used for configuration management of the product modeling process during the design phase

3.7.4
fire zone
an abstract boundary defining a region of the ship requiring special consideration with regard to its ability to contain or withstand a fire

3.7.5
ship arrangement
subdivision of a ship into compartments of various function for use in the operation of the ship, habitation by crew and passengers, and for stowage of cargoes

3.7.6
subdivision
the internal partitioning of a ship into volumetric-based zones or compartments for the purposes of improving survivability in the event of damage or to segregate areas of the vessel for different purposes

3.7.7
subsafe zone
an abstract boundary defining a region of a ship with special design or production requirements with regard to criteria for use on a submersible vessel

3.7.8
vessel heel
rotation of a ship about the longitudinal axis

3.7.9
vessel trim
rotation of a ship about the transverse axis

3.7.10
zone
an abstract boundary identifying a region of a ship with unique requirements or characteristics which must be specially treated in the design or manufacturing process

3.8 Abbreviations

AAM Application Activity Model

AIM Application Interpreted Model

AP	Application Protocol
ARM	Application Reference Model
CAD	Computer Aided Design
CAM	Computer Aided Manufacture
CoG	Centre of Gravity
CPS	Collective Protection System
GUID	globally unambiguous identifier
HVAC	Heating, Ventilation, and Air Conditioning
IMO	International Maritime Organization
PICS	Protocol Implementation Conformance Statement
SCM	Ship Common Model
SI	Système International
SOLAS	Safety of Life at Sea
UoF	Units of Functionality
WT	Watertight

4 Information requirements

This clause specifies the information required for the exchange of ship compartmentation definitions, geometric representations and related design and operational properties.

NOTE 1 A graphical representation of the information requirements is given in annex G.

NOTE 2 The information requirements correspond to those of the activities identified as being within the scope of this application protocol in annex F.

NOTE 3 The mapping specification specified in 5.1 shows how the integrated resources and application interpreted constructs are used to meet the information requirements of this application protocol.

4.1 Units of functionality

This subclause specifies the units of functionality for the ship arrangement application protocol. This part of ISO 10303 specifies the following units of functionality:

— arrangement_relationships;

— cargoes;

— coatings;

— compartment_design_definitions;

— compartment_properties;

— compartment_requirements;

— configuration_management;

— damaged_stability;

— definitions;

— external_references;

— hull_class_applicability;

— items;

— loading_conditions;

— location_concepts;

— product_structures;

— ship_general_characteristics;

— ship_measures;

— spaces;

— surface_representations;

— tonnage;

— weights.

The units of functionality and a description of the functions that each UoF supports are given below. The application objects included in the UoFs are defined in 4.2.

4.1.1 arrangement_relationships

The arrangement_relationships UoF provides information relating to the relationships between compartments that can be used to automate the generation of a preliminary compartmentation arrangement or identify relationships between compartments after they have been populated in the product model.

The following application objects are used by the arrangement_relationships UoF:

— Adjacent_space_surface_area;

— Space_adjacency_relationship;

— Space_arrangement_relationship;

— Space_connection_relationship;

— Space_enclosing_relationship;

— Space_positional_relationship.

4.1.2 cargoes

The cargoes UoF provides the identification of cargoes that can be carried by the ship, applicable properties of those cargoes, and the assignment of those cargoes to compartments in the ship for the purpose of design or operational analysis.

The following application objects are used by the cargoes UoF:

— Absolute_cargo_position;

— Bay_cell_position;

— Bay_position;

— Bulk_cargo;

— Bulk_cargo_assignment;

— Cargo;

— Cargo_assignment;

— Cargo_footprint;

— Cargo_position;

— Compartment_cargo_assignment;

— Dangerous_goods_code;

— Deck_cargo_assignment;

— Detailed_cargo_material_properties;

— Dry_cargo;

— Gaseous_cargo;

— Gaseous_cargo_assignment;

— General_cargo_material_properties;

— Lane_position;

— Liquid_cargo;

— Liquid_cargo_assignment;

— Person_group;

— Relative_cargo_position;

— Unit_cargo;

— Unit_cargo_assignment;

— Unit_cargo_bounding_box;

— Unit_cargo_group.

4.1.3 coatings

The coatings UoF provides the identification of the type of coating used to prevent corrosion of the ship or to protect the ship in the event of fire.

The following application objects are used by the coatings UoF:

— Coating;

— Coating_certification;

— Corrosion_control_coating;

— Fire_safe_coating;

— Primer_coating.

4.1.4 compartment_design_definitions

The compartment_design_definitions UoF provides the design life cycle stage requirements, as-designed definition, and representations for compartments and zones.

The following application objects are used by the compartment_design_definitions UoF:

— Cargo_bay_definition;

— Compartment_design_definition;

— Deck_zone_design_definition;

— Zone_design_definition.

4.1.5 compartment_properties

The compartment_properties UoF provides the required, as-designed, as-built, and operational properties appropriate to a compartment.

The following application objects are used by the compartment_properties UoF:

— Capacity_properties;

— Cargo_compartment_property;

— Coating_level;

— Compartment_abbreviated_name;

— Compartment_acceleration;

— Compartment_access_authorization;

— Compartment_air_circulation_rate;

— Compartment_area_property;

— Compartment_coating;

— Compartment_horizontal_cross_sectional_area_property;

— Compartment_illumination;

— Compartment_insulation;

— Compartment_naval_administrative_property;

— Compartment_noise_category;

— Compartment_nuclear_classification;

— Compartment_occupancy;

— Compartment_property;

— Compartment_safety_class;

— Compartment_security_classification;

— Compartment_stiffened_surface_area_property;

— Compartment_tightness;

— Compartment_unstiffened_surface_area_property;

— Compartment_vertical_longitudinal_cross_sectional_area_property;

— Compartment_vertical_transverse_cross_sectional_area_property;

— Compartment_volume_permeability_property;

— Compartment_volume_property;

— Compartment_ziplist_number;

— Corrosion_protection;

— General_compartment_property;

— Moments_of_inertia;

— Tank_compartment_property;

— Tank_geometric_parameters;

— Tank_piping_design_properties.

4.1.6 compartment_requirements

The compartment_requirements UoF provides the required parameters for which the ship compartmentation must be designed to meet mission requirements or as specified by the applicable classification society.

The following application objects are used by the compartment_requirements UoF:

— Class_bulk_load_requirement_definition;

— Class_compartment_requirement_definition;

— Class_deck_load_requirement_definition;

— Class_tank_requirement_definition;

— Compartment_design_requirement;

— Vehicle_load_description.

4.1.7 configuration_management

The configuration_management UoF covers the characterization of a definition under these principal aspects: its approval, its controlled changes, and the identity and the relationship of different versions within a version history.

The approval information provide when, who and what has been approved and to what level of approval, as well as how approvals are related to each other. The controlled changes information describes when and who modified what definition. Also it describes the impact of the change in terms of whether or not other definitions are created, modified or deleted.

Versions describe what definition is subject to versioning and how different versions are related to each other to provide a version history.

The following application objects are used by the configuration_management UoF:

— Alternative_version_relationship;

— Approval_event;

— Approval_history;

— Change;

— Change_definition;

— Change_impact;

— Change_plan;

— Change_realization;

— Change_request;

— Check;

— Envisaged_version_creation;

— Event;

— Revision;

— Revision_with_context;

— Version_creation;

— Version_deletion;

— Version_history;

— Version_modification;

— Version_relationship;

— Versionable_object_change_event.

4.1.8 damaged_stability

The damaged_stability UoF provides the constructs for predicting the stability of the ship after it has sustained damage to the hull and an identifiable portion of the compartments.

The following application objects are used by the damaged_stability UoF:

— Damage_case;

— Damage_position;

— Damage_stability_definition;

— Floating_position;

— Stability_definition;

— Stability_properties_for_one_floating_position;

— Stability_property;

— Stability_table.

4.1.9 definitions

The definitions UoF describes the abstract concept for the definition of items, item structures and item relationships. The definitions provide the versioned sets of data for each object for various lifecycle phases.

The following application objects are used by the definitions UoF:

— Definition;

— Design_definition;

— Design_requirement;

— Functional_definition;

— General_characteristics_definition.

4.1.10 external_references

The external_references UoF provides the capability and mechanisms by which references can be made to information in other product model data set or refer to information outside a given data exchange or data sharing context, and defines constructs for the identification and reference of standards and documents defined outside of the scope of ISO 10303.

The following application objects are used by the external_references UoF:

— Document;

— Document_portion;

— Document_reference;

— Document_reference_with_address;

— External_instance_reference;

— External_reference;

— External_storage;

— Universal_resource_locator.

4.1.11 hull_class_applicability

The hull_class_applicability UoF allows an element to be related to all hulls in the ship class, or to a specific hull. In a class design, all elements are related to the ship class. By default, an element is applicable to all hulls in the class. Hull specific changes result in elements that are applicable to specific hulls. These changes may be applied at the item level or may be applied at the definition level.

EXAMPLE 1 Types of hull applicability at the item level would be adding a new system, compartment, or equipment.

EXAMPLE 2 Types of hull applicability at the definition level would be different revisions used to call out hull-specific changes to properties or geometry.

The following application object is used by the hull_class_applicability UoF:

— Hull_applicability.

4.1.12 items

The items UoF defines the abstract concepts that comprise the generic product structure for a ship, including the basic objects, relationships between the objects, and the grouping of objects into structures. This UoF also defines the Ship object and the globally unique persistent identifier that is required for certain objects.

The following application objects are used by the items UoF:

— Definable_object;

— Global_id;

— Item;

— Item_relationship;

— Item_structure;

— Ship;

— Versionable_object.

4.1.13 loading_conditions

The loading_conditions UoF provides design and operational definitions the weight of the ship including assigned cargoes and occupants.

The following application objects are used by the loading_conditions UoF:

— Deadweight;

— Floating_position;

— Loading_condition_definition;

— Loading_condition_design_definition;

— Loading_condition_operating_definition.

4.1.14 location_concepts

The location_concepts UoF specifies the information to locate a ship, or any part of it, in a right handed 3D cartesian axis system. Also, it specifies the information required to subdivide any axis into intervals so that they form the reference basis for points in the axis system.

NOTE A coordinate system is either the one and only global coordinate system of the product description and root in the hierarchy, or a local coordinate system. Any number of local coordinate systems may exist. Spacing positions may be defined for any of the three global coordinate system axes. In case, the underlying coordinate system is the global coordinate system, the local origin maybe defined with reference to spacing positions.

The following application objects are used by the location_concepts UoF:

— Buttock_table;

— Frame_table;

— Global_axis_placement;

— Local_co_ordinate_system;

— Local_co_ordinate_system_with_position_reference;

— Longitudinal_position;

— Longitudinal_table;

— Spacing_position;

— Spacing_position_with_offset;

— Spacing_table;

— Station_table;

— Transversal_position;

— Transversal_table;

— Vertical_position;

— Vertical_table;

— Waterline_table.

4.1.15 product_structures

The product_structures UoF provides the constructs for identifying which structural, mechanical, or distributed system parts are contained within or form the boundary of a compartment or zone.

The following application objects are used by the product_structures UoF:

— Space_product_structure;

— Space_product_structure_revision.

4.1.16 ship_general_characteristics

The ship_general_characteristics UoF specifies the basic information that details the ship dimensions and identification. This information is independent of any geometric context. This information includes values for principal dimensions of a ship, designation information for ship related companies, ship class notations, and all relevant rules and regulations. Also the information about the lightship weight distribution and freeboard characteristics is included in this UoF.

The following application objects are used by the ship_general_characteristics UoF:

— Carrier;

— Class_and_statutory_designation;

— Class_notation;

— Class_parameters;

— Freeboard_characteristics;

— Lightship_definition;

— Lightship_weight_item.

— Loadline;

— Navy_ship;

— Owner_designation;

— Principal_characteristics;

— Regulation;

— Research_ship;

— Ship_designation;

— Shiptype;

— Shipyard_designation;

— Working_ship.

4.1.17 ship_measures

The ship_measures UoF specifies the information for representing measures for physical quantities.

The following application objects are used by the ship_measures UoF:

— Centre_location;

— Derived_unit;

— Named_unit;

— Precision.

NOTE See L.1.1 for additional discussion on the use of measures and units in this part of ISO 10303.

4.1.18 spaces

The spaces UoF provides the identification and functional information about the compartments that constitute the spatial partitioning of the interior volume of the ship. It supports both the physical subdivision of the space into compartments, and the logical subdivision of the space into zones.

The following application objects are used by the spaces UoF:

— Compartment;

— Compartment_functional_definition;

— Deck_zone;

— Deck_zone_functional_definition;

— Space;

— Zone;

— Zone_functional_definition.

4.1.19 surface_representations

The surface_representations UoF specifies the information required to represent a compartment or zone as connected surfaces. The UoF combines topological and geometric information together to describe a surface model as a set of connected surface patches. This UoF describes simple geometry such as planes and straight lines, as well as complex geometry such as B-spline curves and surfaces.

If the curves and surfaces are inter-connected then such information is stated explicitly by the use of topology.

The following application object is used by the surface_representations UoF:

— Non_manifold_surface_shape.

4.1.20 tonnage

The tonnage UoF provides the estimated capacity of the ship using rules established by the international classification societies.

The following application objects are used by the tonnage UoF:

— Compartment_group;

— Compensated_gross_tonnage;

— Gross_tonnage;

— Net_tonnage;

— Tonnage_definition;

— Tonnage_measurement.

4.1.21 weights

The weights UoF provides the constructs for weight and centres of gravity of ship items.

The following application objects are used by the weights UoF:

— Moment_3d;

— Weight_and_centre_of_gravity.

4.2 Application objects

This subclause specifies the application objects for the ship arrangement application protocol. Each application object is an atomic element that embodies a unique application concept and contains attributes specifying the data elements of the object. The application objects and their definitions are given below.

NOTE Those attributes inherited from supertype application objects in the Application Reference Model are not repeated in the subtype application object definitions in this subclause. Refer to the appropriate supertype application object definition for the specification of these inherited attributes.

4.2.1 Absolute_cargo_position

An Absolute_cargo_position is a type of Cargo_position (see 4.2.17) that is the position of a Unit_cargo in terms of a position in the global coordinate system of the ship.

The data associated with an Absolute_cargo_position are the following:

— orientation;

— parent_coordinate_system;

— position.

4.2.1.1 orientation

The orientation specifies the orientation of the Unit_cargo. It specifies the angle between the X-axis of the local coordinate system of the Unit_cargo and the X-axis of the global coordinate system of the ship, given that the Z-axes of the two coordinate systems are parallel. The angle is positive if the rotation of the Unit_cargo X-axis is in a counter-clockwise direction with respect to the global X-axis of the ship.

4.2.1.2 parent_coordinate_system

The parent_coordinate_system specifies the parent coordinate system relative to which the unit cargo is placed. See 4.3.1 for the application assertion.

4.2.1.3 position

The position specifies the location of the origin of the local coordinate system of the Unit_cargo within the global coordinate system of the ship.

4.2.2 Adjacent_space_surface_area

An Adjacent_space_surface_area is the area of that portion of the boundary between adjacent spaces that is common to both spaces.

EXAMPLE Two compartments are divided by a longitudinal bulkhead 8 feet high. One compartment is 10 feet wide and the other is 6 feet wide. The adjacent_space_surface_area would be 48 square feet (8 feet x 6 feet).

The data associated with an Adjacent_space_surface_area are the following:

— surface_area.

4.2.2.1 surface_area

The surface_area specifies the area measure of shared boundary between the adjacent spaces.

4.2.3 Alternative_version_relationship

An Alternative_version_relationship is a relationship between two Versionable_object objects (see 4.2.179) of the same type. Each Versionable_object is an alternative to another Versionable_object.

The data associated with an Alternative_version_relationship are the following:

— alternative_1;

— alternative_2;

— reason.

4.2.3.1 alternative_1

The alternative_1 specifies an alternative design for the Versionable_object defined by alternative_2. See 4.3.2 for the application assertion.

4.2.3.2 alternative_2

The alternative_2 specifies an alternative design for the Versionable_object defined by alternative_1. See 4.3.2 for the application assertion.

4.2.3.3 reason

The reason specifies the description of the relationship between the two alternative designs.

4.2.4 Approval_event

An Approval_event is a type of Event (see 4.2.91) that records a change in the status of the organizational review and and acceptance or certification of some product data.

The data associated with an Approval_event are the following:

— approval_reference;

— result;

— user_defined_result.

4.2.4.1 approval_reference

The approval_reference specifies the Approval_history effected by the event. Every Approval_event shall refer to exactly one Approval_history. See 4.3.3 for the application assertion.

4.2.4.2 result

The result specifies product data for a version of the design is to be reviewed, or has been reviewed by an authorized member of the organization and has been approved, rejected, or has some other project-specific status.

The value of result is one of the following:

— approved;

— rejected;

— unapproved;

— user_defined.

NOTE See 4.2.4.2.1 - 4.2.4.2.4 for the definition of each allowable value for result.

4.2.4.2.1 approved

The product data has been reviewed by the appropriate organization and is approved for use in the ship.

4.2.4.2.2 rejected

The product data has been reviewed by the appropriate organization and is not approved for use in the ship. Other product data would normally be created to replace the rejected product data.

4.2.4.2.3 unapproved

The product data has not yet been reviewed or is in the process of being reviewed for approval by the organization.

4.2.4.2.4 user_defined

The product data is assigned a project-specific approval status code to be determined by two or more exchanging organizations.

4.2.4.3 user_defined_result

The user_defined_result specifies a user-defined approval status. The user_defined_result need not be specified for a particular Approval_event. The user_defined_result shall be specified if the result has the value of user_defined.

4.2.5 Approval_history

An Approval_history is a collection of all Approval_events of a specific type defined for a portion of product data.

The data associated with an Approval_history are the following:

— approvals;

— status;

— subject.

4.2.5.1 approvals

The approvals specifies the sequence of Approval_events having occurred up to this point in time. The history must consist of at least one Approval_event. The sequence of Approval_events is assumed to be in chronological order.

4.2.5.2 status

The status specifies the current approval status.

The value of status is one of the following:

— approved;

— rejected;

— unapproved;

— user_defined.

NOTE See 4.2.5.2.1 - 4.2.5.2.4 for the definition of each allowable value for status.

4.2.5.2.1 approved

The product data has been reviewed by the appropriate organization and is approved for use in the ship.

4.2.5.2.2 rejected

The product data has been reviewed by the appropriate organization and is not approved for use in the ship. Other product data would normally be created to replace the rejected product data.

4.2.5.2.3 unapproved

The product data has not yet been reviewed or is in the process of being reviewed for approval by the organization.

4.2.5.2.4 user_defined

The product data is assigned a project-specific approval status code to be determined by two or more exchanging organizations.

4.2.5.3 subject

The subject specifies the product data to which this approval is related. See 4.3.4 for the application assertion.

NOTE A Definition may have zero, one, or many associated approvals. In case it has more than one associated approval, all of them shall be different.

4.2.6 Bay_cell_position

A Bay_cell_position is a type of Bay_position (see 4.2.7) that is the position of a Unit_cargo in terms of cargo bays in a ship, a compartment, or on a deck.

The data associated with a Bay_cell_position are the following:

— bay_number;

— row;

— tier.

4.2.6.1 bay_number

The bay_number specifies the transverse position of the bay within the cargo bay definition. See 4.3.6 for the application assertion.

4.2.6.2 row

The row specifies the longitudinal position of the bay within the cargo bay definition. See 4.3.5 for the application assertion.

4.2.6.3 tier

The tier specifies the vertical position of the bay within the cargo bay definition. See 4.3.7 for the application assertion.

4.2.7 Bay_position

A Bay_position is a type of Cargo_position (see 4.2.17) that defines a bay as a vertical section of the ship, in a transverse orientation, that is one container-length wide but encompasses all of the tiers and

rows in that section. Each Bay_position may be one of the following: a Bay_cell_position (see 4.2.6), or a Lane_position (see 4.2.112).

The data associated with a Bay_position are the following:

— relating_to.

4.2.7.1 relating_to

The relating_to specifies the cargo bays in the ship. See 4.3.8 for the application assertion.

4.2.8 Bulk_cargo

A Bulk_cargo is a type of Dry_cargo (see 4.2.89) that is solid cargo that is not packed, but is carried loose.

EXAMPLE Grain and coal are bulk cargo.

The data associated with a Bulk_cargo are the following:

— natural_angle_of_repose;

— type_of.

4.2.8.1 natural_angle_of_repose

The natural_angle_of_repose specifies the angle naturally subtended with the horizontal by the upper surface of the conic pile, made by the Bulk_cargo when loaded into a hold by a chute using gravity alone.

4.2.8.2 type_of

The type_of specifies the type of Bulk_cargo that can be loaded into the ship.

The value of type_of is one of the following:

— cement;

— coal;

— fish;

— general;

— grain;

— mud;

— ore;

— sugar;

— timber;

— unspecified;

— user_defined.

NOTE See 4.2.8.2.1 - 4.2.8.2.11 for the definition of each allowable value for type_of.

4.2.8.2.1 cement

The Bulk_cargo is cement.

4.2.8.2.2 coal

The Bulk_cargo is coal.

4.2.8.2.3 fish

The Bulk_cargo is fish.

4.2.8.2.4 general

The Bulk_cargo is of a general, non-specific type.

4.2.8.2.5 grain

The Bulk_cargo is grain.

4.2.8.2.6 mud

The Bulk_cargo is mud.

4.2.8.2.7 ore

The Bulk_cargo is ore.

4.2.8.2.8 sugar

The Bulk_cargo is sugar.

4.2.8.2.9 timber

The Bulk_cargo is timber.

4.2.8.2.10 unspecified

The Bulk_cargo is of an unspecified type.

4.2.8.2.11 user_defined

The cargo type is defined by the user.

4.2.9 Bulk_cargo_assignment

A Bulk_cargo_assignment is a type of Compartment_cargo_assignment (see 4.2.43) that is an allocation of Bulk_cargo to a compartment for loading analysis during the design phase or to identify an actual cargo loading during the operating phase.

The data associated with a Bulk_cargo_assignment are the following:

— actual_angle_of_repose;

— cargo_height;

— trimmed.

4.2.9.1 actual_angle_of_repose

The actual_angle_of_repose specifies the actual angle subtended with the horizontal by the upper surface of the conic pile, made by the Bulk_cargo when loaded into a hold.

4.2.9.2 cargo_height

The cargo_height specifies the actual height of the Bulk_cargo loaded. This should be taken as the highest point of the cargo at the top of the cone.

4.2.9.3 trimmed

The trimmed specifies that the natural pile of Bulk_cargo has been flattened and spread out to fill the compartment. A value of TRUE indicates that the Bulk_cargo has been flattened.

4.2.10 Buttock_table

A Buttock_table is a type of Transversal_table (see 4.2.167) that has positions that reference the location of buttocks that are located on the global Y-axis.

4.2.11 Capacity_properties

A Capacity_properties is a measure of volumetric characteristics of a tank or cargo type of compartment computed at some specific combination of level, trim, and heel angle. The level represents the imaginary planar surface at the cargo to non-cargo interface and is relative to a capacity level origin established for the compartment. The attitude of the plane is adjusted to coincide with a vector having a magnitude equal to the level and a direction reflecting the vessel heel and trim. A compartment may have any number of combinations of capacity values, each having a different value for capacity_context.

The data associated with a Capacity_properties are the following:

— capacity_centre;

— capacity_context;

— capacity_heel_angle;

— capacity_level;

— capacity_level_origin;

— capacity_trim_angle;

— capacity_volume;

— user_defined_capacity_context.

4.2.11.1 capacity_centre

The capacity_centre specifies the position of the volumetric centre of the interior region of space formed by the compartment boundaries and the imaginary plane representing the cargo to non-cargo interface. See 4.3.9 for the application assertion.

4.2.11.2 capacity_context

The capacity_context specifies the values representing the significant capacity states.

The value of capacity_context is one of the following:

— full_95_percent;

— full_98_percent;

— pressed_full;

— slack;

— user_defined.

NOTE See 4.2.11.2.1 - 4.2.11.2.5 for the definition of each allowable value for capacity_context.

4.2.11.2.1 full_95_percent

The capacity properties are defined for a 95 percent full tank or cargo compartment.

4.2.11.2.2 full_98_percent

The capacity properties are defined for a 98 percent full tank or cargo compartment.

4.2.11.2.3 pressed_full

The capacity properties are defined for a full tank or cargo compartment in which no volume is left for expansion and the cargo is hard against the tank top and into the vent pipe.

4.2.11.2.4 slack

The capacity properties are defined for a value other than 95 percent, 98 percent, or pressed_full.

4.2.11.2.5 user_defined

The capacity properties are defined for a tank filled to a level specified in the user_defined_capacity_context attribute.

4.2.11.3 capacity_heel_angle

The capacity_heel_angle specifies the amount of rotation about the longitudinal axis of the ship that has been factored into the capacity calculation for the plane representing the interface between the cargo and non-cargo regions of the compartment.

4.2.11.4 capacity_level

The capacity_level specifies the distance between the bottom of the compartment expressed as the capacity_level_origin and the top of an imaginary plane representing the cargo to non-cargo interface. It is measured along a vector offset from the vertical to reflect the capacity heel angle and the capacity trim angle.

4.2.11.5 capacity_level_origin

The capacity_level_origin specifies the point associated with a tank or cargo compartment that represents the vertical reference for measuring the capacity depth levels corresponding to a set of compartment capacities. It may be chosen to represent the bottom of the compartment, the bottom of the sounding tube, or any other convenient location. See 4.3.9 for the application assertion.

4.2.11.6 capacity_trim_angle

The capacity_trim_angle specifies the amount of rotation about the transverse axis of the ship that has been factored into the capacity calculation for the plane representing the interface between the cargo and non-cargo regions of the compartment.

4.2.11.7 capacity_volume

The capacity_volume specifies the enclosed volumetric measurement of the interior region of space formed by the compartment and the imaginary plane representing the cargo to non-cargo interface.

4.2.11.8 user_defined_capacity_context

The user_defined_capacity_context specifies the context for the capacity properties when the value of capacity_context equals "user_defined". The user_defined_capacity_context need not be specified for a particular Capacity_properties.

4.2.12 Cargo

A Cargo is any item of temporary nature loaded onboard a ship for the purpose of being consumed during the voyage, used by the crew or passengers, transferred to another ship while underway, or offloaded at one of the destination ports. Cargo may be secured from shifting during the voyage, but is not permanently afixed to the ship. Each Cargo is either a Dry_cargo (see 4.2.89), a Gaseous_-cargo (see 4.2.100), or a Liquid_cargo (see 4.2.115).

The data associated with a Cargo are the following:

— cargo_hazard;

— description;

— flash_point;

— material_properties;

— pollution_code;

— references;

— required_carriage_temperature;

— un_type_code;

— user_def_cargo.

4.2.12.1 cargo_hazard

The cargo_hazard specifies the classification of the hazards associated with the cargo. The cargo_hazard need not be specified for a particular Cargo. See 4.2.73 for the application assertion.

4.2.12.2 description

The description specifies a free text description of the cargo.

4.2.12.3 flash_point

The flash_point specifies the temperature at which the cargo will spontaneously combust. The flash_point need not be specified for a particular Cargo.

4.2.12.4 material_properties

The material_properties specifies the physical properties of the material which makes up the cargo on the ship. The material_properties need not be specified for a particular Cargo. See 4.3.12 for the application assertion.

4.2.12.5 pollution_code

The pollution_code specifies the degree to which the cargo will cause pollution of the sea if released. This shall be the code as specified in IMO IB520E.

The value of pollution_code is one of the following:

— code_A;

— code_B;

— code_C;

— code_D.

NOTE See 4.2.12.5.1 - 4.2.12.5.4 for the definition of each allowable value for pollution_code.

4.2.12.5.1 code_A

The cargo cannot be released into the sea.

4.2.12.5.2 code_B

The cargo within 0.1 cubic metres per tank can be released into the sea.

4.2.12.5.3 code_C

The cargo within 0.3 cubic metres per tank can be released into the sea.

4.2.12.5.4 code_D

The cargo can be released into the sea so long as it is diluted.

4.2.12.6 references

The references specifies the documents that may be of relevance to the carriage of the cargo. These may be material data sheets, technical specifications, or additional safety information. The references need not be specified for a particular Cargo. There may be more than one references for a Cargo. See 4.3.11 for the application assertion.

4.2.12.7 required_carriage_temperature

The required_carriage_temperature specifies the required temperature of the cargo while it is stowed. The required_carriage_temperature need not be specified for a particular Cargo.

4.2.12.8 un_type_code

The un_type_code specifies the type of the cargo. This shall be the number relating to each product as specified in IMO ID200E, IMO IA772E, and IMO IB520E.

4.2.12.9 user_def_cargo

The user_def_cargo specifies the type of cargo if other than one of the pre-defined cargo types. The user_def_cargo need not be specified for a particular Cargo.

4.2.13 Cargo_assignment

A Cargo_assignment is the allocation of a cargo to a space within a ship. Each Cargo_assignment is either a Compartment_cargo_assignment (see 4.2.43), a Deck_cargo_assignment (see 4.2.75), or a Unit_cargo_assignment (see 4.2.169).

The data associated with a Cargo_assignment are the following:

— allocated_weight;

— assignment_context;

— cargo_identifier.

4.2.13.1 allocated_weight

The allocated_weight specifies the weight and centre of gravity of the cargo with respect to the compartment or deck to which it is assigned. The allocated_weight need not be specified for a particular Cargo_assignment. See 4.3.13 for the application assertion.

4.2.13.2 assignment_context

The assignment_context specifies the context for the loading of cargo on the ship.

The value of assignment_context is one of the following:

— accommodation;

— cargo_load;

— consumables;

— stores;

— unspecified.

NOTE See 4.2.13.2.1 - 4.2.13.2.5 for the definition of each allowable value for assignment_context.

4.2.13.2.1 accommodation

The Cargo_assignment is the furnishings installed on the ship.

4.2.13.2.2 cargo_load

The Cargo_assignment is an allocation of Cargo for design analysis or for an actual voyage during the operation of the ship.

4.2.13.2.3 consumables

The Cargo_assigment is the allocation of consumables that may be depleted during a voyage.

4.2.13.2.4 stores

The Cargo_assignment is the allocation of stores for possible use by the passengers and crew on the journey.

4.2.13.2.5 unspecified

The Cargo_assignment is of no specific type. This can be used for defining theoretical loads for analytical purposes.

4.2.13.3 cargo_identifier

The cargo_identifier specifies the label used to identify each cargo that is assigned.

4.2.14 Cargo_bay_definition

A Cargo_bay_definition is a type of Definition (see 4.2.80) that represents a grid of positions within a compartment that are used to specify the location of placement of cargo within that compartment.

The data associated with a Cargo_bay_definition are the following:

— cargo_positions;

— defined_for.

4.2.14.1 cargo_positions

The cargo_positions specifies a set of Spacing_tables, which may be either Longitudinal_table, Transversal_table, or Vertical_table. These tables each provide a set of Spacing_positions, constrained in orientation, to locate cargo within spaces. There may be more than one cargo_positions for a Cargo_bay_definition. See 4.3.15 for the application assertion.

4.2.14.2 defined_for

The defined_for specifies the Compartment for which the Cargo_bay_definition is applicable. There may be more than one defined_for for a Cargo_bay_definition. See 4.3.14 for the application assertion.

4.2.15 Cargo_compartment_property

A Cargo_compartment_property is a type of Compartment_property (see 4.2.56) that describes properties for cargo capacities and cargo densities for which the cargo compartment is designed.

The data associated with a Cargo_compartment_property are the following:

— bulk_cargo_capacity;

— design_stowage_density.

4.2.15.1 bulk_cargo_capacity

The bulk_cargo_capacity specifies volumetric characteristic of a cargo compartment. The bulk_cargo_capacity need not be specified for a particular Cargo_compartment_property.

4.2.15.2 design_stowage_density

The design_stowage_density specifies the measure of the quantity per unit volume of the Bulk_cargo for which the cargo compartment is designed.

4.2.16 Cargo_footprint

A Cargo_footprint is the size and position of the area of contact of a cargo with the structure of the ship.

The data associated with a Cargo_footprint are the following:

— contact_material;

— shape;

— transferred_mass.

4.2.16.1 contact_material

The contact_material specifies the type of material that is in contact with the structure of the ship.

The value of contact_material is one of the following:

— metal;

— other;

— pneumatic;

— rubber.

NOTE See 4.2.16.1.1 - 4.2.16.1.4 for the definition of each allowable value for contact_material.

4.2.16.1.1 metal

Metal is in contact with the ship structure.

4.2.16.1.2 other

The material in contact with the ship structure is not specified.

4.2.16.1.3 pneumatic

Air filled material is in contact with the ship structure.

4.2.16.1.4 rubber

Solid rubber is in contact with the ship structure.

4.2.16.2 shape

The shape specifies the definition of the shape of the footprint. The footprint bounding curve is relative to the local coordinate system.

4.2.16.3 transferred_mass

The transferred_mass specifies the mass of the cargo that is transferred to the structure of the ship via the footprint.

4.2.17 Cargo_position

A Cargo_position is the position of a Unit_cargo in terms of either the bays in a compartment or on the deck of a ship or by a ship coordinate. Each Cargo_position may be one of the following: an Absolute_cargo_position (see 4.2.1), a Bay_position (see 4.2.7), or a Relative_cargo_position (see 4.2.137).

4.2.18 Carrier

A Carrier is a type of Shiptype (see 4.2.143) that represents a ship that transports goods or passengers.

The data associated with a Carrier are the following:

— has_type.

4.2.18.1 has_type

The has_type specifies the type of the carrier ship.

The value of has_type is one of the following:

— barge;

— barge_for_deck_loading;

— barge_for_liquefied_gas;

— barge_for_oil;

— barge_pontoon;

— bulk_carrier;

— car_carrier;

— car_ferry;

— cargo_ship_carrying_passengers;

— chemical_tanker;

— chemical_tanker_type_1;

— container_carrier;

— cruise_liner;

— dry_cargo_vessel;

— ferry;

— gas_carrier;

— general_cargo_carrier;

— highspeedcraft_cargo;

— highspeedcraft_passenger;

— hydrofoil;

— liquefied_gas_tanker;

— LNG_carrier;

— LPG_carrier;

— oil_tanker;

— ore_carrier;

— passenger_vessel;

— product_tanker;

— refrigerated_cargo_carrying_ship;

— roro_vessel;

— tanker_for_refrigerated_fruit_juice;

— user_defined.

NOTE See 4.2.18.1.1 - 4.2.18.1.31 for the definition of each allowable value for has_type.

4.2.18.1.1 barge

The ship is a carrier that has no machinery for self-propulsion and is designed for transporting cargo.

4.2.18.1.2 barge_for_deck_loading

The ship is a carrier designed as a service platform for loading decks on ships.

4.2.18.1.3 barge_for_liquefied_gas

The ship is a carrier that has no machinery for self-propulsion and is designed for the carriage of liquefied gas.

4.2.18.1.4 barge_for_oil

The ship is a carrier that has no machinery for self-propulsion and is designed for the carriage of oil .

4.2.18.1.5 barge_pontoon

The ship is a carrier with pontoons coupled to a barge that serves as a floating utility platform.

4.2.18.1.6 bulk_carrier

The ship is a cargo carrier constructed for the transportation of bulk cargo.

4.2.18.1.7 car_carrier

The ship is a carrier used for transportation of cars from the manufacturer to the end-user.

4.2.18.1.8 car_ferry

The ship is a carrier constructed for the transport of automobiles.

4.2.18.1.9 cargo_ship_carrying_passengers

The ship is a cargo carrier constructed for the transport of passengers.

4.2.18.1.10 chemical_tanker

The ship is a cargo carrier specially designed to transport chemicals.

4.2.18.1.11 chemical_tanker_type_1

The ship is a cargo carrier specially designed to transport Type 1 chemicals.

4.2.18.1.12 container_carrier

The ship is a cargo carrier constructed for the transport of containers.

4.2.18.1.13 cruise_liner

The ship is a carrier constructed for the transport of passengers for pleasure.

4.2.18.1.14 dry_cargo_vessel

The ship is a cargo carrier constructed to transport dry, loose cargo.

EXAMPLE Types of dry cargo are grain or coal.

4.2.18.1.15 ferry

The ship is a carrier constructed for the transport of passengers, coaches, lorries, and trains across water.

NOTE The ferry could be designed for only one type of cargo such as passengers.

4.2.18.1.16 gas_carrier

The ship is a cargo carrier specially designed for transportation of gaseous products.

4.2.18.1.17 general_cargo_carrier

The ship is a cargo carrier specially designed for transportation of general cargo.

NOTE The general_cargo_carrier carries any dry cargo which is carried loose such as grain or coal.

4.2.18.1.18 highspeedcraft_cargo

The ship is a carrier designed for carrying cargo at high speeds.

4.2.18.1.19 highspeedcraft_passenger

The ship is a carrier designed for carrying passengers at high speeds.

4.2.18.1.20 hydrofoil

The ship is a carrier designed to achieve speed by raising the hull of the vessel out of the water on hydrofoil surfaces, thus avoiding hull drag and wave resistance.

4.2.18.1.21 liquefied_gas_tanker

The ship is a cargo carrier specially designed for transportation of liquefied gas products.

4.2.18.1.22 LNG_carrier

The ship is a carrier specially designed to transport liquid natural gas.

4.2.18.1.23 LPG_carrier

The ship is a carrier specially designed to transport liquid petroleum gas.

4.2.18.1.24 oil_tanker

The ship is a carrier specially designed for the transportation of oil.

4.2.18.1.25 ore_carrier

The ship is a carrier constructed for the transportation of ore.

4.2.18.1.26 passenger_vessel

The ship is a carrier constructed for the transport of passengers.

4.2.18.1.27 product_tanker

The ship is a cargo carrier specially designed for transportation of oil products.

4.2.18.1.28 refrigerated_cargo_carrying_ship

The ship is a carrier constructed for transporting freight which requires cooling or chilling.

4.2.18.1.29 roro_vessel

The ship is a carrier constructed for the transport of roll-on, roll-off type vehicles.

4.2.18.1.30 tanker_for_refrigerated_fruit_juice

The ship is a tanker carrier specially designed for transportation of fruit juices which require cooling or chilling.

4.2.18.1.31 user_defined

The ship is any carrier type that is other than one of the pre-defined types. Details can be found in the description attribute for Shiptype.

4.2.19 Centre_location

The Centre_location is a location in terms of the three principal axis directions of the ship global coordinate system.

The data associated with a Centre_location are the following:

— longitudinal_location;

— transversal_location;

— vertical_location.

4.2.19.1 longitudinal_location

The longitudinal_location specifies the distance along the X axis from the origin of the global coordinate system of the ship.

4.2.19.2 transversal_location

The transversal_location specifies the distance along the Y axis from the origin of the global coordinate system of the ship.

4.2.19.3 vertical_location

The vertical_location specifies the distance along the Z axis from the origin of the global coordinate system of the ship.

4.2.20 Change

A Change is a type of Item (see 4.2.109) that represents the focus of all stages associated with a potential or actual change to the product model resulting from a customer or design organization change order. The change may or may not result in modifications to the product model data. Any planned or actual changes to the product model are documented in the associated Change_definitions.

The data associated with a Change are the following:

— the_class.

4.2.20.1 the_class

The the_class specifies the qualification of the organizational role of the change.

EXAMPLE Headquater Modification Request or Engineering Change Proposal.

4.2.21 Change_definition

A Change_definition is a type of Definition (see 4.2.80) that is the generalization of the major discrete stages of a Change. Each Change_definition is either a Change_request (see 4.2.25), a Change_plan (see 4.2.23), or a Change_realization (see 4.2.24).

The data associated with a Change_definition are the following:

— author;

— date_time;

— defined_for;

— local_units.

4.2.21.1 author

The author specifies the person or organization responsible for the Change activities during the period lasting from the end of the previous, if it exists, up to the end of this Change_definition.

4.2.21.2 date_time

The date_time specifies the date and time when the state of the Change_definition was reached.

4.2.21.3 defined_for

The defined_for specifies the Change to which the Change_definition is applicable. There may be more than one defined_for for a Change_definition. See 4.3.17 for the application assertion.

4.2.21.4 local_units

The local_units specifies the units that the Definition makes use of if they differ from the units globally defined for the ship.

NOTE For Change_definition, the local_units attribute inherited from Definition (see 4.2.80.3) has been redeclared in the ARM to be a set of zero, which is interpreted to mean that the attribute shall not be populated in the AIM.

4.2.22 Change_impact

A Change_impact defines the effect a Change will cause or has caused.

The data associated with a Change_impact are the following:

— impact.

4.2.22.1 impact

The impact specifies the effect of a Change in terms of the creation, modification or deletion of some Definitions, Item_structures, or Item_relationships. There may be more than one impact for a Change_impact. See 4.3.18 for the application assertion.

4.2.23 Change_plan

A Change_plan is a type of Change_definition (see 4.2.21) that defines the proposed solution for a Change. It is the basis for the activities, the Versionable_object_change_events, necessary to implement the Change in the product model.

The data associated with a Change_plan are the following:

— checks;

— chosen_solution_for;

— planned_impact.

4.2.23.1 checks

The checks specifies the verifications planned for the Change. The checks need not be specified for a particular Change_plan. There may be more than one checks for a Change_plan. See 4.3.21 for the application assertion.

4.2.23.2 chosen_solution_for

The chosen_solution_for specifies identification of the Change_request for which a Change_plan is applicable. See 4.3.20 for the application assertion.

4.2.23.3 planned_impact

The planned_impact specifies the estimated or calculated effects of the Change. This impact is usually chosen from the set of solution alternatives of a Change_request. See 4.3.19 for the application assertion.

4.2.24 Change_realization

A Change_realization is a type of Change_definition (see 4.2.21) that defines the actual, observed effects of a Change.

The data associated with a Change_realization are the following:

— checks;

— impact;

— realization_of.

4.2.24.1 checks

The checks specifies organizational approval of the product model revisions made to implement the Change. The checks need not be specified for a particular Change_realization. There may be more than one checks for a Change_realization. See 4.3.24 for the application assertion.

4.2.24.2 impact

The impact specifies identification of the revisions made to the product model. See 4.3.22 for the application assertion.

4.2.24.3 realization_of

The realization_of specifies the Change_plan for which a product model Change is being implemented. See 4.3.23 for the application assertion.

4.2.25 Change_request

A Change_request is a type of Change_definition (see 4.2.21) that is the first phase of a Change, where the need for a Change and possible solution alternatives are established.

The data associated with a Change_request are the following:

— addressee;

— initiator;

— problem;

— solution_alternatives;

— solution_description.

4.2.25.1 addressee

The addressee specifies the person and organization the request is addressed to. The addressee need not be specified for a particular Change_request.

4.2.25.2 initiator

The initiator specifies the person and organization from which the change request comes.

4.2.25.3 problem

The problem specifies a description of the problem having induced the request.

4.2.25.4 solution_alternatives

The solution_alternatives specifies alternative solutions proposed to solve the problem. A solution is described in terms of the effect on Versionable_objects. The solution_alternatives need not be specified for a particular Change_request. There may be more than one solution_alternatives for a Change_request. See 4.3.25 for the application assertion.

4.2.25.5 solution_description

The solution_description specifies a description of one or more possible solutions for the problem. This textual description should be present, if the solution_alternatives are not yet established, or may enhance the information provided by the solution_alternatives. The solution_description need not be specified for a particular Change_request.

4.2.26 Check

A Check is a type of Event (see 4.2.91) that defines the details of a planned or fulfilled approval within an organization for a Change_plan or a Change_realization.

4.2.27 Class_and_statutory_designation

A Class_and_statutory_designation is a type of General_characteristics_definition (see 4.2.103) that specifies the identification given to the ship by the classification society for the purpose of design, manufacture, and in-service approval, and applicable statutory documents which govern the design.

The data associated with a Class_and_statutory_designation are the following:

— class_number;

— local_units;

— the_class;

— the_statutory.

4.2.27.1 class_number

The class_number specifies a classification society specific identifier to a ship.

4.2.27.2 local_units

The local_units specifies the units that the Definition makes use of if they differ from the units globally defined for the ship.

NOTE For Class_and_statutory_designation, the local_units attribute inherited from Definition (see 4.2.80.3) has been redeclared in the ARM to be a set of zero, which is interpreted to mean that the attribute shall not be populated in the AIM.

4.2.27.3 the_class

The the_class specifies the applicable Class_notation with information about the ship type and the Cargo. See 4.3.26 for the application assertion.

4.2.27.4 the_statutory

The the_statutory specifies the set of national and international regulations and standards with which the ship is intended to comply. See 4.3.27 for the application assertion.

4.2.28 Class_bulk_load_requirement_definition

A Class_bulk_load_requirement_definition is a type of Class_compartment_requirement_definition (see 4.2.29) that describes properties for bulk load compartments that are required for performing a ship design approval by rules.

The data associated with a Class_bulk_load_requirement_definition are the following:

— angle_of_repose;

— bulk_cargo_mass;

— permeability;

— top_of_hatch.

4.2.28.1 angle_of_repose

The angle_of_repose specifies the angle subtended with the horizontal by the upper surface of the conic pile made by the Bulk_cargo when loaded into a hold.

4.2.28.2 bulk_cargo_mass

The bulk_cargo_mass specifies the maximum mass of the Bulk_cargo that is supposed to be carried in the compartment.

4.2.28.3 permeability

The permeability specifies the percentage of the total volume of a compartment that is not occupied by the cargo.

4.2.28.4 top_of_hatch

The top_of_hatch specifies the height from baseline to the highest point of the hold including hatchway.

4.2.29 Class_compartment_requirement_definition

A Class_compartment_requirement_definition is a type of Design_requirement (see 4.2.83) that describes properties of a compartment that are required for performing a ship design approval by rules. Each Class_compartment_requirement_definition is either a Class_bulk_load_requirement_-definition (see 4.2.28), or a Class_tank_requirement_definition (see 4.2.33).

The data associated with a Class_compartment_requirement_definition are the following:

— ambient_temperature;

— cargo_density;

— cargo_height;

— coating;

— damage_waterline;

— defined_for;

— max_pressure;

— max_temperature;

— min_pressure;

— min_temperature.

4.2.29.1 ambient_temperature

The ambient_temperature specifies whether or not the cargo is to be transported without air conditioning.

NOTE A value of true indicates that air conditioning is not required.

4.2.29.2 cargo_density

The cargo_density specifies the highest cargo density for ships with tanks for liquid; otherwise, it is the stowage rate of bulk load, the total cargo capacity of the ship divided by the total hold volume.

4.2.29.3 cargo_height

The cargo_height specifies the filling height in the case of liquid load; otherwise, it is the height from the baseline to the top of the hatch in the case of bulk load.

4.2.29.4 coating

The coating specifies the indication whether the compartment is fully coated or not.

4.2.29.5 damage_waterline

The damage_waterline specifies the height from the baseline to the waterline in a damaged condition of the compartment.

4.2.29.6 defined_for

The defined_for specifies the compartment for which the Class_compartment_requirement_-definition shall be applicable. There may be more than one defined_for for a Class_compartment_-requirement_definition. See 4.3.28 for the application assertion.

4.2.29.7 max_pressure

The max_pressure specifies the maximum pressure inside the compartment.

4.2.29.8 max_temperature

The max_temperature specifies the maximum temperature inside the compartment.

4.2.29.9 min_pressure

The min_pressure specifies the minimum pressure inside the compartment.

4.2.29.10 min_temperature

The min_temperature specifies the minimum temperature inside the compartment.

4.2.30 Class_deck_load_requirement_definition

A Class_deck_load_requirement_definition is a type of Design_requirement (see 4.2.83) that describes properties for deck loads that are required for performing a ship design approval by rules.

The data associated with a Class_deck_load_requirement_definition are the following:

— defined_for;

— grab_weight;

— stowage_height;

— stowage_rate;

— vehicle_load.

4.2.30.1 defined_for

The defined_for specifies the Deck_zone for which the Class_deck_load_requirement_definition shall be applicable. There may be more than one defined_for for a Class_deck_load_-requirement_definition. See 4.3.29 for the application assertion.

4.2.30.2 grab_weight

The grab_weight specifies the maximum weight of grabs that shall be used for unloading of cargo. The grab_weight need not be specified for a particular Class_deck_load_requirement_definition.

4.2.30.3 stowage_height

The stowage_height specifies the maximum height of cargo that may be stowed on the deck. The stowage_height need not be specified for a particular Class_deck_load_requirement_definition.

4.2.30.4 stowage_rate

The stowage_rate specifies the maximum density with which cargo may be stowed on the deck. The stowage_rate need not be specified for a particular Class_deck_load_requirement_definition.

4.2.30.5 vehicle_load

The vehicle_load specifies the maximum load that may be imposed by a vehicle. The vehicle_load need not be specified for a particular Class_deck_load_requirement_definition. See 4.3.30 for the application assertion.

4.2.31 Class_notation

A Class_notation specifies the notations given to the hull and machinery of the ship by the classification society as a result of its approval activities during the design, manufacture, and in-service maintenance of the ship.

The data associated with a Class_notation are the following:

— approval_required_for_heavy_cargo;

— approval_required_for_oil_cargo;

— approval_required_loading_unloading_aground;

— approval_required_loading_unloading_grabs;

— class_notations_hull;

— class_notations_machinery;

— class_society;

— ice_class_notation;

— service_area;

— service_factor.

4.2.31.1 approval_required_for_heavy_cargo

The approval_required_for_heavy_cargo specifies a flag indicating whether or not approval for special strengthening for heavy cargoes is necessary. These notations are valid for bulk carriers to indicate the distribution of loads across the cargo holds. The approval_required_for_heavy_cargo need not be specified for a particular Class_notation.

The value of approval_required_for_heavy_cargo is one of the following:

— HC;

— HC_E;

— HC_EA.

NOTE See 4.2.31.1.1 - 4.2.31.1.3 for the definition of each allowable value for approval_required_for_-heavy_cargo.

4.2.31.1.1 HC

The ship must be strengthened for heavy cargo. Heavy bulk cargo may be unevenly distributed among the cargo holds.

4.2.31.1.2 HC_E

The ship must be strengthened for heavy cargo. In addition to HC, a non-homogeneous loading condition with empty holds on full draught is approved. The approved combination of empty holds is added to the notation.

EXAMPLE Holds 2, 3, and 5 are empty.

4.2.31.1.3 HC_EA

The ship must be strengthened for heavy cargo. Any cargo hold may be empty at full draught. The approved combinations of empty holds are added to the notation.

EXAMPLE Holds 2, 3, and 5 are empty, or Hold 4 is empty.

4.2.31.2 approval_required_for_oil_cargo

The approval_required_for_oil_cargo specifies a flag indicating whether or not approval is required for the carriage of oil cargoes. A value of true indicates approval is required.

4.2.31.3 approval_required_loading_unloading_aground

The approval_required_loading_unloading_aground specifies a flag indicating whether or not approval for loading and unloading aground is necessary. A value of true indicates approval is required.

4.2.31.4 approval_required_loading_unloading_grabs

The approval_required_loading_unloading_grabs specifies a flag indicating whether or not approval for loading and unloading using grabs is necessary. A value of true indicates approval is required.

4.2.31.5 class_notations_hull

The class_notations_hull specifies the notation given to the hull of the ship by the classification society as a result of its approval activities done on the hull. There may be more than one class_notations_hull for a Class_notation.

4.2.31.6 class_notations_machinery

The class_notations_machinery specifies the notation given to the machinery on the ship by the classification society as a result of its approval activities done on the machinery. There may be more than one class_notations_machinery for a Class_notation.

4.2.31.7 class_society

The class_society specifies the name and organizational details of the classification society whose rules and regulations are being used to assess the ship.

4.2.31.8 ice_class_notation

The ice_class_notation specifies the type of class notation given to the ship indicating the ice conditions in which the ship has been approved to operate. The ice_class_notation need not be specified for a particular Class_notation.

4.2.31.9 service_area

The service_area specifies the area or route in which the ship operates. This may include information about waterway, wave, weather, and wind conditions.

4.2.31.10 service_factor

The service_factor specifies the service area of the ship and the waves that occur in that area. The service_factor should be in the range of 0.5 to 1. The service_factor need not be specified for a particular Class_notation.

4.2.32 Class_parameters

A Class_parameters is a type of General_characteristics_definition (see 4.2.103) that specifies design and intended performance characteristics of the ship in accordance with classification society rules and statutory regulations.

The data associated with a Class_parameters are the following:

— block_coefficient_class;

— design_speed_ahead;

— design_speed_astern;

— length_class;

— length_solas;

— scantlings_draught.

4.2.32.1 block_coefficient_class

The block_coefficient_class specifies the ratio of the moulded displacement volume to the volume of a block that has its length equal to the length_class, its breadth equal to the moulded_breadth, and its depth equal to the scantlings_draught.

4.2.32.2 design_speed_ahead

The design_speed_ahead specifies the forward speed at which the ship is designed to operate.

4.2.32.3 design_speed_astern

The design_speed_astern specifies the reverse speed at which the ship is designed to operate.

4.2.32.4 length_class

The length_class specifies a length measurement for the ship that is defined in classification society rules.

4.2.32.5 length_solas

The length_solas specifies a length measurement for the ship measured in accordance with IMO IC110E.

4.2.32.6 scantlings_draught

The scantlings_draught specifies the summer load draught used by the classification society in its calculations for structural integrity and strength.

4.2.33 Class_tank_requirement_definition

A Class_tank_requirement_definition is a type of Class_compartment_requirement_definition (see 4.2.29) that describes properties for tanks that are required for performing a ship design approval by rules.

The data associated with a Class_tank_requirement_definition are the following:

— free_surface_parameters;

— overflow_height;

— partial_filling;

— pressure_relief_setting.

4.2.33.1 free_surface_parameters

The free_surface_parameters specifies the length and breadth of the free surface in the tank if there is any free surface. The free_surface_parameters need not be specified for a particular Class_tank_requirement_definition.

4.2.33.2 overflow_height

The overflow_height specifies the maximum filling height in a tank just before overflow.

NOTE This is also referred to as the top of the air pipe.

4.2.33.3 partial_filling

The partial_filling specifies whether or not the compartment may be partially filled.

4.2.33.4 pressure_relief_setting

The pressure_relief_setting specifies the pressure at which the inert gas relief valve will open.

4.2.34 Coating

A Coating is the definition of the protective coating applied to the ship structure to protect it from corrosion from water or cargoes. Each Coating is either a Corrosion_control_coating (see 4.2.68), a Fire_safe_coating (see 4.2.95), or a Primer_coating (see 4.2.134).

The data associated with a Coating are the following:

— certification;

— description;

— dry_film_thickness;

— manufacturer;

— name;

— number_of_coats;

— surface_preparation.

4.2.34.1 certification

The certification specifies whether the Coating, and all the given attributes, has been certified for the specified use by an Organization. The certification need not be specified for a particular Coating. There may be more than one certification for a Coating. See 4.3.31 for the application assertion.

EXAMPLE An Organization such as a classification society can issue a Coating_certification.

4.2.34.2 description

The description specifies a brief description of the Coating.

4.2.34.3 dry_film_thickness

The dry_film_thickness specifies the thickness of the Coating film.

4.2.34.4 manufacturer

The manufacturer specifies the company that makes the Coating.

4.2.34.5 name

The name specifies the trade name of the Coating.

4.2.34.6 number_of_coats

The number_of_coats specifies the number of coats which must be applied to the surface.

4.2.34.7 surface_preparation

The surface_preparation specifies the codes used for the grade of preparation required for steel surfaces prior to application of Coating. These are defined in ISO 8501-1.

The value of surface_preparation is one of the following:

— Fl;

— Sa1;

— Sa2;

— Sa2_5;

— Sa3;

— St2;

— St3.

NOTE See 4.2.34.7.1 - 4.2.34.7.7 for the definition of each allowable value for surface_preparation.

4.2.34.7.1 Fl

The surfaces are to be prepared by flame cleaning prior to application of the coating.

4.2.34.7.2 Sa1

The surfaces are to be prepared by light blast cleaning prior to application of the coating.

4.2.34.7.3 Sa2

The surfaces are to be prepared with a thorough blast cleaning prior to application of the coating.

4.2.34.7.4 Sa2_5

The surfaces are to be prepared with a very thorough blast cleaning prior to application of the coating.

4.2.34.7.5 Sa3

The surfaces are to be prepared with a blast cleaning to visually clean steel prior to application of the coating.

4.2.34.7.6 St2

The surfaces are to be prepared with a thorough hand and power tool cleaning prior to application of the coating.

4.2.34.7.7 St3

The surfaces are to be prepared with a very thorough hand and power tool cleaning prior to application of the coating.

4.2.35 Coating_certification

A Coating_certification identifies the organization and time limit on the certification of a Coating for usage.

The data associated with a Coating_certification are the following:

— certifying_organization;

— expiry_date.

4.2.35.1 certifying_organization

The certifying_organization specifies the organization that certified the Coating for use.

4.2.35.2 expiry_date

The expiry_date specifies the date and time when the certification shall expire.

4.2.36 Coating_level

A Coating_level is the extent of coating in a compartment.

The data associated with a Coating_level are the following:

— lower_extent;

— upper_extent.

4.2.36.1 lower_extent

The lower_extent specifies the percentage of the height from the base of the compartment to the lowest level of the coating.

4.2.36.2 upper_extent

The upper_extent specifies the percentage of the height from the base of the compartment to the highest level of the coating.

4.2.37 Compartment

A Compartment is a type of Space (see 4.2.145) that is a physical subdivision of space on a ship, designed to hold dry or liquid cargo, fuel, water, passengers, crew, machinery, and equipment. Each Compartment may have one or more functional definitions to specify the intended function or functions of the Compartment, as well as one or more design definitions, which specify the boundaries of the Compartment as well as its physical and geometric properties.

4.2.38 Compartment_abbreviated_name

A Compartment_abbreviated_name is a type of General_compartment_property (see 4.2.104) that specifies a short, compact, efficient reference to a particular compartment on a ship.

The data associated with a Compartment_abbreviated_name are the following:

— name.

4.2.38.1 name

The name specifies a short, compact, efficient means of referring to a particular compartment on a ship. The abbreviated name may or may not have an embedded meaning.

EXAMPLE The abbreviation of 6-55-1-F for the freshwater tank above the 6th deck, beginning at frame 55, and located on the starboard side.

NOTE In Naval vessels, abbreviated names are commonly used and are encoded such that they indicate the type of cargo, and the vertical, transverse, and longitudinal position.

4.2.39 Compartment_acceleration

A Compartment_acceleration is a type of General_compartment_property (see 4.2.104) that is the acceleration of gravity of the compartment while the ship is underway.

The data associated with a Compartment_acceleration are the following:

— acceleration_g_force.

4.2.39.1 acceleration_g_force

The acceleration_g_force specifies a measure of the allowable acceleration force, expressed as a ratio compared to the acceleration of gravity, allowed for a compartment. This force is represented as a single value and governs the accelerations in all three principal directions: vertical, longitudinal, and transverse.

EXAMPLE A value of 1.5 would represent 1.5 times the acceleration of gravity (9.81 metres per second per second), or 14.72 metres per second per second.

4.2.40 Compartment_access_authorization

A Compartment_access_authorization is a type of General_compartment_property (see 4.2.104) that is an indication of a limit on allowed accessibility to a compartment based on rank.

The data associated with a Compartment_access_authorization are the following:

— authorization_classification;

— user_defined_value.

4.2.40.1 authorization_classification

The authorization_classification specifies a type of crew restriction placed on the use of the compartment.

The value of authorization_classification is one of the following:

— crew_only;

— officers_only;

— restricted;

— unrestricted;

— user_defined.

NOTE See 4.2.40.1.1 - 4.2.40.1.5 for the definition of each allowable value for authorization_classification.

4.2.40.1.1 crew_only

The compartment is designated for use by crew.

4.2.40.1.2 officers_only

The compartment is designated for use by officers.

4.2.40.1.3 restricted

The compartment is designated as a restricted area.

4.2.40.1.4 unrestricted

The compartment access is not limited to any particular group.

4.2.40.1.5 user_defined

The authorization_classification of the compartment is defined by the user_defined_value attribute.

4.2.40.2 user_defined_value

The user_defined_value specifies classification for the case of authorization_classification having the value of user_defined. The user_defined_value need not be specified for a particular Compartment_access_authorization.

4.2.41 Compartment_air_circulation_rate

A Compartment_air_circulation_rate is a type of General_compartment_property (see 4.2.104) that defines the volume of air changes for the compartment per unit of time.

The data associated with a Compartment_air_circulation_rate are the following:

— air_circulation_rate.

4.2.41.1 air_circulation_rate

The air_circulation_rate specifies measure of the volume of air changed for the compartment per unit of time.

NOTE This value is used by applications performing HVAC load analyses.

4.2.42 Compartment_area_property

A Compartment_area_property is a type of General_compartment_property (see 4.2.104) that defines different types of area properties for a compartment. Each Compartment_area_property is either a Compartment_vertical_longitudinal_cross_sectional_area_property (see 4.2.62), a Compartment_-vertical_transverse_cross_sectional_area_property (see 4.2.63), a Compartment_horizontal_cross_-sectional_area_property (see 4.2.49), a Compartment_stiffened_surface_area_property (see 4.2.59), or a Compartment_unstiffened_surface_area_property (see 4.2.61).

4.2.43 Compartment_cargo_assignment

A Compartment_cargo_assignment is a type of Cargo_assignment (see 4.2.13) that is the allocation of a cargo to a compartment or space within a ship. Each Compartment_cargo_assignment may be one of the following: a Bulk_cargo_assignment (see 4.2.9), a Gaseous_cargo_assignment (see 4.2.101), or a Liquid_cargo_assignment (see 4.2.116).

The data associated with a Compartment_cargo_assignment are the following:

— cargo;

— compartment.

4.2.43.1 cargo

The cargo specifies the type of cargo that has been loaded. The cargo need not be specified for a particular Compartment_cargo_assignment. See 4.3.32 and 4.3.34 for the application assertions.

4.2.43.2 compartment

The compartment specifies the compartment into which the cargo has been loaded. See 4.3.33 for the application assertion.

4.2.44 Compartment_coating

A Compartment_coating is a type of General_compartment_property (see 4.2.104) that specifies the type of painting or coating required for a compartment.

The data associated with a Compartment_coating are the following:

— corrosion_protection.

4.2.44.1 corrosion_protection

The corrosion_protection specifies a reference to the collection of properties for protecting compartment internals and boundaries from corrosion. See 4.3.35 for the application assertion.

4.2.45 Compartment_design_definition

A Compartment_design_definition is a type of Design_definition (see 4.2.82) that defines a version of a Compartment from a design perspective.

The data associated with a Compartment_design_definition are the following:

— boundaries;

— defined_for;

— properties;

— representations.

4.2.45.1 boundaries

The boundaries specifies External_instance_references to the ISO 10303-216 Moulded_form objects or ISO 10303-218 Structural_system objects that bound the Compartment. The boundaries need not be specified for a particular Compartment_design_definition. There may be more than one boundaries for a Compartment_design_definition. See 4.3.38 for the application assertion.

4.2.45.2 defined_for

The defined_for specifies the Compartment for which a Compartment_design_definition is applicable. There may be more than one defined_for for a Compartment_design_definition. See 4.3.36 for the application assertion.

4.2.45.3 properties

The properties specifies a collection of properties applicable to or derived from the design of a Compartment. The properties need not be specified for a particular Compartment_design_definition. There may be more than one properties for a Compartment_design_definition. See 4.3.37 for the application assertion.

4.2.45.4 representations

The representations specifies the Compartment_shape_representations for a Compartment_design_-definition. The representations need not be specified for a particular Compartment_design_-definition. There may be more than one representations for a Compartment_design_definition. See 4.3.39 for the application assertion.

4.2.46 Compartment_design_requirement

A Compartment_design_requirement is a type of Design_requirement (see 4.2.83) that defines a type of specification that represents a constraint placed on a design.

NOTE These requirements could be in the form of a reference to a set of rules or formula, such as design specifications, classification society rules, or welding society rules, or it may be an explicit requirement, such as an electrical requirement for 110 volt power or an HVAC requirement for operating temperatures in the range of 50 degrees to 90 degrees Farenheit.

The data associated with a Compartment_design_requirement are the following:

— defined_for;

— description;

— requirement_type.

4.2.46.1 defined_for

The defined_for specifies the Compartment or Zone for which the Compartment_design_-requirement is applicable. There may be more than one defined_for for a Compartment_design_-requirement. See 4.3.40 for the application assertion.

4.2.46.2 description

The description specifies a textual description of the requirement that is to be met.

4.2.46.3 requirement_type

The requirement_type specifies an indicator used to denote the source placing the design requirement on the space. This indicator identifies the discipline or design function that governs some aspect of the design or operation of the space.

The value of requirement_type is one of the following:

— combat_system;

— electrical;

— electronic;

— hvac;

— naval_architecture;

— outfit_furnishing;

— painting_coating;

— piping;

— structural;

— user_defined.

NOTE See 4.2.46.3.1 - 4.2.46.3.10 for the definition of each allowable value for requirement_type.

4.2.46.3.1 combat_system

The Compartment design requirement originates from the combat system discipline.

4.2.46.3.2 electrical

The Compartment design requirement originates from the electrical discipline.

4.2.46.3.3 electronic

The Compartment design requirement originates from the electronic discipline.

4.2.46.3.4 hvac

The Compartment design requirement originates from the hvac discipline.

4.2.46.3.5 naval_architecture

The Compartment design requirement originates from the naval architecture discipline.

4.2.46.3.6 outfit_furnishing

The Compartment design requirement originates from the outfit and furnishings discipline.

4.2.46.3.7 painting_coating

The Compartment design requirement originates from the painting or coating discipline.

4.2.46.3.8 piping

The Compartment design requirement originates from the piping discipline.

4.2.46.3.9 structural

The Compartment design requirement originates from the structural discipline.

4.2.46.3.10 user_defined

The Compartment_design_requirement originates from some other source than one of the specified design disciplines.

4.2.47 Compartment_functional_definition

A Compartment_functional_definition is a type of Functional_definition (see 4.2.99) that defines the functional role of a Compartment. The role may be a pre-defined one that corresponds to the intended use of the Compartment or may be a user-defined one.

The data associated with a Compartment_functional_definition are the following:

— defined_for;

— used_for.

4.2.47.1 defined_for

The defined_for specifies the Compartment for which the Compartment_functional_definition is applicable. There may be more than one defined_for for a Compartment_functional_definition. See 4.3.41 for the application assertion.

4.2.47.2 used_for

The used_for specifies the intended use of a Compartment. The pre-defined functions correspond to either a general compartment use or a specialized function. The user_defined value allows identification of a type not specified by one of the pre-defined values.

The value of used_for is one of the following:

— access_trunk;

— aft_peak_tank;

— auxiliary_engine_room;

— ballast_tank;

— battery_room;

— berthing_compartment;

— boiler_room;

— bottom_wing_tank;

— bow_thruster_room;

— cabin;

— cargo_compartment;

— centre_tank;

— chainlocker;

— cofferdam;

— compressor_room;

— control;

— crossover_tank;

— deck_tank;

— deep_tank;

— diving_well;

— double_bottom_and_side_tank;

— double_bottom_tank;

— double_side_tank;

— drill_well;

— duct_keel;

— electric_motor_room;

— emergency_fire_pump_room;

— equipment_room;

— escape_trunk;

— fore_peak_tank;

— forecastle;

— habitable_compartment;

— heeling_tank;

— hopper_tank;

— insulated_tank;

— lounge;

— machinery_compartment;

— main_engine_room;

— medical;

— passageway;

— poop;

— pump_room;

— rudder_trunk;

— separator_room;

— settling_tank;

— shaft_tunnel;

— side_tank;

— side_wing_tank;

— stabiliser_room;

— stability_tank;

— steering_gear_room;

— stern_tank;

— stool_tank;

— tank;

— thruster_room;

— top_wing_tank;

— trimming_tank;

— trunk;

— user_defined;

— void;

— waterjet_room;

— wheelhouse;

— wing_tank.

NOTE See 4.2.47.2.1 - 4.2.47.2.63 for the definition of each allowable value for used_for.

4.2.47.2.1 access_trunk

The compartment is designed to be used as an access trunk.

4.2.47.2.2 aft_peak_tank

The compartment is a tank aft of the aftmost bulkhead of the ship.

4.2.47.2.3 auxiliary_engine_room

The compartment is designed to be used as an auxiliary engine room space.

4.2.47.2.4 ballast_tank

The compartment is a watertight compartment to hold water ballast.

4.2.47.2.5 battery_room

The compartment is designated for batteries.

4.2.47.2.6 berthing_compartment

The compartment is designed to be used as a berthing space.

4.2.47.2.7 boiler_room

The compartment is designated for boilers.

4.2.47.2.8 bottom_wing_tank

The compartment is a tank located between the ship side/bottom and a sloping longitudinal bulkhead.

4.2.47.2.9 bow_thruster_room

The compartment is designed to be used as the bow thruster room.

4.2.47.2.10 cabin

The compartment is designed to be used as a cabin space.

4.2.47.2.11 cargo_compartment

The compartment is designed to carry liquid, bulk, or containerized goods.

NOTE 1 These goods may be consumed during the voyage, as in the case of food or fuel, or they may be temporarily stored for transport between ports.

NOTE 2 A tank compartment may also be used for the storage or transportation of liquid cargo.

4.2.47.2.12 centre_tank

The compartment is a tank that occupies a large part of a ship cross section and that is symmetrical to the ship centre line.

EXAMPLE Crude oil tankers often have several centre tanks.

4.2.47.2.13 chainlocker

The compartment is where anchor chain is stored.

4.2.47.2.14 cofferdam

The compartment is a narrow void space between two bulkheads or floors that prevents leakage between the adjoining compartments.

4.2.47.2.15 compressor_room

The compartment is designated for compressors.

4.2.47.2.16 control

The compartment is designed to be used for ship command and control functions.

4.2.47.2.17 crossover_tank

The compartment is a tank used for dynamic stabilisation of the ship.

4.2.47.2.18 deck_tank

The compartment is an independent tank located on the main deck.

4.2.47.2.19 deep_tank

The compartment is a tank extending from the bottom or inner bottom up to, or higher than, the lower deck. A deep tank is often fitted with hatches so that they may also be used for dry cargo in lieu of fuel oil, ballast water, or liquid cargo.

4.2.47.2.20 diving_well

The compartment is a space for divers to exit to the sea through a hatch in the bottom of the vessel.

4.2.47.2.21 double_bottom_and_side_tank

The compartment is a tank located between the outer and inner bottom and bounded by an outer and inner side of the ship.

4.2.47.2.22 double_bottom_tank

The compartment is a tank located between the outer and inner bottom of the ship.

4.2.47.2.23 double_side_tank

The compartment is a tank between the outer ship side and the inner ship side.

4.2.47.2.24 drill_well

The compartment is a space for drilling through an opening in the bottom of the vessel.

4.2.47.2.25 duct_keel

The compartment is located between the bottom and the inner bottom running along the the centre line of the ship.

4.2.47.2.26 electric_motor_room

The compartment is designated for electric motors.

4.2.47.2.27 emergency_fire_pump_room

The compartment is designated for emergency fire pumps.

4.2.47.2.28 equipment_room

The compartment is designated to be used as an equipment room.

4.2.47.2.29 escape_trunk

The compartment is a vertical trunk fitted with a ladder to permit personnel to escape if trapped. Usually provided from the after end of the shaft tunnel to topside spaces.

4.2.47.2.30 fore_peak_tank

The compartment is a tank located forward of the foremost bulkhead of the ship.

4.2.47.2.31 forecastle

The compartment is a superstructure fitted at the extreme forward end of the upper deck.

4.2.47.2.32 habitable_compartment

The compartment is designed as a habitable space, which is primarily designated as suitable for occupancy by humans. Passenger safety and comfort are subject to international, national, class society, or other regulations usually covered by product specifications and applicable class and register notations.

4.2.47.2.33 heeling_tank

The compartment is a tank used for adjusting the heeling of the ship.

4.2.47.2.34 hopper_tank

The compartment is a tank located between the ship side/bottom and a sloping longitudinal bulkhead.

4.2.47.2.35 insulated_tank

The compartment is a container constructed to hold one or more thermally insulated tanks for liquids

4.2.47.2.36 lounge

The compartment is designed to be used as a lounge space.

4.2.47.2.37 machinery_compartment

The compartment is designed to contain machinery for the operation of the ship or in support of its mission.

EXAMPLE Engine room and bow thruster room are types of machinery compartments.

4.2.47.2.38 main_engine_room

The compartment is designed to be used as the main engine room.

4.2.47.2.39 medical

The compartment is designed to be used as a medical space.

4.2.47.2.40 passageway

The compartment is designed to be used as a passageway.

4.2.47.2.41 poop

The compartment is a superstructure fitted at the after end of the upper deck.

4.2.47.2.42 pump_room

The compartment is designated for pumps.

4.2.47.2.43 rudder_trunk

The compartment is the trunk housing the rudder shaft.

4.2.47.2.44 separator_room

The compartment is designated for separators.

4.2.47.2.45 settling_tank

The compartment is a fuel oil tank used for separating entrained water from oil.

4.2.47.2.46 shaft_tunnel

The compartment is a watertight enclosure for the propeller shafting, large enough to walk in, extending aft from the engine room to provide access and protection to the shafting. Also known as a shaft alley.

4.2.47.2.47 side_tank

The compartment is a tank located between the ship side and a longitudinal bulkhead.

4.2.47.2.48 side_wing_tank

The compartment is a tank located between the ship side/bottom and a longitudinal bulkhead.

4.2.47.2.49 stabiliser_room

The compartment is designated for stabilisers.

4.2.47.2.50 stability_tank

The compartment is a tank used for dynamic stabilisation of the ship.

4.2.47.2.51 steering_gear_room

The compartment is designated for steering gear.

4.2.47.2.52 stern_tank

The compartment is a tank located at the aftmost location in the ship.

4.2.47.2.53 stool_tank

The compartment is a tank in the closed hull structure at either upper or lower end of a transverse bulkhead.

4.2.47.2.54 tank

The compartment is designed to carry liquids used in the mission of the ship, or for the storage of liquid cargoes transported by the ship.

EXAMPLE Fuels for propulsion of the ship, potable water for the passengers and crew, waste products, petroleum product cargo, and fuel for aircraft supported by the ship are carried in tank compartments.

4.2.47.2.55 thruster_room

The compartment is designated for thrusters.

4.2.47.2.56 top_wing_tank

The compartment is a tank located between the ship side/deck and a sloping longitudinal bulkhead.

4.2.47.2.57 trimming_tank

The compartment is a tank located near the ends of a ship. Seawater or fuel oil is carried in such tanks as necessary to change trim.

4.2.47.2.58 trunk

The compartment is a vertical or inclined space or passage formed by bulkheads or casings, extending one or more deck heights, around openings in the decks, through which access can be obtained and cargo or stores handled, or ventilation provided without disturbing or interfering with the contents or arrangements of the adjoining spaces.

4.2.47.2.59 user_defined

The compartment function is other than one of the pre-defined values and is specified by the user_def_function attribute.

4.2.47.2.60 void

The compartment is designed as an inaccessible, closed space that is never used to carry cargo or to be regularly occupied by humans. The main uses of a void compartment are segregating the cargo and fluids that are necessary to operate the ship, or to provide emergency access to other spaces.

4.2.47.2.61 waterjet_room

The compartment is designated for waterjets.

4.2.47.2.62 wheelhouse

The compartment is designated for primary steerage and control of the ship. Also known as the bridge.

4.2.47.2.63 wing_tank

The compartment is a tank located well outboard adjacent to the side shell plating, often consisting of a continuation of the double bottom up the sides to a deck or flat.

4.2.48 Compartment_group

A Compartment_group defines the compartments and their associated volume that has been used in Tonnage measurement calculations.

The data associated with a Compartment_group are the following:

— compartment;

— tonnage_volume.

4.2.48.1 compartment

The compartment specifies the compartment or group of compartments that are used in the Tonnage_measurement calculation. The compartment need not be specified for a particular

Compartment_group. There may be more than one compartment for a Compartment_group. See 4.3.42 for the application assertion.

4.2.48.2 tonnage_volume

The tonnage_volume specifies the volume of the compartment or group of compartments that are used for the Tonnage_measurement calculation.

4.2.49 Compartment_horizontal_cross_sectional_area_property

A Compartment_horizontal_cross_sectional_area_property is a type of Compartment_area_property (see 4.2.42) that represents a two dimensional cross-sectional area for a compartment.

NOTE Typically this area is used to reserve space early in the design process, such as the area needed for placement of a large piece of equipment.

The data associated with a Compartment_horizontal_cross_sectional_area_property are the following:

— horizontal_cross_sectional_area.

4.2.49.1 horizontal_cross_section_area

The horizontal_cross_sectional_area specifies an area measurement on a plane parallel to the baseline plane.

4.2.50 Compartment_illumination

A Compartment_illumination is a type of General_compartment_property (see 4.2.104) that defines the lighting requirements for a compartment.

The data associated with a Compartment_illumination are the following:

— illumination_value.

4.2.50.1 illumination_value

The illumination_value specifies amount of lighting required for a compartment.

NOTE This value is used by applications in performing lighting analysis.

4.2.51 Compartment_insulation

A Compartment_insulation is a type of General_compartment_property (see 4.2.104) that identifies the type of thermal insulation required for a Compartment.

The data associated with a Compartment_insulation are the following:

— insulation_category;

— user_defined_value.

4.2.51.1 insulation_category

The insulation_category specifies an indicator used to denote what type of thermal insulation is to be applied to compartment boundaries to reduce the rate of heat transfer to or from heated, ventilated, and air-conditioned spaces; to reduce condensation; and to retard excessive temperature rise in the

event of fire in adjacent spaces. Thermal insulation may be installed in conjunction with antisweat treatments to reduce condensation, and to serve as a vapor barrier to prevent the insulation from absorbing condensation.

The value of insulation_category is one of the following:

— A;

— B;

— C;

— D;

— E;

— F;

— G;

— H;

— I;

— J;

— K;

— L;

— M;

— N;

— O;

— P;

— Q;

— R;

— user_defined.

NOTE See 4.2.51.1.1 - 4.2.51.1.19 for the definition of each allowable value for insulation_category.

4.2.51.1.1 A

The insulation material is fibrous-glass faced thermal insulation board.

4.2.51.1.2 B

The insulation material is fibrous-glass unfaced thermal felt.

4.2.51.1.3 C

The insulation material is fibrous-glass faced thermal and sound absorbing felt.

4.2.51.1.4 D

The insulation material is sheathing, consisting of perforated aluminum.

4.2.51.1.5 E

The insulation material is fibrous-glass tape.

4.2.51.1.6 F

The insulation material is latex adhesive.

4.2.51.1.7 G

The insulation material is epoxy adhesive.

4.2.51.1.8 H

The insulation material is aluminum alloy stud.

4.2.51.1.9 I

The insulation material is carbon steel stud.

4.2.51.1.10 J

The insulation material is aluminum alloy or steel spacers.

4.2.51.1.11 K

The insulation material is adhesive-attached studs.

4.2.51.1.12 L

The insulation material is elastomeric foam.

4.2.51.1.13 M

The insulation material is adhesive for securing polyimide foam thermal insulation panels.

4.2.51.1.14 N

The insulation material is polyimide foam faced thermal insulation panel.

4.2.51.1.15 O

The insulation material is closed cell foam.

4.2.51.1.16 P

The insulation material is polyimide foam.

4.2.51.1.17 Q

The insulation material is fibrous-glass.

4.2.51.1.18 R

The insulation material is acrylic tape.

4.2.51.1.19 user_defined

The insulation_category of the Compartment is defined by a user specified value.

4.2.51.2 user_defined_value

The user_defined_value specifies the description if the insulation_category has a value of user_defined. The user_defined_value need not be specified for a particular Compartment_-insulation.

4.2.52 Compartment_naval_administrative_property

A Compartment_naval_administrative_property is a type of Compartment_property (see 4.2.56) that represents a collection of identification and Compartment design parameters that are applicable only to the design of naval vessels. Each Compartment_naval_administrative_property is either a Compartment_ziplist_number (see 4.2.66), a Compartment_nuclear_classification (see 4.2.54), a Compartment_safety_class (see 4.2.57), or a Compartment_security_classification (see 4.2.58).

4.2.53 Compartment_noise_category

A Compartment_noise_category is a type of General_compartment_property (see 4.2.104) that defines the design requirements for the internal level of sound of a compartment.

The data associated with a Compartment_noise_category are the following:

— noise_category;

— user_defined_value.

4.2.53.1 noise_category

The noise_category specifies a single alphabetical character key used to denote whether special consideration is to be given to the compartment with respect to the internal level of sound.

The value of noise_category is one of the following:

— A;

— B;

— C;

— D;

— E;

— F;

— user_defined.

NOTE See 4.2.53.1.1 - 4.2.53.1.7 for the definition of each allowable value for noise_category.

4.2.53.1.1 A

The compartment shall be designed for intelligible speech-low noise.

4.2.53.1.2 B

The compartment shall be designed for comfort.

4.2.53.1.3 C

The compartment shall be designed for quiet.

4.2.53.1.4 D

The compartment shall be designed for deafness avoidance.

4.2.53.1.5 E

The compartment shall be designed for intelligible speech-high noise.

4.2.53.1.6 F

The compartment shall be designed for intelligible speech-topside.

4.2.53.1.7 user_defined

The noise_category of the compartment is defined by the user_defined_value attribute.

4.2.53.2 user_defined_value

The user_defined_value specifies the description if noise_category has a value of user_defined. The user_defined_value need not be specified for a particular Compartment_noise_category.

4.2.54 Compartment_nuclear_classification

A Compartment_nuclear_classification is a type of Compartment_naval_administrative_property (see 4.2.52) that specifies whether a compartment is designated as containing nuclear reactors or is used for storage or repair of nuclear weapons; otherwise the compartment is classified as non-nuclear.

The data associated with a Compartment_nuclear_classification are the following:

— nuclear_classification.

4.2.54.1 nuclear_classication

The nuclear_classification specifies an indicator used to denote whether the compartment is designated a nuclear or non-nuclear space. This designation applies to spaces specifically designed to contain such things as nuclear reactors as well as spaces used for the storage or repair of nuclear weapons.

The value of nuclear_classification is one of the following:

— non_nuclear;

— nuclear.

NOTE See 4.2.54.1.1 - 4.2.54.1.2 for the definition of each allowable value for nuclear_classification.

4.2.54.1.1 non_nuclear

The compartment does not contain nuclear propulsion systems nor is it used for storage or repair of nuclear weapons.

4.2.54.1.2 nuclear

The compartment contains nuclear propulsion systems or is used for storage or repair of nuclear weapons.

4.2.55 Compartment_occupancy

A Compartment_occupancy is a type of General_compartment_property (see 4.2.104) that specifies the design requirements for the number of people that are permitted to occupy a compartment.

The data associated with a Compartment_occupancy are the following:

— occupancy.

4.2.55.1 occupancy

The occupancy specifies the number of persons who are permitted to occupy a compartment simultaneously.

4.2.56 Compartment_property

A Compartment_property is a measure of some significant characteristic of a compartment associated with a specific context. Each Compartment_property is either a General_compartment_property (see 4.2.104), a Compartment_naval_administrative_property (see 4.2.52), a Cargo_compartment_-property (see 4.2.15), or a Tank_compartment_property (see 4.2.161).

The data associated with a Compartment_property are the following:

— context.

4.2.56.1 context

The context specifies an indicator used to associate a design meaning with a compartment property. The maximum and minimum contexts serve to define the design limits for the property, while the estimated, calculated, and measured contexts associate a degree of accuracy for the property value.

The value of context is one of the following:

— calculated;

— estimated;

— maximum;

— measured;

— minimum.

NOTE See 4.2.56.1.1 - 4.2.56.1.5 for the definition of each allowable value for context.

4.2.56.1.1 calculated

The Compartment_property specifies the calculated design value for the property.

4.2.56.1.2 estimated

The Compartment_property specifies the estimated design value for the property.

4.2.56.1.3 maximum

The Compartment_property specifies the maximum design values for the property.

4.2.56.1.4 measured

The Compartment_property specifies the measured as-built value for the property.

4.2.56.1.5 minimum

The Compartment_property specifies the minimum design value for the property.

4.2.57 Compartment_safety_class

A Compartment_safety_class is a type of Compartment_naval_administrative_property (see 4.2.52) that specifies the safety classification of a compartment with regards to a hazardous working environment for humans.

The data associated with a Compartment_safety_class are the following:

— safety_category;

— user_defined_value.

4.2.57.1 safety_category

The safety_category specifies an indicator used to denote special consideration for the compartment with regard to a hazardous working environment for humans.

The value of safety_category is one of the following:

— A;

— B;

— C;

— user_defined.

NOTE See 4.2.57.1.1 - 4.2.57.1.4 for the definition of each allowable value for safety_category.

4.2.57.1.1 A

The compartment is designated as safety class A.

4.2.57.1.2 B

The compartment is designated as safety class B.

4.2.57.1.3 C

The compartment is designated as safety class C.

4.2.57.1.4 user_defined

The safety category of the compartment is defined by the user_defined_value attribute.

4.2.57.2 user_defined_value

The user_defined_value specifies the description if the safety_category has a value of user_defined. The user_defined_value need not be specified for a particular Compartment_safety_class.

4.2.58 Compartment_security_classification

A Compartment_security_classification is a type of Compartment_naval_administrative_property (see 4.2.52) that specifies the security requirements of a compartment with regards to personnel accessibility and security clearances.

The data associated with a Compartment_security_classification are the following:

— security_classification;

— user_defined_value.

4.2.58.1 security_classification

The security_classification specifies an indicator used to denote special considerations for the compartment with respect to accessibility and security clearances.

The value of security_classification is one of the following:

— classified;

— secret;

— unclassified;

— user_defined.

NOTE See 4.2.58.1.1 - 4.2.58.1.4 for the definition of each allowable value for security_classification.

4.2.58.1.1 classified

The compartment is designated for access only by persons with classified-level security clearance.

4.2.58.1.2 secret

The compartment is designated for access only by persons with secret-level security clearance.

4.2.58.1.3 unclassified

The compartment is designated for unclassified access.

4.2.58.1.4 user_defined

The security classification of the compartment is defined by the user_defined_value attribute.

4.2.58.2 user_defined_value

The user_defined_value specifies the description if the security_classification has a value of user_defined. The user_defined_value need not be specified for a particular Compartment_-security_classification.

4.2.59 Compartment_stiffened_surface_area_property

A Compartment_stiffened_surface_area_property is a type of Compartment_area_property (see 4.2.42) that is a measure of the amount of surface area for the compartment including the surface area of any interior stiffeners on the bounding bulkheads, decks, and hull structure.

NOTE The stiffened_surface_area is used to estimate amount of coating materials to be applied to the compartment surfaces and the attached stiffeners, such as primer or paint.

The data associated with a Compartment_stiffened_surface_area_property are the following:

— stiffened_surface_area.

4.2.59.1 stiffened_surface_area

The stiffened_surface_area specifies the value of the stiffened surface area including the surface area of the bulkhead and attached stiffeners.

4.2.60 Compartment_tightness

A Compartment_tightness is a type of General_compartment_property (see 4.2.104) that is an indicator as to the degree of tightness, the ability to prevent the passage of air and liquid, required of all bulkheads forming the boundary of the compartment.

The data associated with a Compartment_tightness are the following:

— required_bulkhead_tightness;

— user_defined_value.

4.2.60.1 required_bulkhead_tightness

The required_bulkhead_tightness specifies an indicator of the ability to prevent the passage of air and liquid for all bulkheads forming the boundary of the compartment.

The value of required_bulkhead_tightness is one of the following:

— air_tight;

— non_tight;

— oil_tight;

— user_defined;

— water_tight;

— weather_tight.

NOTE See 4.2.60.1.1 - 4.2.60.1.6 for the definition of each allowable value for required_bulkhead_tightness.

4.2.60.1.1 air_tight

An air tight boundary is one that will prevent the passage of air, gas, or fumes.

NOTE Testing could be similar to that used for water tight test with no passage of liquid, a chalk test on doors and other closures to check contact, or a pressure drop test.

4.2.60.1.2 non_tight

There is no requirement to prevent the passage of gases or liquids.

4.2.60.1.3 oil_tight

An oil tight boundary is one that can withstand the pressure of a liquid without deformation or damage.

4.2.60.1.4 user_defined

The tightness of the compartment is defined in the user_defined_value attribute.

4.2.60.1.5 water_tight

A water tight boundary is one that will not pass water.

4.2.60.1.6 weather_tight

A weather tight boundary is one that may allow light leakage when exposed to water.

NOTE This is in general used for doors and some hatch covers on the decks and bulkheads exposed to weather where water tightness is not required.

4.2.60.2 user_defined_value

The user_defined_value specifies the description for the tightness when a required_bulkhead_-tightness value of user_defined has been specified. The user_defined_value need not be specified for a particular Compartment_tightness.

4.2.61 Compartment_unstiffened_surface_area_property

A Compartment_unstiffened_surface_area_property is a type of Compartment_area_property (see 4.2.42) that is a measure of the amount of surface area for the compartment excluding the surface area of any interior stiffeners on the bounding bulkheads, decks, and hull structure.

NOTE The unstiffened_surface_area is used to estimate amount of coating materials to be applied to compartment surfaces but not the attached stiffeners, such as insulation.

The data associated with a Compartment_unstiffened_surface_area_property are the following:

— unstiffened_surface_area.

4.2.61.1 unstiffened_surface_area

The unstiffened_surface_area specifies the value of the unstiffened surface area.

4.2.62 Compartment_vertical_longitudinal_cross_sectional_area_property

A Compartment_vertical_longitudinal_cross_sectional_area_property is a type of Compartment_-area_property (see 4.2.42) that specifies a two dimensional cross-sectional area for a compartment.

NOTE Typically this area is used to reserve space early in the design process, such as the area needed for placement of a large piece of equipment.

The data associated with a Compartment_vertical_longitudinal_cross_sectional_area_property are the following:

— vertical_longitudinal_cross_sectional_area.

4.2.62.1 vertical_longitudinal_cross_sectional_area

The vertical_longitudinal_cross_sectional_area specifies an area measurement corresponding to a plane defined by the vertical and longitudinal global axes.

4.2.63 Compartment_vertical_transverse_cross_sectional_area_property

A Compartment_vertical_transverse_cross_sectional_area_property is a type of Compartment_area_-property (see 4.2.42) that specifies a two dimensional cross-sectional area for a compartment.

NOTE Typically this area is used to reserve space early in the design process, such as the area needed for placement of a large piece of equipment.

The data associated with a Compartment_vertical_transverse_cross_sectional_area_property are the following:

— vertical_transverse_cross_sectional_area.

4.2.63.1 vertical_transverse_cross_sectional_area

The vertical_transverse_cross_sectional_area specifies an area measurement corresponding to a plane defined by the vertical and transverse global axes.

4.2.64 Compartment_volume_permeability_property

A Compartment_volume_permeability_property is a type of General_compartment_property (see 4.2.104) that is a measure, expressed as a percentage of the volume of the compartment, representing open space that is not occupied by equipment, structure, or machinery that would flood in the event the watertight integrity of the compartment was damaged. This is a key parameter for the damage stability calculations for a ship.

The data associated with a Compartment_volume_permeability_property are the following:

— permeability.

4.2.64.1 permeability

The permeability specifies the percentage of the total volume of a compartment that is not occupied by the ship structure, systems, or permanently attached outfitting and furnishing objects.

4.2.65 Compartment_volume_property

A Compartment_volume_property is a type of General_compartment_property (see 4.2.104) that describes the volumetric properties of a compartment.

The data associated with a Compartment_volume_property are the following:

— centre_of_volume;

— volume.

4.2.65.1 centre_of_volume

The centre_of_volume specifies the centre of volume of a compartment in relation to the global coordinate system of the ship. See 4.3.43 for the application assertion.

4.2.65.2 volume

The volume specifies the volume of a compartment.

4.2.66 Compartment_ziplist_number

A Compartment_ziplist_number is a type of Compartment_naval_administrative_property (see 4.2.52) that specifies an organization-specific identifier used for departmental or divisional control over a compartment during an overhaul or repair availability.

The data associated with a Compartment_ziplist_number are the following:

— department_ziplist_number;

— division_ziplist_number.

4.2.66.1 department_ziplist_number

The department_ziplist_number specifies an organization-specific identifier used for departmental control over the compartment during an overhaul or repair availability.

4.2.66.2 division_ziplist_number

The division_ziplist_number specifies an organization-specific identifier used for departmental or division control over the compartment during an overhaul or repair availability.

4.2.67 Compensated_gross_tonnage

A Compensated_gross_tonnage represents a value for Gross_tonnage that reflects the complexity of the work involved in the construction of the ship.

The data associated with a Compensated_gross_tonnage are the following:

— compensation_factor;

— gross_tonnage_measurement;

— tonnage_value.

4.2.67.1 compensation_factor

The compensation_factor specifies the multiplication factor applied to the Gross_tonnage value in order to obtain the Compensated_gross_tonnage.

NOTE 1 The compensation factor is derived by the Association of West European Shipyards and the Shipbuilding Association of Japan, and varies according to the type of ship, the Deadweight for cargo ships, and Gross_tonnage for passenger ships.

NOTE 2 For any particular ship type, the compensation factor decreases with increasing ship size, i.e., the larger the ship, the smaller the man-hour requirement per Gross_tonnage.

4.2.67.2 gross_tonnage_measurement

The gross_tonnage_measurement specifies the Gross_tonnage measurement that is the basis for the compensation_factor. See 4.3.44 for the application assertion.

4.2.67.3 tonnage_value

The tonnage_value specifies the value of the Compensated_gross_tonnage resulting from the multiplication of the Gross_tonnage measurement by the compensation factor.

4.2.68 Corrosion_control_coating

A Corrosion_control_coating is a type of Coating (see 4.2.34) that is to be applied to a compartment to prevent corrosion of the steel due to contact with the enironmental elements or with a cargo.

The data associated with a Corrosion_control_coating are the following:

— applicability;

— primer;

— type_of;

— user_defined_type.

4.2.68.1 applicability

The applicability specifies the circumstances where the coating can be used.

The value of applicability is one of the following:

— B;

— C;

— RS;

— V.

NOTE See 4.2.68.1.1 - 4.2.68.1.4 for the definition of each allowable value for applicability.

4.2.68.1.1 B

The coating is suitable for ballast water.

4.2.68.1.2 C

The coating is suitable for crude oil.

4.2.68.1.3 RS

The coating is suitable for refined spirits.

4.2.68.1.4 V

The coating is suitable for void spaces.

4.2.68.2 primer

The primer specifies the Primer_coating that is required by the Corrosion_control_coating. See 4.3.45 for the application assertion.

4.2.68.3 type_of

The type_of specifies the chemical compound used to coat the hull structure.

The value of type_of is one of the following:

— aluminium;

— bituminous;

— chlorinated_rubber;

— coal_tar;

— epoxy;

— glassflake;

— isocynate;

— micaceous_iron_oxide;

— non_oxidising;

— phenolic;

— pitch;

— polyester;

— polyurethane;

— tar;

— user_defined;

— vinyl;

— water_based;

— zinc_rich;

— zinc_silicate.

NOTE See 4.2.68.3.1 - 4.2.68.3.19 for the definition of each allowable value for type_of.

4.2.68.3.1 aluminium

The coating is aluminium.

4.2.68.3.2 bituminous

The coating is bituminous.

4.2.68.3.3 chlorinated_rubber

The coating is chlorinated rubber.

4.2.68.3.4 coal_tar

The coating is coal tar.

4.2.68.3.5 epoxy

The coating is epoxy.

4.2.68.3.6 glassflake

The coating is glassflake.

4.2.68.3.7 isocynate

The coating is isocynate.

4.2.68.3.8 micaceous_iron_oxide

The coating is micaceous iron oxide.

4.2.68.3.9 non_oxidising

The coating is of a non-oxidising type.

4.2.68.3.10 phenolic

The coating is phenolic.

4.2.68.3.11 pitch

The coating is pitch.

4.2.68.3.12 polyester

The coating is polyester.

4.2.68.3.13 polyurethane

The coating is polyurethane.

4.2.68.3.14 tar

The coating is tar.

4.2.68.3.15 user_defined

The coating type is defined within the user_defined_type attribute.

4.2.68.3.16 vinyl

The coating is vinyl.

4.2.68.3.17 water_based

The coating is of a water-based type.

4.2.68.3.18 zinc_rich

The coating is zinc rich.

4.2.68.3.19 zinc_silicate

The coating is zinc silicate.

4.2.68.4 user_defined_type

The user_defined_type specifies a text description of the type of Corrosion_control_coating if the value of type_of is user_defined. The user_defined_type need not be specified for a particular Corrosion_control_coating.

4.2.69 Corrosion_protection

A Corrosion_protection describes properties for protecting compartment internals and boundaries from corrosion.

The data associated with a Corrosion_protection are the following:

— cathodic_protection;

— coating_height;

— coating_material.

4.2.69.1 cathodic_protection

The cathodic_protection specifies whether cathodic corrosion protection is applicable or not.

4.2.69.2 coating_height

The coating_height specifies the vertical range of the coating of the compartment. See 4.3.47 for the application assertion.

EXAMPLE A tank would be coated from 80 percent to 90 percent.

4.2.69.3 coating_material

The coating_material specifies the material that is to be used to coat the metal making up the compartment boundaries. See 4.3.46 for the application assertion.

4.2.70 Damage_case

A Damage_case is a representation of the state of the ship when it has sustained damage. The state is defined by the Loading_condition_definitions of the ship before damage occurred, those compartments that have been damaged, and the associated Stability_property objects.

The data associated with a Damage_case are the following:

— damage_cause;

— damaged_compartments;

— original_loads;

— position_of_damage;

— relative_damage_position;

— user_defined.

4.2.70.1 damage_cause

The damage_cause specifies the type of incident that has caused the damage to the associated compartments.

The value of damage_cause is one of the following:

— collision;

— explosion;

— grounding;

— user_defined.

NOTE See 4.2.70.1.1 - 4.2.70.1.4 for the definition of each allowable value for damage_cause.

4.2.70.1.1 collision

The damage is caused by an impact of the hull of the ship with some other object.

4.2.70.1.2 explosion

The damage is caused by a violent combustion of material resulting in a force impacting upon the hull of the ship, or internal compartments.

4.2.70.1.3 grounding

The damage is caused by an impact of the hull of the ship with the sea bed.

4.2.70.1.4 user_defined

The damage is caused by some means other than one of the pre-defined causes.

4.2.70.2 damaged_compartments

The damaged_compartments specifies the set of Compartment_design_definitions that are damaged. Each Compartment_design_definition provides the properties and related specification data such as permeability, volume, and capacity for the compartment. There may be more than one damaged_compartments for a Damage_case. See 4.3.48 for the application assertion.

4.2.70.3 original_loads

The original_loads specifies a definition describing the relationship between the compartments, the cargo and the original Floating_position of the ship for a given Deadweight. The condition can be either for the design or the operation of the ship. See 4.3.50 for the application assertion.

4.2.70.4 position_of_damage

The position_of_damage specifies a reference point within the ship of where the centre of the damage is believed to be. The data associated with this information is a qualifier to state whether this is an estimate or known fact. See 4.3.49 for the application assertion.

4.2.70.5 relative_damage_position

The relative_damage_position specifies a simple indication of whether the damage sustained is above the waterline, below the waterline, or on the waterline. This provides an early indication of whether the compartments affected are likely to become filled with water or not. If the damage is below the waterline, then it can be assumed that the compartment will become flooded to the maximum extent possible.

The value of relative_damage_position is one of the following:

— above_waterline;

— below_waterline;

— on_waterline.

NOTE See 4.2.70.5.1 - 4.2.70.5.3 for the definition of each allowable value for relative_damage_position.

4.2.70.5.1 above_waterline

The damage is centred above the current waterline.

4.2.70.5.2 below_waterline

The damage is centred below the current waterline.

4.2.70.5.3 on_waterline

The damage is centred on the current waterline.

4.2.70.6 user_defined

The user_defined specifies a text string to identify causes of damage not enumerated by the damage_type attribute. The user_defined need not be specified for a particular Damage_case.

4.2.71 Damage_position

A Damage_position is a reference point within the ship where the centre of the damage is believed to be, and a qualifier to state whether this is an estimate or known fact.

The data associated with a Damage_position are the following:

— centre_of_damage;

— position_accuracy.

4.2.71.1 centre_of_damage

The centre_of_damage specifies a reference for the centre of the damage. See 4.3.51 for the application assertion.

4.2.71.2 position_accuracy

The position_accuracy specifies a qualifier to state whether this is an estimate or an actual known fact.

The value of position_accuracy is one of the following:

— actual;

— estimate.

NOTE See 4.2.71.2.1 - 4.2.71.2.2 for the definition of each allowable value for position_accuracy.

4.2.71.2.1 actual

The centre_of_damage is at a known position.

4.2.71.2.2 estimate

The centre_of_damage is at an estimated position.

4.2.72 Damage_stability_definition

A Damage_stability_definition is a type of Stability_definition (see 4.2.156) that defines the stability properties for a given ship having been damaged. The results are defined in a tabular form for a given set of loading and damaged conditions, the associated Floating_position objects, and represents the righting arms and the centre of buoyancy for the heel angles attained through the damage inflicted on the ship.

The data associated with a Damage_stability_definition are the following:

— defined_for;

— extent_of_damage;

— representations.

4.2.72.1 defined_for

The defined_for specifies the ship for which the Damage_stability_definition is defined. See 4.3.53 for the application assertion.

4.2.72.2 extent_of_damage

The extent_of_damage specifies a representation of the ship in a damaged state. This information includes the Loading_condition_definitions of the ship before damage occurred, those compartments that have been damaged, and the associated stability properties. There may be more than one extent_of_damage for a Damage_stability_definition. See 4.3.52 for the application assertion.

4.2.72.3 representations

The representations specifies the Stability_table that represents the ship in the damaged state. There may be more than one representations for a Damage_stability_definition. See 4.3.54 for the application assertion.

4.2.73 Dangerous_goods_code

A Dangerous_goods_code identifies the nature of the danger associated with the specific cargo as specified by the International Maritime Dangerous Goods code and the International Convention for the Safety of Life at Sea.

The data associated with a Dangerous_goods_code are the following:

— class;

— subsidiary_risks.

4.2.73.1 class

The class specifies the primary hazard class of the cargo.

The value of class is one of the following:

— class_1;

— class_3;

— class_8;

— class_9;

— class_21;

— class_22;

— class_23;

— class_41;

— class_42;

— class_43;

— class_51;

— class_52;

— class_61;

— class_62;

— class_71;

— class_72;

— class_73.

NOTE See 4.2.73.1.1 - 4.2.73.1.4 for the definition of each allowable value for class.

4.2.73.1.1 class_1

The cargo is explosive.

4.2.73.1.2 class_3

The cargo is a flammable liquid.

4.2.73.1.3 class_8

The cargo is corrosive.

4.2.73.1.4 class_9

The cargo is a miscellaneous, dangerous substance. It is any other substance that experience has shown, or may show, to be of such a dangerous character that the provisions of IMO IC110E apply to it.

4.2.73.1.5 class_21

The cargo is a flammable gas.

4.2.73.1.6 class_22

The cargo is a non-flammable compressed gas.

4.2.73.1.7 class_23

The cargo is a poisonous gas.

4.2.73.1.8 class_41

The cargo is a flammable solid.

4.2.73.1.9 class_42

The cargo is a substance likely to spontaneously combust.

4.2.73.1.10 class_43

The cargo will emit flammable gas when in contact with water.

4.2.73.1.11 class_51

The cargo is an oxidizing agent.

4.2.73.1.12 class_52

The cargo is an organic peroxide.

4.2.73.1.13 class_61

The cargo is toxic.

4.2.73.1.14 class_62

The cargo is an infectious substance.

4.2.73.1.15 class_71

The cargo is a Category I radioactive substance.

4.2.73.1.16 class_72

The cargo is a Catagory II radioactive substance.

4.2.73.1.17 class_73

The cargo is a Category III radioactive substance.

4.2.73.2 subsidiary_risks

The subsidiary_risks specifies additional risks associated with the cargo.

The value of subsidiary_risks is one of the following:

— class_1;

— class_3;

— class_8;

— class_9;

— class_21;

— class_22;

— class_23;

— class_41;

— class_42;

— class_43;

— class_51;

— class_52;

— class_61;

— class_62;

— class_71;

— class_72;

— class_73.

NOTE See 4.2.73.2.1 - 4.2.73.2.4 for the definition of each allowable value for subsidiary_risks.

4.2.73.2.1 class_1

The cargo is explosive.

4.2.73.2.2 class_3

The cargo is a flammable liquid.

4.2.73.2.3 class_8

The cargo is corrosive.

4.2.73.2.4 class_9

The cargo is a miscellaneous, dangerous substance. It is any other substance that experience has shown, or may show, to be of such a dangerous character that the provisions of IMO IC110E apply to it.

4.2.73.2.5 class_21

The cargo is a flammable gas.

4.2.73.2.6 class_22

The cargo is a non-flammable compressed gas.

4.2.73.2.7 class_23

The cargo is a poisonous gas.

4.2.73.2.8 class_41

The cargo is a flammable solid.

4.2.73.2.9 class_42

The cargo is a substance likely to spontaneously combust.

4.2.73.2.10 class_43

The cargo will emit flammable gas when in contact with water.

4.2.73.2.11 class_51

The cargo is an oxidizing agent.

4.2.73.2.12 class_52

The cargo is an organic peroxide.

4.2.73.2.13 class_61

The cargo is toxic.

4.2.73.2.14 class_62

The cargo is an infectious substance.

4.2.73.2.15 class_71

The cargo is a Category I radioactive substance.

4.2.73.2.16 class_72

The cargo is a Catagory II radioactive substance.

4.2.73.2.17 class_73

The cargo is a Category III radioactive substance.

4.2.74 Deadweight

The Deadweight is the weight of the passengers, crew, cargo, stores, ballast, fresh water, fuel oil, and other comsumables being carried by a ship.

The data associated with a Deadweight are the following:

— deadweight_items;

— deadweight_value.

4.2.74.1 deadweight_items

The deadweight_items specifies the items on the ship that constitute the Deadweight measurement. There may be more than one deadweight_items for a Deadweight. See 4.3.55 for the application assertion.

4.2.74.2 deadweight_value

The deadweight_value specifies the value of the deadweight.

4.2.75 Deck_cargo_assignment

A Deck_cargo_assignment is a type of Cargo_assignment (see 4.2.13) that is an allocation of Unit_cargo to spaces on the deck of a ship for loading analysis during the design phase or to identify an actual cargo loading during the operating phase.

The data associated with a Deck_cargo_assignment are the following:

— cargo;

— deck_zone;

— position.

4.2.75.1 cargo

The cargo specifies the Unit_cargo_group that has been loaded on to the deck. See 4.3.58 for the application assertion.

4.2.75.2 deck_zone

The deck_zone specifies the area of deck to which the cargo is assigned. See 4.3.57 for the application assertion.

4.2.75.3 position

The position specifies the position on the deck where the cargo has been loaded. See 4.3.56 for the application assertion.

4.2.76 Deck_zone

A Deck_zone is a type of Space (see 4.2.145) that identifies a bounded region of a deck with design requirements or characteristics that must be managed in the design process.

4.2.77 Deck_zone_design_definition

A Deck_zone_design_definition is a type of Design_definition (see 4.2.82) that is the abstract definition of a version of a Deck_zone from a design perspective.

The data associated with a Deck_zone_design_definition are the following:

— constituent_compartments;

— deck_for_zone;

— defined_for;

— inner_boundaries;

— outer_boundary;

— properties;

— representations.

4.2.77.1 constituent_compartments

The constituent_compartments specifies the Compartments, or portions of Compartments, which consitute the Deck_zone. The constituent_compartments need not be specified for a particular Deck_zone_design_definition. There may be more than one constituent_compartments for a Deck_zone_design_definition. See 4.3.59 and 4.3.62 for the application assertions.

4.2.77.2 deck_for_zone

The deck_for_zone specifies External_instance_reference to the ISO 10303-216 Moulded_form object or ISO 10303-218 Structural_system object that is the deck on which the Deck_zone is defined. See 4.3.62 for the application assertion.

4.2.77.3 defined_for

The defined_for specifies the Deck_zones for which the Deck_zone_design_definition is applicable. There may be more than one defined_for for a Deck_zone_design_definition. See 4.3.61 for the application assertion.

4.2.77.4 inner_boundaries

The inner_boundaries specifies closed curves which specify any internal voids in the Deck_zone. There may be more than one inner_boundaries for a Deck_zone_design_definition. The inner_boundaries need not be specified for a particular Deck_zone_design_definition.

4.2.77.5 outer_boundary

The outer_boundary specifies a closed curve which specifies the outer boundary the Deck_zone. The outer_boundary need not be specified for a particular Deck_zone_design_definition.

4.2.77.6 properties

The properties specifies a collection of properties applicable to or derived from the design of a Zone. The properties need not be specified for a particular Deck_zone_design_definition. There may be more than one properties for a Deck_zone_design_definition. See 4.3.60 for the application assertion.

4.2.77.7 representations

The representations specifies the Compartment_shape_representations for the Deck_zone_design_-definition. The representations need not be specified for a particular Deck_zone_design_definition. There may be more than one representations for a Deck_zone_design_definition. See 4.3.63 for the application assertion.

4.2.78 Deck_zone_functional_definition

A Deck_zone_functional_definition is a type of Functional_definition (see 4.2.99) that defines the functional role of a Deck_zone. The role may be a pre-defined one or may be user-defined.

The data associated with a Deck_zone_functional_definition are the following:

— defined_for;

— used_for.

4.2.78.1 defined_for

The defined_for specifies the Deck_zones for which the Deck_zone_functional_definition is applicable. There may be more than one defined_for for a Deck_zone_functional_definition. See 4.3.64 for the application assertion.

4.2.78.2 used_for

The used_for specifies the name of a function that a specific Deck_zone may have in a ship.

The value of used_for is one of the following:

— boat_deck_zone;

— bridge_deck_zone;

— car_deck_zone;

— deck_compartment_arrangement;

— deck_loading_zone;

— main_deck_zone;

— sun_deck_zone;

— topside_zone;

— user_defined;

— weather_deck_zone.

NOTE See 4.2.78.2.1 - 4.2.78.2.10 for the definition of each allowable value for used_for.

4.2.78.2.1 boat_deck_zone

The deck zone is on the deck giving access to lifeboats.

4.2.78.2.2 bridge_deck_zone

The deck zone is on a deck on the same level as the wheelhouse.

4.2.78.2.3 car_deck_zone

The deck zone is an area on a deck intended for carriage of vehicles.

4.2.78.2.4 deck_compartment_arrangement

The deck zone is a zone that defines the arrangement of compartments on a single deck.

4.2.78.2.5 deck_loading_zone

The deck zone is a bounded area for the application of design loads for structural analysis or the assignment of cargo.

4.2.78.2.6 main_deck_zone

The deck zone is on an internal part of the main deck intended amongst others for carriage of cargo.

4.2.78.2.7 sun_deck_zone

The deck zone is an area on an open deck.

4.2.78.2.8 topside_zone

The deck zone is the uppermost exterior area of the ship.

4.2.78.2.9 user_defined

The Deck_zone function is other than one of the pre-defined functions and is defined in the user_def_function attribute.

4.2.78.2.10 weather_deck_zone

The deck zone is the uppermost deck, or portions of different decks, which are exposed to the elements.

4.2.79 Definable_object

A Definable_object is any type of product for which basic properties can be described. Each Definable_object is either an Item (see 4.2.109), an Item_relationship (see 4.2.110), or an Item_structure (see 4.2.111).

The data associated with a Definable_object are the following:

— id.

4.2.79.1 id

The id specifies the globally unique identifier for the Definable_object. See 4.3.65 for the application assertion.

4.2.80 Definition

A Definition is a type of Versionable_object (see 4.2.179) that is a collection of properties for a particular Definable_object object. Each Definition is either a Cargo_bay_definition (see 4.2.14), a Change_definition (see 4.2.21), a Design_definition (see 4.2.82), a Design_requirement (see 4.2.83), a Functional_definition (see 4.2.99), a General_characteristics_definition (see 4.2.103), a Lightship_-definition (see 4.2.113), a Loading_condition_definition (see 4.2.117), a Local_co_ordinate_system (see 4.2.121), a Spacing_table (see 4.2.155), or a Tonnage_definition (see 4.2.164).

NOTE Definitions support the following concepts in shipbuilding: design, function, manufacturing, general ship characteristics, design requirements, and parametric and library descriptions of objects.

The data associated with a Definition are the following:

— defined_for;

— id;

— local_units.

4.2.80.1 defined_for

The defined_for specifies the Definable_object objects to which the Definition applies. There may be more than one defined_for for a Definition. See 4.3.66 for the application assertion.

4.2.80.2 id

The id specifies the globally unambiguous identifier for the Definition. See 4.3.68 for the application assertion.

4.2.80.3 local_units

The local_units specifies the units that the Definition makes use of if they differ from the Units globally defined for the ship. Each Local_units may be one of the following: a Derived_unit (see 4.2.81), or a Named_unit (see 4.2.127). Local_units need not be specified for a particular Definition.

There may be more than one local_units for a Definition. See 4.3.67 and 4.3.69 for the application assertions.

4.2.81 Derived_unit

A Derived_unit is a unit of measure that is composed of elements that are pre-defined Named_unit objects with exponents.

4.2.82 Design_definition

A Design_definition is a type of Definition (see 4.2.80) that is the basis for all types of design definitions. The ability to reference representations differentiates a Design_definition from a Definition. Each Design_definition is either a Compartment_design_definition (see 4.2.45), a Deck_zone_design_definition (see 4.2.77), a Stability_definition (see 4.2.156), or a Zone_design_-definition (see 4.2.187).

The data associated with a Design_definition are the following:

— representations.

4.2.82.1 representations

The representations specifies the representations of the design definition. The representations need not be specified for a particular Design_definition. There may be more than one representations for a Design_definition.

4.2.83 Design_requirement

A Design_requirement is a type of Definition (see 4.2.80) that represents a constraint placed on a design. These constraints identify the set of rules to which the design must adhere. Each Design_requirement is either a Compartment_design_requirement (see 4.2.46), a Class_-compartment_requirement_definition (see 4.2.29), or a Class_deck_load_requirement_definition (see 4.2.30).

The data associated with a Design_requirement are the following:

— specification.

4.2.83.1 specification

The specification specifies a Document_reference that define a design requirement or rule. There may be more than one specification for a Design_requirement. See 4.3.70 for the application assertion.

4.2.84 Detailed_cargo_material_properties

A Detailed_cargo_material_properties is a type of General_cargo_material_properties (see 4.2.102) that specifies the detailed physical properties associated with the cargo.

The data associated with a Detailed_cargo_material_properties are the following:

— expansion_coefficient;

— specific_heat_capacity;

— thermal_conductivity;

— viscosity.

4.2.84.1 expansion_coefficient

The expansion_coefficient specifies the coefficient of thermal expansion. It is used to define the relationship between expansion and temperature change of the cargo material. The expansion_coefficient need not be specified for a particular Detailed_cargo_material_properties.

4.2.84.2 specific_heat_capacity

The specific_heat_capacity specifies the amount of energy required to raise the temperature of a gram of cargo material by one degree Centigrade. The specific_heat_capacity need not be specified for a particular Detailed_cargo_material_properties.

4.2.84.3 thermal_conductivity

The thermal_conductivity specifies the rate at which the cargo will conduct heat. The thermal_conductivity need not be specified for a particular Detailed_cargo_material_properties.

4.2.84.4 viscosity

The viscosity specifies the kinematic viscosity of the Liquid_cargo. When multiplied by the Reynolds number, gives the fluid velocity over a linear dimension. The viscosity need not be specified for a particular Detailed_cargo_material_properties.

4.2.85 Document

A Document is a type of Versionable_object (see 4.2.179) that represents an unambiguous identification of some human readable data item defined outside of ISO 10303. A document has an author and may be versioned.

The data associated with a Document are the following:

— author;

— description;

— source_type;

— title.

4.2.85.1 author

The author specifies the person, organization, or person and organization, that authored the Document.

4.2.85.2 description

The description specifies the textual description of the content of the Document. The description need not be specified for a particular Document.

4.2.85.3 source_type

The source_type specifies the format of the document.

EXAMPLE The Document may be in a printed copy of a book or in a file format.

4.2.85.4 title

The title specifies a description of the subject matter within the Document.

4.2.86 Document_portion

A Document_portion is the qualification of a specific subset or portion of a Document in generic terms.

EXAMPLE 1 A subsection of a book could be a subsection identified by its section number. Document_portion.element_type = 'subsection' and Document_portion.element_value = '3.3.2'.

EXAMPLE 2 A subsection of a book could be a subsection identified by its section title. Document_portion.element_type = 'subsection' and Document_portion.element_value = 'Introduction'.

EXAMPLE 3 A subsection of a report could be a range of pages identified by their numbers. Document_portion.element_type 'page' and Document_portion.element_value '1 -10, 15'.

The data associated with a Document_portion are the following:

— element_type;

— element_value;

— source.

4.2.86.1 element_type

The element_type specifies the name for this subset of the Document.

EXAMPLE Types of elements are page numbers, section numbers, or section title.

4.2.86.2 element_value

The element_value specifies the value for this subset of the Document.

EXAMPLE Possible element_values are: page numbers = "1-10, 15", section numbers = "3.2.4", section title = "Introduction".

4.2.86.3 source

The source specifies the Document to which the specified subset is related. See 4.3.71 for the application assertion.

4.2.87 Document_reference

A Document_reference is the qualification of a Document or sections of a Document in terms of its source or location. Each Document_reference may be a Document_reference_with_address (see 4.2.88).

EXAMPLE If the Document_reference source is a book, the pointer could be a section label or a page number.

The data associated with a Document_reference are the following:

— assigned_document.

4.2.87.1 assigned_document

The assigned_document specifies the Document or Document_portion that is to be associated with the product data. See 4.3.72 and 4.3.73 for the application assertion.

4.2.88 Document_reference_with_address

A Document_reference_with_address is a type of Document_reference (see 4.2.87) and External_reference (see 4.2.93) that specifies a pointer to a location inside the source.

4.2.89 Dry_cargo

A Dry_cargo is a type of Cargo (see 4.2.12) that is not in liquid or gaseous form. Each Dry_cargo may be one of the following: a Bulk_cargo (see 4.2.8), or a Unit_cargo (see 4.2.168).

The data associated with a Dry_cargo are the following:

— permeability;

— stowage_factor.

4.2.89.1 permeability

The permeability specifies the amount by which the Cargo takes up water. The permeability need not be specified for a particular Dry_cargo.

4.2.89.2 stowage_factor

The stowage_factor specifies the average specific volume for a dry cargo. The stowage_factor need not be specified for a particular Dry_cargo.

NOTE The stowage_factor is usually expressed as the volume in cubic meters that is occupied by one metric ton of the cargo.

4.2.90 Envisaged_version_creation

An Envisaged_version_creation is a type of Versionable_object_change_event (see 4.2.180) that is the event leading to a new Versionable_object. The event is an envisaged event and has not yet happened. The Definition, Item_structure or Item_relationship as the subject of the Event does not yet exist and is described in terms of descriptive, non-formal properties.

The data associated with an Envisaged_version_creation are the following:

— base;

— category.

4.2.90.1 base

The base specifies the Versionable_object objects that the envisaged new version is derived from. The base need not be specified for a particular Envisaged_version_creation. There may be more than one base for an Envisaged_version_creation. See 4.3.74 for the application assertion.

4.2.90.2 category

The category specifies the description of the envisaged Versionable_object.

4.2.91 Event

An Event is identification that something has happened at a certain time, activated by a certain Person_and_organization, for a certain reason. Each Event is either an Approval_event (see 4.2.4), a Check (see 4.2.26), or a Versionable_object_change_event (see 4.2.180).

The data associated with an Event are the following:

— caused_by;

— caused_when;

— description.

4.2.91.1 caused_by

The caused_by specifies the person and organization creating the Event

4.2.91.2 caused_when

The caused_when specifies the date and time that the Event occurred.

4.2.91.3 description

The description specifies a description for the reason of the Event.

4.2.92 External_instance_reference

An External_instance_reference is an instance of an entity that does not exist in the same scope.

NOTE The entity that is referenced must be a type of either Definable_object or Definition in order to be referable via a globally unambiguous identifier.

The data associated with an External_instance_reference are the following:

— entity_type;

— schema_name;

— target_GUID.

4.2.92.1 entity_type

The entity_type specifies the name of the type of the externally referenced instance.

4.2.92.2 schema_name

The schema_name specifies the schema in which the externally referenced instance is defined.

4.2.92.3 target_GUID

The target_GUID specifies the globally unambiguous identifier of the externally referenced instance. See 4.3.75 for the application assertion.

4.2.93 External_reference

An External_reference is the abstract denotion of a data source external to the data set where an instance of this entity exists. Each External_reference may be a Document_reference_with_address (see 4.2.88).

EXAMPLE A Universal_resource_locator denotes such a data source.

The data associated with an External_reference are the following:

— description;

— location.

4.2.93.1 description

The description specifies some additional information regarding the External_reference.

4.2.93.2 location

The location specifies the location of an external reference. In the case of a Universal_resource_-locator, the location is computer accessible by a specified transmission protocol. See 4.3.76 and 4.3.77 for the application assertions.

4.2.94 External_storage

An External_storage is the location of physical documents or items external to the current exchange. This entity may be relevant when identifying the location of items such as CD-ROMs, floppy disks, video tapes, or books.

EXAMPLE A public libary or a company archive may serve as External_storage.

The data associated with an External_storage are the following:

— location.

4.2.94.1 location

The location specifies the identification of an external storage place, typically without possibility for direct computer network access.

4.2.95 Fire_safe_coating

A Fire_safe_coating is a type of Coating (see 4.2.34) that is used on the structure to retard the spread of fire.

The data associated with a Fire_safe_coating are the following:

— low_flame_spread;

— nitro_cellulose_based;

— primer.

4.2.95.1 low_flame_spread

The low_flame_spread specifies whether or not the Coating has low flame spread characteristics, as specified by applicable standards.

NOTE A value of true indicates that the Coating has low flame spread characteristics.

4.2.95.2 nitro_cellulose_based

The nitro_cellulose_based specifies whether or not the Coating has a nitro-cellulose or other highly inflammable base.

NOTE A value of true indicates that the Coating has a nitro-cellulose or other highly inflammable base.

4.2.95.3 primer

The primer specifies the Primer_coating that is required by the Fire_safe_coating. See 4.3.78 for the application assertion.

4.2.96 Floating_position

A Floating_position is the draught and attitude of the ship when immersed and the resulting displacement volume.

The data associated with a Floating_position are the following:

— angle_of_heel;

— angle_of_trim;

— breadth_of_waterline;

— draught_at_amidships;

— length_of_waterline;

— moulded_form_displacement.

4.2.96.1 angle_of_heel

The angle_of_heel specifies the angle of rotation around the X-axis of the ship measured in radians and measured on a line parallel to the global Y-axis and the waterplane. The angle_of_heel is equal to zero when the centreplane is perpendicular to the waterplane. The angle_of_heel has positive values if the starboard side of the ship moves down.

4.2.96.2 angle_of_trim

The angle_of_trim specifies the angle of rotation around the Y-axis of the ship measured in radians and measured on a line parallel to the global X-axis and the waterplane. The angle_of_trim is equal to zero when the transverse cross-section is perpendicular to the waterplane. The angle_of_trim has positive values if the bow of the ship moves up.

4.2.96.3 breadth_of_waterline

The breadth_of_waterline specifies the breadth of the current waterline.

4.2.96.4 draught_at_amidships

The draught_at_amidships specifies the distance from the operating waterplane to the moulded bottom of the ship measured perpendicular at the centreline on the transverse cross-section amidships.

4.2.96.5 length_of_waterline

The length_of_waterline specifies the length of the current waterline.

4.2.96.6 moulded_form_displacement

The moulded_form_displacement specifies the wetted displacement of the ship.

4.2.97 Frame_table

A Frame_table is a type of Longitudinal_table (see 4.2.124) that has positions that reference the location of frames that are located along the global X-axis.

NOTE Frames are used for the internal structure of the ship and they are structural elements. A ship can have more than 100 frames. The intersection curve between a frame and the hull moulded form is a curve of transversal section through the hull of the ship.

4.2.98 Freeboard_characteristics

A Freeboard_characteristics is a type of General_characteristics_definition (see 4.2.103) that gives details of the assignment of freeboard for a ship. Freeboard is the distance measured from the waterline to the upper edge of the deck plating at the side of the freeboard deck amidships. The freeboard deck is the uppermost continuous deck exposed to the weather and the sea which has permanent means for the watertight closure of all exposed openings on the deck and in the side shell below.

NOTE The minimum freeboard is required principally to ensure that the ship is seaworthy when loaded and provides the ship with sufficient bouancy reserves to rise as it passes through waves and thus to remain largely dry on its decks.

The data associated with a Freeboard_characteristics are the following:

— applicable_loadline;

— assigned_code;

— date_freeboard_assigned;

— freeboard;

— freeboard_assigned_by.

4.2.98.1 applicable_loadline

The applicable_loadline specifies the resulting loadline for the assigned freeboard. See 4.3.79 for the application assertion.

4.2.98.2 assigned_code

The assigned_code specifies the type of loadline according to ship type, as specified in IMO IA701E.

The value of assigned_code is one of the following:

— A;

— A_PLUS;

— B;

— B_PLUS;

— B_60;

— B_100;

— other.

NOTE See 4.2.98.2.1 - 4.2.98.2.7 for the definition of each allowable value for assigned_code.

4.2.98.2.1 A

The ship is a type A ship, which is designed to carry only liquid cargoes in bulk, and in which the cargo tanks have only small access openings closed by watertight gasketed covers of steel or equivalent material.

4.2.98.2.2 A_PLUS

The ship is a type A ship over 150 metres in length.

4.2.98.2.3 B

The ship is a type B ship. Type B ships are those which do not fall into the category of type A ships.

4.2.98.2.4 B_PLUS

The ship is a type B ship with hatchways fitted with portable beams and covers on exposed freeboard or raised quarterdecks, and within 25 percent of the ship superstructure decks. These have the basic freeboard increased.

4.2.98.2.5 B_60

The ship is a type B ship which complies with less stringent subdivision requirements than the B_100 type. It may be assigned a basic freeboard reduced by up to 60 percent of the difference between B and A basic values.

4.2.98.2.6 B_100

The ship is a type B ship which is effectively adopting a type A freeboard by virtue of having steel weathertight covers fitted with gaskets and clamping devices, improved measures for the protection of the crew, better freeing arrangements, and satisfactory subdivision characteristics. It may be assigned a basic freeboard reduced by up to 100 percent of the difference between B and A basic values.

4.2.98.2.7 other

The ship is an unspecified type of vessel for the freeboard calculation.

4.2.98.3 date_freeboard_assigned

The date_freeboard_assigned specifies the date and time the freeboard is assigned.

4.2.98.4 freeboard

The freeboard specifies the assigned freeboard, which is the difference between the actual depth at side including the thickness of the stringer plate and wooden deck sheathing, if fitted, and the summer load draught.

4.2.98.5 freeboard_assigned_by

The freeboard_assigned_by specifies the organization which made the freeboard assignment.

4.2.99 Functional_definition

A Functional_definition is a type of Definition (see 4.2.80) that provides the capability to specify a role or purpose for an object. Each Functional_definition is either a Compartment_functional_-definition (see 4.2.47), a Deck_zone_functional_definition (see 4.2.78), a Shiptype (see 4.2.143), or a Zone_functional_definition (see 4.2.188).

The data associated with a Functional_definition are the following:

— local_units;

— user_def_function.

4.2.99.1 local_units

The local_units specifies the units that the Definition makes use of if they differ from the units globally defined for the ship.

NOTE For Functional_definition, the local_units attribute inherited from Definition (see 4.2.80.3) has been redeclared in the ARM to be a set of zero, which is interpreted to mean that the attribute shall not be populated in the AIM.

4.2.99.2 user_def_function

The user_def_function specifies a user-defined role or purpose of the Functional_definition. The user_def_function need not be specified for a particular Functional_definition.

4.2.100 Gaseous_cargo

A Gaseous_cargo is a type of Cargo (see 4.2.12) that has a natural condition of a non-solid, non-liquid gaseous state.

The data associated with a Gaseous_cargo are the following:

— cargo_type;

— carried_in_liquid_state;

— required_carriage_pressure.

4.2.100.1 cargo_type

The cargo_type specifies the type of Gaseous_cargo that can be loaded into the ship.

The value of cargo_type is one of the following:

— acetaldehyde;

— anhydrous_ammonia;

— avcat;

— butane;

— butadiene;

— butylene;

— chlorine;

— diethyl_ether;

— dimethylamine;

— ethylene;

— ethyl_chlorine;

— ethylene_oxide;

— inert_gas;

— isoprene;

— isopropylamine;

— liquified_natural_gas;

— liquified_petroleum_gas;

— methane;

— methyl_chloride;

— monoethylamine;

— naptha;

— propane;

— propane_butane_mix;

— propylene_oxide;

— propylene;

— user_defined;

— vinyl_ethyl_ether;

— vinyl_chloride_monomer.

NOTE See 4.2.100.1.1 - 4.2.100.1.28 for the definition of each allowable value for cargo_type.

4.2.100.1.1 acetaldehyde

The cargo is acetaldehyde.

4.2.100.1.2 anhydrous_ammonia

The cargo is anhydrous ammonia.

4.2.100.1.3 avcat

The cargo is avcat.

4.2.100.1.4 butadiene

The cargo is butadiene.

4.2.100.1.5 butane

The cargo is butane.

4.2.100.1.6 butylene

The cargo is butylene.

4.2.100.1.7 chlorine

The cargo is chlorine.

4.2.100.1.8 diethyl_ether

The cargo is diethyl ether.

4.2.100.1.9 dimethylamine

The cargo is dimethylamine.

4.2.100.1.10 ethyl_chlorine

The cargo is ethyl chlorine.

4.2.100.1.11 ethylene

The cargo is ethylene.

4.2.100.1.12 ethylene_oxide

The cargo is ethylene oxide.

4.2.100.1.13 inert_gas

The cargo is an inert gas.

4.2.100.1.14 isoprene

The cargo is isoprene.

4.2.100.1.15 isopropylamine

The cargo is isopropylamine.

4.2.100.1.16 liquified_natural_gas

The cargo is liquified natural gas.

4.2.100.1.17 liquified_petroleum_gas

The cargo is liquified petroleum gas.

4.2.100.1.18 methane

The cargo is methane.

4.2.100.1.19 methyl_chloride

The cargo is methyl chloride.

4.2.100.1.20 monoethylamine

The cargo is monoethylamine.

4.2.100.1.21 naptha

The cargo is naptha.

4.2.100.1.22 propane

The cargo is propane.

4.2.100.1.23 propane_butane_mix

The cargo is a mix of propane and butane.

4.2.100.1.24 propylene

The cargo is propylene.

4.2.100.1.25 propylene_oxide

The cargo is propylene oxide.

4.2.100.1.26 user_defined

The cargo is defined by the user.

4.2.100.1.27 vinyl_chloride_monomer

The cargo is vinyl chloride monomer.

4.2.100.1.28 vinyl_ethyl_ether

The cargo is vinyl ethyl ether.

4.2.100.2 carried_in_liquid_state

The carried_in_liquid_state specifies whether or not the gas is transported as a liquid.

NOTE A value of true indicates that the gas is transported as a liquid.

4.2.100.3 required_carriage_pressure

The required_carriage_pressure specifies the required pressure of the compartment in which the cargo is to be carried. The required_carriage_pressure need not be specified for a particular Gaseous_cargo.

NOTE The pressure is normally used to keep the gas in a liquid state. However, it may also be used in association with an inerting gas.

4.2.101 Gaseous_cargo_assignment

A Gaseous_cargo_assignment is a type of Compartment_cargo_assignment (see 4.2.43) that is an allocation of Gaseous_cargo to a tank for loading analysis during the design phase or to identify an actual cargo loading during the operating phase.

NOTE This assignment is used for inert gases. Most gaseous cargo is transported in liquid form.

4.2.102 General_cargo_material_properties

A General_cargo_material_properties specifies the general physical properties associated with the cargo. Each General_cargo_material_properties may be a Detailed_cargo_material_properties (see 4.2.84).

The data associated with a General_cargo_material_properties are the following:

— density;

— description;

— material_reference.

4.2.102.1 density

The density specifies the density of the cargo. The density need not be specified for a particular General_cargo_material_properties.

4.2.102.2 description

The description specifies a textual statement of the type of material. The description need not be specified for a particular General_cargo_material_properties.

4.2.102.3 material_reference

The material_reference specifies any reference to a source where more specific information about a material is available. The material_reference need not be specified for a particular General_-cargo_material_properties. See 4.3.80 for the application assertion.

4.2.103 General_characteristics_definition

A General_characteristics_definition is a type of Definition (see 4.2.80) that provides the primary dimensions and capacities of the ship. Each General_characteristics_definition is either a Class_and_statutory_designation (see 4.2.27), a Class_parameters (see 4.2.32), a Freeboard_-characteristics (see 4.2.98), a Global_axis_placement (see 4.2.105), an Owner_designation (see 4.2.131), a Principal_characteristics (see 4.2.135), a Ship_designation (see 4.2.142), or a Shipyard_-designation (see 4.2.144).

The data associated with a General_characteristics_definition are the following:

— defined_for.

4.2.103.1 defined_for

The defined_for specifies the Ship object for which the General_characteristics_definition applies. There may be more than one defined_for for a Ship. See 4.3.81 for the application assertion.

4.2.104 General_compartment_property

A General_compartment_property is a type of Compartment_property (see 4.2.56) that defines generic properties that are applicable to all types of compartments. Each General_compartment_property is either a Compartment_volume_property (see 4.2.65), a Compartment_area_property (see 4.2.42), a Compartment_volume_permeability_property (see 4.2.64), a Compartment_coating (see 4.2.44), a Compartment_tightness (see 4.2.60), a Compartment_occupancy (see 4.2.55), a Compartment_air_circulation_rate (see 4.2.41), a Compartment_illumination (see 4.2.50), a Compartment_abbreviated_name (see 4.2.38), a Compartment_acceleration (see 4.2.39), a Compartment_noise_category (see 4.2.53), a Compartment_insulation (see 4.2.51), or a Compartment_access_authorization (see 4.2.40).

4.2.105 Global_axis_placement

A Global_axis_placement is a type of General_characteristics_definition (see 4.2.103) that defines a fixed system of right handed orthogonal axes to which geometric data are referred. A Global_axis_placement shall have a positive Z-axis in an upward direction starting from the base of the ship and a positive X-axis running along the ship on the intersection of the centreline with the base. In one case it is directed from the after part of the ship to the forward part of the ship; in the other it is directed from the forward part of the ship to the aft part of the ship. The origin of the Global_axis_placement can be any point on the X-axis. The distance of the after perpendicular from the origin and the orientation of the X-axis shall be specified. If any other system of axes is used, local or global, then the transformation relations between it and the Global_axis_placement shall be specified.

NOTE Figure 4 illustrates different Global_axis_placement objects.

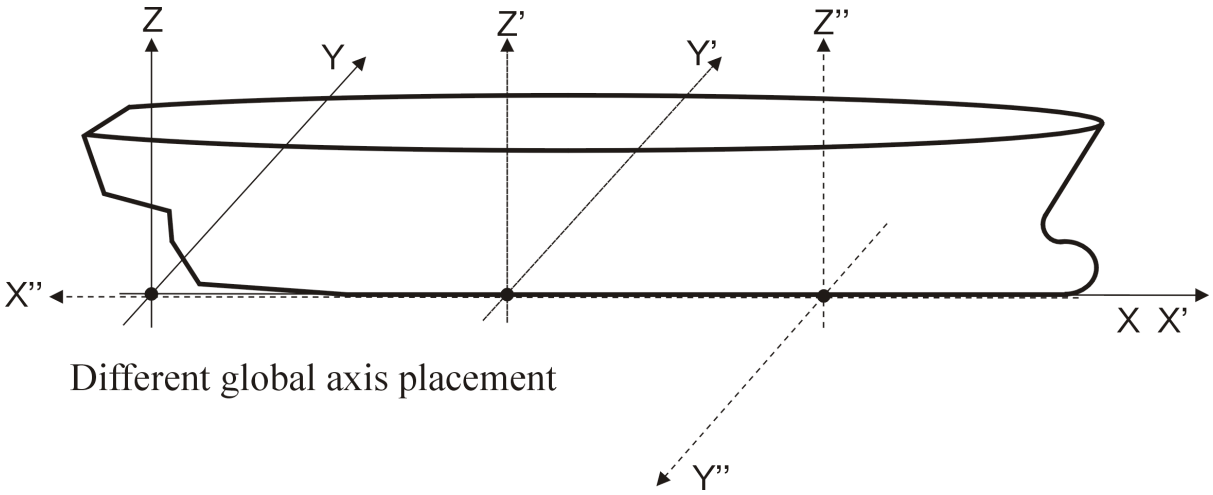

Figure 4 — Global axis placement

The data associated with a Global_axis_placement are the following:

— after_perpendicular_offset;

— orientation.

4.2.105.1 after_perpendicular_offset

The after_perpendicular_offset specifies the distance from the origin of the Global_axis_placement to the after perpendicular.

4.2.105.2 orientation

The orientation specifies the direction of the X-axis.

The value of orientation is one of the following:

— aft_pointing;

— forward_pointing.

NOTE See 4.2.105.2.1 - 4.2.105.2.2 for the definition of each allowable value for orientation.

4.2.105.2.1 aft_pointing

The orientation of the global ship coordinate system is a right handed system that has the positive X-axis from the forward part of the ship directed to the aft part of the ship.

4.2.105.2.2 forward_pointing

The orientation of the global ship coordinate system is a right handed ship system that has the positive X-axis from the aft part of the ship directed to the forward part of the ship.

4.2.106 Global_id

A Global_id is a persistent, global identifier that uniquely identifies the product data.

The data associated with a Global_id are the following:

— id.

4.2.106.1 id

The id specifies a unique, persistent identifier generated by the company that creates the product data.

4.2.107 Gross_tonnage

A Gross_tonnage is a type of Tonnage_measurement (see 4.2.165) that is the result of a calculation representing the total volume of a ship. It is the sum of the overdeck and underdeck tonnages.

The data associated with a Gross_tonnage are the following:

— overdeck_tonnage;

— underdeck_tonnage.

4.2.107.1 overdeck_tonnage

The overdeck_tonnage specifies the volume to the inside of the frames and deck plating of the tween decks, poop, bridge, forecastle, deckhouses, and erections above the tonnage deck less the exempted spaces. Spaces exempted include dry cargo space, unless in a break in the deck, and certain enclosed spaces associated with machinery, safety equipment, navigation, galleys, washrooms, water ballast, and workshops.

4.2.107.2 underdeck_tonnage

The underdeck_tonnage specifies the total volume of the ship below the tonnage deck to the inside of the frames, underside of the deck plating, and above the inner bottom.

4.2.108 Hull_applicability

A Hull_applicability is the identification of a ship hull, or a range of hulls within a class of ships, for which particular product data are applicable.

The data associated with a Hull_applicability are the following:

— definitions_for_hulls;

— end_hull;

— items_for_hulls;

— start_hull.

4.2.108.1 definitions_for_hulls

The definitions_for_hulls specifies the Definition objects (see 4.2.80) that are applicable for the range of hulls specified in start_hull and end_hull. The definitions_for_hulls need not be specified for a particular Hull_applicability. There may be more than one definitions_for_hulls for a Hull_applicability. See 4.3.82 for the application assertion.

4.2.108.2 end_hull

The end_hull specifies the final hull in a range of hulls for which the product data is applicable. The end_hull need not be specified for a particular Hull_applicability. If the end_hull is not specified, the product data is applicable to only the start_hull.

4.2.108.3 items_for_hulls

The items_for_hulls specifies the Item objects (see 4.2.109) that are applicable for the range of hulls specified in start_hull and end_hull. The items_for_hulls need not be specified for a particular Hull_applicability. There may be more than one items_for_hulls for a Hull_applicability. See 4.3.83 for the application assertion.

4.2.108.4 start_hull

The start_hull specifies the first hull in a range of hulls for which the product data is applicable.

4.2.109 Item

An Item is a type of Definable_object (see 4.2.79) that is a discrete, identifiable object used in one or more design, production, or operational activities. An Item is something to be created by a physical or mental activity or automatically derived from one or more other Items. An Item need not represent a physically realizable object; it may also represent an abstract concept. An Item may have relationships to other Items and may be a member in an Item_structure. Each Item is either a Change (see 4.2.20), a Ship (see 4.2.141), a Space (see 4.2.145), or a Space_product_structure (see 4.2.151).

NOTE An Item may represent some abstract concept such as an activity or task.

The data associated with an Item are the following:

— description;

— documentation;

— name;

— ship_context.

4.2.109.1 description

The description specifies the description for an Item. The description need not be specified for a particular Item.

4.2.109.2 documentation

The documentation specifies documentation available for an Item. The documentation need not be specified for a particular Item. There may be more than one documentation for an Item. See 4.3.84 for the application assertion.

4.2.109.3 name

The name specifies the human readable name of the concept that is represented by an Item.

4.2.109.4 ship_context

The ship_context specifies that an Item is applicable to a Ship. The ship_context need not be specified for a particular Item. See 4.3.85 for the application assertion.

4.2.110 Item_relationship

An Item_relationship is a type of Definable_object (see 4.2.79) and Versionable_object (see 4.2.179) that defines the association of two Item (see 4.2.109) objects. The related Item objects may share a common function or activity, or are dependent on each other. Each Item_relationship is a Space_arrangement_relationship (see 4.2.147).

NOTE The Item objects related by an Item_relationship may be either local instances or external instances; constraints ensure that either the local or the external instances exist.

The data associated with an Item_relationship are the following:

— external_item_1;

— external_item_2;

— item_1;

— item_2.

4.2.110.1 external_item_1

The external_item_1 specifies the relating Item of the relationship in the case where it is an externally referenced instance of an Item. The external_item_1 need not be specified for a particular Item_relationship. See 4.3.86 for the application assertion.

4.2.110.2 external_item_2

The external_item_2 specifies the related Item of the relationship in the case where it is an externally referenced instance of an Item. The external_item_2 need not be specified for a particular Item_relationship. See 4.3.86 for the application assertion.

4.2.110.3 item_1

The item_1 specifies the relating Item of the relationship in the case where it is in the same instance model as the Item. The item_1 need not be specified for a particular Item_relationship. See 4.3.87 for the application assertion.

4.2.110.4 item_2

The item_2 specifies the related Item of the relationship in the case where it is in the same instance model as the Item. The item_2 need not be specified for a particular Item_relationship. See 4.3.87 for the application assertion.

4.2.111 Item_structure

An Item_structure is a type of Definable_object (see 4.2.79) and Versionable_object (see 4.2.179) that is a collection of Item (see 4.2.109) objects possibly related by Item_relationship (see 4.2.110) objects. Each Item_structure is a Space_product_structure (see 4.2.151).

NOTE An Item_structure forms a graph without any restriction regarding the number of entries, the connectivity, nor the cyclicity.

The data associated with an Item_structure are the following:

— external_items;

— external_relationships;

— items;

— relationships.

4.2.111.1 external_items

The external_items specifies the Item objects outside of a single exchange file belonging to an Item_structure. The external_items need not be specified for a particular Item_structure. There may be more than one external_items for an Item_structure. See 4.3.88 for the application assertion.

4.2.111.2 external_relationships

The external_relationships specifies the relationships outside of a single exchange file between the Item objects belonging to an Item_structure. The external_relationships need not be specified for a particular Item_structure. There may be more than one external_relationships for an Item_structure. See 4.3.88 for the application assertion.

4.2.111.3 items

The items specifies the Item objects within a single exchange file belonging to an Item_structure. The items need not be specified for a particular Item_structure. There may be more than one items for an Item_structure. See 4.3.89 for the application assertion.

4.2.111.4 relationships

The relationships specifies the relationships within a single exchange file between the Item objects belonging to an Item_structure. The relationships need not be specified for a particular Item_structure. There may be more than one relationships for an Item_structure. See 4.3.90 for the application assertion.

4.2.112 Lane_position

A Lane_position is a type of Bay_position (see 4.2.7) that is the positon of a Unit_cargo using a definition of the lanes on a deck.

NOTE This will usually apply to ships used for vehicle stowage.

EXAMPLE A Ro-Ro ferry would use lane positions to locate vehicles.

The data associated with a Lane_position are the following:

— deck_number;

— frame_number;

— lane_number.

4.2.112.1 deck_number

The deck_number specifies the vertical position of the deck onto which the cargo has been loaded. See 4.3.93 for the application assertion.

4.2.112.2 frame_number

The frame_number specifies the longitudinal position of the cargo on the deck. See 4.3.91 for the application assertion.

4.2.112.3 lane_number

The lane_number specifies the transverse position of the Unit_cargo on the deck. See 4.3.92 for the application assertion.

4.2.113 Lightship_definition

A Lightship_definition is a type of Definition (see 4.2.80) that specifies the weight of the hull structure of the ship, including the weight of any installed machinery and outfitting, but excluding the weight of the crew, any passengers and cargoes.

The data associated with a Lightship_definition are the following:

— defined_for;

— lightship_centre_of_gravity;

— lightship_items;

— lightship_weight.

4.2.113.1 defined_for

The defined_for specifies the ship for which the Lightship_definition is applicable. There may be more than one defined_for for a Lightship_definition. See 4.3.96 for the application assertion.

4.2.113.2 lightship_centre_of_gravity

The lightship_centre_of_gravity specifies the centre of gravity of the ship in the global coordinate system. See 4.3.94 for the application assertion.

4.2.113.3 lightship_items

The lightship_items specifies the components that make up the lightship weight definition. The lightship_items need not be specified for a particular Lightship_definition. There may be more than one lightship_items for a Lightship_definition. See 4.3.95 for the application assertion.

4.2.113.4 lightship_weight

The lightship_weight specifies the light weight of the ship expressed in units of mass.

4.2.114 Lightship_weight_item

A Lightship_weight_item is a type of Weight_and_centre_of_gravity (see 4.2.184) that identifies the component that is a part of the total lightship weight. It may include the hull structure of the ship, machinery or outfitting, but does not include cargo, crew or passengers.

The data associated with a Lightship_weight_item are the following:

— aft_weight_extent;

— fwd_weight_extent;

— lightship_item_description.

4.2.114.1 aft_weight_extent

The aft_weight_extent specifies the length in the local coordinate system of the Lightship_weight_item identifying the aft extent.

4.2.114.2 fwd_weight_extent

The fwd_weight_extent specifies the length in the local coordinate system of the Lightship_weight_item identifying the forward extent.

4.2.114.3 lightship_item_description

The lightship_item_description specifies a descriptive label for the Lightship_weight_item.

4.2.115 Liquid_cargo

A Liquid_cargo is a type of Cargo (see 4.2.12) whose natural condition is a non-solid, non-gaseous liquid state.

The data associated with a Liquid_cargo are the following:

— cargo_type;

— required_carriage_pressure.

4.2.115.1 cargo_type

The cargo_type specifies the type of Liquid_cargo that can be loaded into the ship.

The value of cargo_type is one of the following:

— alcohol;

— ammonia;

— asphalt;

— aviation_oil;

— caustic_soda;

— cement;

— chemical;

— crude_oil;

— edible_oil;

— fuel_oil;

— fresh_water;

— fruit_juice;

— hydrocloric_acid;

— lubricating_oil;

— methanol;

— molasses;

— product_oil;

— salt_water;

— sullage;

— sludge;

— sulphur;

— user_defined;

— vegetable_oil;

— water_ballast;

— wine.

NOTE See 4.2.115.1.1 - 4.2.115.1.25 for the definition of each allowable value for cargo_type.

4.2.115.1.1 alcohol

The cargo is alcohol.

4.2.115.1.2 ammonia

The cargo is ammonia.

4.2.115.1.3 asphalt

The cargo is asphalt.

4.2.115.1.4 aviation_oil

The cargo is aviation oil.

4.2.115.1.5 caustic_soda

The cargo is caustic soda.

4.2.115.1.6 cement

The cargo is cement.

4.2.115.1.7 chemical

The cargo is a chemical substance.

4.2.115.1.8 crude_oil

The cargo is crude oil.

4.2.115.1.9 edible_oil

The cargo is edible oil.

4.2.115.1.10 fuel_oil

The cargo is fuel oil.

4.2.115.1.11 fresh_water

The cargo is fresh water.

4.2.115.1.12 fruit_juice

The cargo is fruit juice.

4.2.115.1.13 hydrocloric_acid

The cargo is hydrocloric acid.

4.2.115.1.14 lubricating_oil

The cargo is lubricating oil.

4.2.115.1.15 methanol

The cargo is methanol.

4.2.115.1.16 molasses

The cargo is molasses.

4.2.115.1.17 product_oil

The cargo is product oil.

4.2.115.1.18 salt_water

The cargo is salt water.

4.2.115.1.19 sullage

The cargo is oily water with other chemical impurities.

4.2.115.1.20 sludge

The cargo is sludge.

4.2.115.1.21 sulphur

The cargo is sulphur.

4.2.115.1.22 user_defined

The cargo is defined by the user.

4.2.115.1.23 vegetable_oil

The cargo is vegetable oil.

4.2.115.1.24 water_ballast

The cargo is water for ballast.

4.2.115.1.25 wine

The cargo is wine.

4.2.115.2 required_carriage_pressure

The required_carriage_pressure specifies the required pressure of the compartment in which the cargo is to be carried. The pressure is used to prevent vapourisation of the liquid. The required_carriage_pressure need not be specified for a particular Liquid_cargo.

4.2.116 Liquid_cargo_assignment

A Liquid_cargo_assignment is a type of Compartment_cargo_assignment (see 4.2.43) that is an allocation of Liquid_cargo or Gaseous_cargo to a tank for loading analysis during the design phase or to identify an actual cargo loading during the operating phase.

NOTE Gaseous cargoes are often carried in liquid form.

The data associated with a Liquid_cargo_assignment are the following:

— cargo_height.

4.2.116.1 cargo_height

The cargo_height specifies the actual height of the cargo loaded. This should be taken as the highest point of the cargo.

4.2.117 Loading_condition_definition

A Loading_condition_definition is a type of Definition (see 4.2.80) that defines a loading of the ship. The loading includes cargo loads that have been allocated and loaded into compartments or on decks, the associated Deadweight, and Floating_position. Each Loading_condition_definition may be one of the following: a Loading_condition_design_definition (see 4.2.118), or a Loading_condition_-operating_definition (see 4.2.119).

The data associated with a Loading_condition_definition are the following:

— cargo_loads;

— deadweight;

— defined_for;

— description;

— floating_position.

4.2.117.1 cargo_loads

The cargo_loads specifies the assignment of cargo that has been loaded onto the ship for a particular Loading_condition. There may be more than one cargo_loads for a Loading_condition_definition. See 4.3.97 for the application assertion.

4.2.117.2 deadweight

The deadweight specifies a measure of the mass for use during design and operation. The deadweight need not be specified for a particular Loading_condition_definition. See 4.3.98 for the application assertion.

NOTE The value is usually fixed for design but may vary during operation depending upon the loading condition.

4.2.117.3 defined_for

The defined_for specifies the Ship object for which the loading condition applies. There may be more than one defined_for for a Loading_condition_definition. See 4.3.100 for the application assertion.

4.2.117.4 description

The description specifies a free text description of the loading condition. The description need not be specified for a particular Loading_condition_definition.

4.2.117.5 floating_position

The floating_position specifies the attitude of the ship in the water in its present loading condition. See 4.3.99 for the application assertion.

4.2.118 Loading_condition_design_definition

A Loading_condition_design_definition is a type of Loading_condition_definition (see 4.2.117) that defines a hypothetical loading of the ship during design development for analysis of the expected behavior of the ship. The loading includes cargo loads that have been allocated and loaded onto a ship, the associated Deadweight, and Floating_position necessary for analysis.

The data associated with a Loading_condition_design_definition are the following:

— type_of.

4.2.118.1 type_of

The type_of specifies the qualification on the loading conditions used during design.

The value of type_of is one of the following:

— actual;

— expected;

— maximum;

— minimum;

— other.

NOTE See 4.2.118.1.1 - 4.2.118.1.5 for the definition of each allowable value for type_of.

4.2.118.1.1 actual

The loading condition is the normal design loading condition.

4.2.118.1.2 expected

The loading condition is the loading condition used to perform analyses.

4.2.118.1.3 maximum

The loading condition is the maximum design loading condition.

4.2.118.1.4 minimum

The loading condition is the minimum design loading condition.

4.2.118.1.5 other

The loading condition is any other design loading condition that is required.

4.2.119 Loading_condition_operating_definition

A Loading_condition_operating_definition is a type of Loading_condition_definition (see 4.2.117) that defines an actual loading of the ship while in service. The loading includes cargo loads that have been allocated and loaded onto a ship, the associated Deadweight, and the place and date of loading.

The data associated with a Loading_condition_operating_definition are the following:

— date_of_loading;

— place_of_loading;

— type_of.

4.2.119.1 date_of_loading

The date_of_loading specifies the date and time when the ship was loaded to its present condition. The date_of_loading need not be specified for a particular Loading_condition_operating_definition.

4.2.119.2 place_of_loading

The place_of_loading specifies a string value identifying the port at which the ship was last loaded. The place_of_loading need not be specified for a particular Loading_condition_operating_definition.

4.2.119.3 type_of

The type_of specifies the qualification on the loading conditions used during operation of the ship.

The value of type_of is one of the following:

— actual;

— arrival;

— departure;

— other.

NOTE See 4.2.119.3.1 - 4.2.119.3.4 for the definition of each allowable value for type_of.

4.2.119.3.1 actual

The loading condition is the loading condition when the ship is in normal service.

4.2.119.3.2 arrival

The loading condition is the loading condition when the ship docked at port.

4.2.119.3.3 departure

The loading condition is the loading condition when the ship embarked from port.

4.2.119.3.4 other

The loading condition is any other operating loading condition that is required.

4.2.120 Loadline

A Loadline is a mark placed on the side of the ship in the form of a ring of 300 mm outside diameter and 25 mm wide, intersected by a horizontal line 450 mm long and 25 mm wide. The upper edge of this line passes through the centre of the ring. The ring is positioned at midships and at a distance below the upper edge of the deck line which corresponds to the assigned minumum freeboard. This value may not be less than 50 mm. A series of load line marks are situated forward of this mark and these denote the minimum freeboards within certain geographical zones or in fresh water. The summer load line is level with the centre of the ring and is marked with the character S.

The data associated with a Loadline are the following:

— load_line_block_coefficient;

— load_line_depth;

— load_line_displacement;

— load_line_draught;

— load_line_length;

— load_line_regulation.

4.2.120.1 load_line_block_coefficient

The load_line_block_coefficient specifies the ratio measure obtained when the load_line_displacement is divided by the product of load_line_length, load_line_depth and the moulded breadth of the ship.

4.2.120.2 load_line_depth

The load_line_depth specifies the least moulded depth of the ship.

4.2.120.3 load_line_displacement

The load_line_displacement specifies the moulded displacement of the ship measured at the load_line_draught.

4.2.120.4 load_line_draught

The load_line_draught specifies the draught corresponding to 0.85 times the load_line_depth of the ship.

4.2.120.5 load_line_length

The load_line_length specifies the length of the ship measured in accordance with the specified load_line_regulation.

4.2.120.6 load_line_regulation

The load_line_regulation specifies the relevant IMO Load Line Convention containing the rules to which the load line has been calculated.

The value of load_line_regulation is one of the following:

— ILLC_1930;

— ILLC_1966;

— other.

NOTE See 4.2.120.6.1 - 4.2.120.6.3 for the definition of each allowable value for load_line_regulation.

4.2.120.6.1 ILLC_1930

The applicable regulation is the 1930 International loadline convention.

4.2.120.6.2 ILLC_1966

The applicable regulation is the 1966 International loadline convention, IMO IA701E.

4.2.120.6.3 other

The applicable regulation is another set of regulations for the calculation of the load lines.

4.2.121 Local_co_ordinate_system

A Local_co_ordinate_system is a type of Definition (see 4.2.80) that is used to locate an object in space. A Local_co_ordinate_system is always defined with respect to another coordinate system. This might be the Global_axis_placement or another Local_co_ordinate_system that is a member in

the same hierarchy. Each Local_co_ordinate_system may be Local_co_ordinate_system_with_-position_reference (see 4.2.122).

NOTE 1 Local axes and origin are handled in the same way as for axis2_placement_3d. Axis2_placement_3d is a STEP resource entity defined in ISO 10303-42.

NOTE 2 A Local_co_ordinate system will always form a right-handed system.

The data associated with a Local_co_ordinate_system are the following:

— parent.

4.2.121.1 parent

The parent specifies the underlying coordinate system that serves as definition space for the current coordinate system. See 4.3.101 and 4.3.102 for the application assertions.

4.2.122 Local_co_ordinate_system_with_position_reference

A Local_co_ordinate_system_with_position_reference is a type of Local_co_ordinate_system (see 4.2.121) that directly refers to the unique Global_axis_placement as its parent. Its location is defined by references to longitudinal, vertical, or transversal frames, possibly using an additional offset distance value. Alternatively absolute coordinates may be specified. Also, combinations of coordinates and references are valid. A Local_co_ordinate_system_with_position_reference shall not specify rotations as transformation to the global system; its axes are required to be parallel to the axes of the Global_axis_placement.

The data associated with a Local_co_ordinate_system_with_position_reference are the following:

— longitudinal_ref;

— transversal_ref;

— vertical_ref.

4.2.122.1 longitudinal_ref

The longitudinal_ref specifies a Longitudinal_position, possibly with an offset value or an absolute coordinate value, along the longitudinal axis of the global coordinate system. The longitudinal_ref need not be specified for a particular Local_co_ordinate_system_with_position_reference. See 4.3.103 for the application assertion.

4.2.122.2 transversal_ref

The transversal_ref specifies a Transversal_position, possibly with an offset value or an absolute coordinate value, along the transversal axis of the global coordinate system. The transversal_ref need not be specified for a particular Local_co_ordinate_system_with_position_reference. See 4.3.103 for the application assertion.

4.2.122.3 vertical_ref

The vertical_ref specifies a Vertical_position, possibly with an offset value or an absolute coordinate value, along the vertical axis of the global coordinate system. The vertical_ref need not be specified for a particular Local_co_ordinate_system_with_position_reference. See 4.3.103 for the application assertion.

4.2.123 Longitudinal_position

A Longitudinal_position is a type of Spacing_position (see 4.2.153) that is located on the global X-axis.

NOTE Each Longitudinal_position may also be combined with a Spacing_position_with_offset (see 4.2.154).

4.2.124 Longitudinal_table

A Longitudinal_table is a type of Spacing_table (see 4.2.155) that has positions that are along the longitudinal axis (the X-axis) of the global coordinate system. Each Longitudinal_table may be either a Frame_table (see 4.2.97), or a Station_table (see 4.2.160).

The data associated with a Longitudinal_table are the following:

— spacing_table_representations.

4.2.124.1 spacing_table_representations

The spacing_table_representations specifies the longitudinal positions that make up the Longitudinal_table. The spacing_table_representations need not be specified for a particular Longitudinal_table. There may be more than one spacing_table_representations for a Longitudinal_-table. See 4.3.104 for the application assertion.

4.2.125 Moment_3d

A Moment_3d is a collection of moment components at three major coordinate system axes: vertical, transveral, and longitudinal.

The data associated with a Moment_3d are the following:

— longitudinal_moment;

— origin;

— transverse_moment;

— vertical_moment.

4.2.125.1 longitudinal_moment

The longitudinal_moment specifies the moment component along the longitudinal axis.

4.2.125.2 origin

The origin specifies the location in the local coordinate system about which the moment component is defined. See 4.3.105 for the application assertion.

4.2.125.3 transverse_moment

The transverse_moment specifies the moment component along the transverse axis.

4.2.125.4 vertical_moment

The vertical_moment specifies the moment component along the vertical axis.

4.2.126 Moments_of_inertia

A Moments_of_inertia is the inertial resistance to motion of a fluid cargo in a tank.

The data associated with a Moments_of_inertia are the following:

— long_moment_of_inertia;

— trans_moment_of_inertia.

4.2.126.1 long_moment_of_inertia

The long_moment_of_inertia specifies the value of the second moment of the boundary formed by the intersection of the compartment and a plane representing the Liquid_cargo static waterline. The lever of the moment is parallel to the longitudinal axis of the ship.

4.2.126.2 trans_moment_of_inertia

The trans_moment_of_inertia specifies the value of the second moment of the boundary formed by the intersection of the compartment and a plane representing the Liquid_cargo static waterline. The lever of the moment is parallel to the transverse axis of the ship.

4.2.127 Named_unit

A Named_unit is a pre-defined unit specified in ISO 10303-41.

4.2.128 Navy_ship

A Navy_ship is a type of Shiptype (see 4.2.143) that is a ship operating under military command.

The data associated with a Navy_ship are the following:

— has_type.

4.2.128.1 has_type

The has_type specifies the type of the Navy_ship.

The value of has_type is one of the following:

— aircraft_carrier;

— auxiliary_oiler;

— corvette;

— cruiser;

— destroyer;

— fleet_auxiliary_vessel;

— frigate;

— landing_platform_dock;

— landing_platform_helicopter;

— mine_warfare_ship;

— patrol_force_vessel;

— service_craft;

— submarine;

— user_defined.

NOTE See 4.2.128.1.1 - 4.2.128.1.14 for the definition of each allowable value for has_type.

4.2.128.1.1 aircraft_carrier

The ship is designed to deploy aircraft or helicopters in sustained anti-submarine warfare operations and fighter protection, with full reconnaissance and strike capability.

EXAMPLE Deployed aircraft may include short take off and vertical landing, short take off but arrested recovery, or conventional take off.

4.2.128.1.2 auxiliary_oiler

The ship is designed to replenish ships at sea with liquids during world-wide operations, with vertical-replenishment services for the transfer of solids.

NOTE The ship has a stable platform suitable for helicopters, including stowage and maintenance facilities plus emergency landing of other helicopters.

4.2.128.1.3 corvette

The ship is designed as a small escort vessel to a task group. Corvettes are fitted primarily to fulfil an anti-submarine warfare role.

NOTE Corvettes are fitted primarily to fulfil an anti-submarine warfare role.

4.2.128.1.4 cruiser

The ship is designed to screen carrier task forces.

NOTE Cruisers, many with guided missiles or carrying a helicopter, provide anti-air warfare and anti-submarine capabilities. Cruisers also provide protection against anti-ship cruise missile threats at extended ranges, particularly in the presence of enemy electronic countermeasures.

4.2.128.1.5 destroyer

The ship is a major surface combatant ship that is used to conduct operations with strike, anti-submarine warfare and amphibious forces, and to perform screening and convoy duties.

NOTE A destroyer may also be equipped with helicopters, providing an enhanced capability.

4.2.128.1.6 fleet_auxiliary_vessel

The ship is designed to supply warships at sea with fuel, food, stores, and ammunition.

NOTE An auxiliary also provides aviation platforms, amphibious support for the navy and marines and sea transport for army units. There are many types of fleet auxiliary vessels.

4.2.128.1.7 frigate

The ship is a general purpose, ocean escort vessel. Operational requirements necessitate that frigates perform the duties of area defense ships, capable of defending a task group against air threats.

NOTE Frigates may also provide command facilities and accommodation. Secondary capabilities include anti-surface warfare, naval gunfire support and anti-submarine warfare.

4.2.128.1.8 landing_platform_dock

The ship is designed to transport a large embarked military force plus support equipment across open oceans; support a flexible landing on hostile shores using onboard helicopters and landing craft, and coordinate the naval, air, and land aspects of amphibious operations through command, control, and communications facilities.

4.2.128.1.9 landing_platform_helicopter

The ship is designed as an amphibious helicopter carrier which enables the rapid tactical deployment of airborne troops and equipment to spearhead amphibious operations ashore.

NOTE It can also stand off a coast at strategic range in a deterrent operational role. Peacetime roles include troop and equipment transport, and humanitarian tasks such as disaster relief.

4.2.128.1.10 mine_warfare_ship

The ship is a mine countermeasures ship that is specially constructed for the hunting, sweeping by mechanical, acoustic, or magnetic means, and clearance of mines in both inshore coastal and deep, exposed waters. Since the ship is designed to have minimal magnetic signature, the hulls are generally constructed of low-magnetic steel or laminated wood

4.2.128.1.11 patrol_force_vessel

The ship is a general purpose naval craft.

NOTE Larger offshore patrol vessels are used for firefighting, rescue or supply tasks, others are used as protection vessels, sometimes carrying a marine detachment and semi-rigid craft to act as a rapid response squadron. Some have ice-strengthened hulls and a helicopter landing deck for operation as survey vessels in the Arctic or Antarctic regions. The size, manoeuvrability, and other operational characteristics of smaller coastal patrol craft indicate that they are ideally suited for patrol, search, and rescue duties in coastal areas, or ports, harbours, and other restricted waters.

4.2.128.1.12 service_craft

The ship is a small ship designed to provide services to a fleet in harbors and ports.

NOTE These include tugs, tenders, barges, patrol craft, recovery vessels, floating docks, etc. Other service craft providing support are larger ocean-going vessels, such as transports, survey and research ships, repair vessels, cargo ships, and hospital ships.

4.2.128.1.13 submarine

The ship is a manned submersible vessel that is designed to perform surveillance, personnel delivery and recovery, and anti-submarine and anti-surface warfare.

NOTE To perform these functions, generally the weapon system of a submarine is capable of stowing, firing, and controlling heavyweight torpedoes, underwater-to-surface guided weapons, or submarine-laid mines.

Submarines are able to operate submerged in shallow waters, in open ocean, and are capable of operation in waters from the tropics to the arctic.

4.2.128.1.14 user_defined

The ship type is not one of the pre-defined types.

NOTE Details may be found in the description attribute for Shiptype.

4.2.129 Net_tonnage

A Net_tonnage is a type of Tonnage_measurement (see 4.2.165) that is a calculation of the cargo carrying space within the ship. It is the Gross_tonnage with deductions for crew spaces, engine room, water ballast, and any space not used for passengers or cargo.

4.2.130 Non_manifold_surface_shape

A Non_manifold_surface_shape is a shape representation that conforms to ISO 10303-508.

4.2.131 Owner_designation

An Owner_designation is a type of General_characteristics_definition (see 4.2.103) that specifies the organizations that order, own, and manage the ship.

The data associated with an Owner_designation are the following:

— local_units;

— managing_company;

— ordering_company;

— owner_approval;

— owning_company.

4.2.131.1 local_units

The local_units specifies the units that the Definition makes use of if they differ from the units globally defined for the ship.

NOTE For Owner_designation, the local_units attribute inherited from Definition (see 4.2.80.3) has been redeclared in the ARM to be a set of zero, which is interpreted to mean that the attribute shall not be populated in the AIM.

4.2.131.2 managing_company

The managing_company specifies the organization that is responsible for managing and operating the ship.

4.2.131.3 ordering_company

The ordering_company specifies the organization that ordered the ship at a shipyard.

4.2.131.4 owner_approval

The owner_approval specifies an indication that the ship owner has approved the design of the ship. The owner_approval need not be specified for a particular Owner_designation.

4.2.131.5 owning_company

The owning_company specifies the organization that legally owns the ship.

4.2.132 Person_group

A Person_group is a type of Unit_cargo_group (see 4.2.171) that is either passengers or members of the crew.

The data associated with a Person_group are the following:

— number_of_people;

— person_type.

4.2.132.1 number_of_people

The number_of_people specifies the number of people in the group.

4.2.132.2 person_type

The person_type specifies the role of the people in the group.

The value of person_type is one of the following:

— crew;

— enlisted;

— officers;

— passengers.

NOTE See 4.2.132.2.1 - 4.2.132.2.4 for the definition of each allowable value for person_type.

4.2.132.2.1 crew

The role of the Person_group is as members of the crew.

4.2.132.2.2 enlisted

The role of the Person_group is as enlisted personnel.

4.2.132.2.3 officers

The role of the Person_group is as officers.

4.2.132.2.4 passengers

The role of the Person_group is as passengers.

4.2.133 Precision

A Precision is the geometric precision of the CAD system from which the product data originated.

The data associated with a Precision are the following:

— minimum_point_spacing.

4.2.133.1 minimum_point_spacing

The minimum_point_spacing specifies the minimum distance between two points that are considered to be coincident in the originating CAD system.

4.2.134 Primer_coating

A Primer_coating is a type of Coating (see 4.2.34) that is used to coat steel after surface preparation and prior to fabrication, such that it has no significant deleterious effect on subsequent welding work.

4.2.135 Principal_characteristics

A Principal_characteristics is a type of General_characteristics_definition (see 4.2.103) that specifies the main shape parameters of the ship. Principal_characteristics also include data that is required in subsequent iterations of the hull development process when one is considering hydrostatics and damaged stability.

The data associated with a Principal_characteristics are the following:

— block_coefficient;

— design_deadweight;

— design_draught;

— length_between_perpendiculars;

— max_draught_at_AP;

— max_draught_at_FP;

— min_draught_at_AP;

— min_draught_at_FP;

— moulded_breadth;

— moulded_depth.

4.2.135.1 block_coefficient

The block_coefficient specifies the ratio of the moulded displacement volume to the volume of a block that has its length equal to the length_between_perpendiculars, its breadth equal to the maximum immersed moulded_breadth and its depth equal to the design_draught. The block_coefficient need not be specified for a particular Principal_characteristics.

NOTE The block_coefficient should be defined only for mono hull ships.

4.2.135.2 design_deadweight

The design_deadweight specifies the weight of the ship representing the weight of cargo, bunker fuel, water, passengers, crew and consumables that a ship can carry when loaded to the summer load line. The design_deadweight need not be specified for a particular Principal_characteristics.

4.2.135.3 design_draught

The design_draught specifies the draught to which the ship has been designed to operate. The design_draught need not be specified for a particular Principal_characteristics.

4.2.135.4 length_between_perpendiculars

The length_between_perpendiculars specifies the length measured from the after perpendicular to the forward perpendicular of the ship.

4.2.135.5 max_draught_at_AP

The max_draught_at_AP specifies the maximum possible draught at the aft perpendicular during the operation of the ship. The max_draught_at_AP need not be specified for a particular Principal_characteristics.

NOTE The max_draught_at_AP is used for hull cross section approval for ice class notation.

4.2.135.6 max_draught_at_FP

The max_draught_at_FP specifies the maximum possible draught at the forward perpendicular during the operation of the ship. The max_draught_at_FP need not be specified for a particular Principal_characteristics.

NOTE The max_draught_at_FP is used for hull cross section approval for ice class notation.

4.2.135.7 min_draught_at_AP

The min_draught_at_AP specifies the minimum possible draught at the aft perpendicular during the operation of the ship. The min_draught_at_AP need not be specified for a particular Principal_characteristics.

NOTE The min_draught_at_AP is used for hull cross section approval for ice class notation.

4.2.135.8 min_draught_at_FP

The min_draught_at_FP specifies the minimum possible draught at the forward perpendicular during the operation of the ship. The min_draught_at_FP need not be specified for a particular Principal_characteristics.

NOTE The min_draught_at_FP is used for hull cross section approval for ice class notation.

4.2.135.9 moulded_breadth

The moulded_breadth specifies the maximum breath of the ship amidships and at the design_draught.

4.2.135.10 moulded_depth

The moulded_depth specifies the vertical distance above the baseline to the uppermost deck where the deck joins the side of the ship measured amidships.

4.2.136 Regulation

A Regulation is a set of international and national regulations as well as standards that apply to the ship.

The data associated with a Regulation are the following:

— international_regulations;

— national_regulations;

— standards.

4.2.136.1 international_regulations

The international_regulations specifies all relevant international regulations that apply to the ship. The international_regulations need not be specified for a particular Regulation. There may be more than one international_regulations for a Regulation. See 4.3.106 for the application assertion.

4.2.136.2 national_regulations

The national_regulations specifies all relevant national regulations that apply to the ship. The national_regulations need not be specified for a particular Regulation. There may be more than one national_regulations for a Regulation. See 4.3.106 for the application assertion.

4.2.136.3 standards

The standards specifies all relevant standards that apply to the ship. The standards need not be specified for a particular Regulation. There may be more than one standards for a Regulation. See 4.3.106 for the application assertion.

4.2.137 Relative_cargo_position

A Relative_cargo_position is a type of Cargo_position (see 4.2.17) that defines a relative position for the cargo in terms of the longitudinal, transverse, and vertical axis in the global coordinate system of the ship.

The data associated with a Relative_cargo_position are the following:

— longitudinal_location;

— transverse_location;

— vertical_location.

4.2.137.1 longitudinal_location

The longitudinal_location specifies the longitudinal placement of the cargo. See 4.3.107 for the application assertion.

4.2.137.2 transverse_location

The transverse_location specifies the transversal placement of the cargo. See 4.3.108 for the application assertion.

4.2.137.3 vertical_location

The vertical_location specifies the vertical placement of the cargo. See 4.3.109 for the application assertion.

4.2.138 Research_ship

A Research_ship is a type of Shiptype (see 4.2.143) that specifies the research function performed by a ship.

The data associated with a Research_ship are the following:

— has_type.

4.2.138.1 has_type

The has_type specifies the type of the Research_ship.

The value of has_type is one of the following:

— user_defined.

4.2.138.1.1 user_defined

The ship type is not of a pre-defined type.

NOTE Details may be found in the description attribute for Shiptype.

4.2.139 Revision

A Revision is a type of Versionable_object (see 4.2.179) that serves as the link between the object of interest and the definitions of its constituents and associated members. Each Revision may be a Revision_with_context (see 4.2.140).

EXAMPLE The object of interest can be a compartment whose members are versions of structural, piping and equipment parts.

NOTE A Revision is comparable to a drawing revision; a snapshot of a version of the design at a point in time. A Revision is not created automatically, but must be created explicitly each time one is needed.

The data associated with a Revision are the following:

— description;

— members;

— name.

4.2.139.1 description

The description specifies the cause for the creation of a new Revision.

4.2.139.2 members

The members specifies the Versionable_object objects within the Revision. There may be more than one members for a Revision. See 4.3.110 for the application assertion.

4.2.139.3 name

The name specifies a label that identifies a particular Revision.

4.2.140 Revision_with_context

A Revision_with_context is a type of Revision (see 4.2.139) that serves as the link between the object of interest, the context, and the definitions of its constituents and associated members. Each Revision_with_context may be a Space_product_structure_revision (see 4.2.152).

The data associated with a Revision_with_context are the following:

— context_of_revision.

4.2.140.1 context_of_revision

The context_of_revision specifies the link to a Definable_object (see 4.2.79) to which the Revision applies. See 4.3.111 for the application assertion.

4.2.141 Ship

A Ship is a type of Item (see 4.2.109) that is a large seagoing vessel. Product data related to the ship is supported for many stages of the life cycle of a ship, including new project, early and detailed design, production engineering, manufacturing, operations, and scrapping. The ship may represent a single hull, or it may represent a range of hulls within a class of sister ships that share identical product data.

NOTE 1 The ship is the primary product supported by the ISO 10303 shipbuilding application protocols. All data defining attributes, properties, and the geometric representation of the ship are exchanged as definitions and properties of this product.

NOTE 2 The name of the ship is specified in Ship_designation, where it may be versioned.

The data associated with a Ship are the following:

— single_hull_or_class;

— ship_items;

— units.

4.2.141.1 single_hull_or_class

The single_hull_or_class specifies whether the exchange of data is applicable to a single hull or to multiple hulls in a class of ships.

The value of the single_hull_or_class shall be one of the following:

— design_for_multiple_hulls;

— design_for_single_hull.

NOTE See 4.2.141.1.2 - 4.2.141.1.1 for the definition of each allowable value for single_hull_or_class.

4.2.141.1.1 design_for_multiple_hulls

The product data is applicable to multiple hulls in a class of ships.

4.2.141.1.2 design_for_single_hull

The product data is applicable to a single hull.

4.2.141.2 ship_items

The ship_items specifies the constituent objects that are applicable to a particular ship. The ship_items need not be specified for a particular Ship. There may be more than one ship_items for a Ship. See 4.3.113 for the application assertion.

4.2.141.3 units

The units specifies a reference to a set of pre-defined units for all types of measures that may appear in the ship model. Each unit may be either a Derived_unit or a Named_unit. Units may have no value contained in it or it may have one or more values, without any duplications. The units need not be specified for a particular Ship. There may be more than one units for a Ship. See 4.3.112 and 4.3.114 for the application assertions.

4.2.142 Ship_designation

A Ship_designation is a type of General_characteristics_definition (see 4.2.103) that specifies the identification given to the ship in order that it can be categorized by any shipping related organization.

The data associated with a Ship_designation are the following:

— call_sign;

— description;

— flag_state;

— local_units;

— port_of_registration;

— ship_identification;

— ship_name.

4.2.142.1 call_sign

The call_sign specifies a unique, lifecycle identifier assigned to the ship by the flag_state for radio communication.

4.2.142.2 description

The description specifies details about the function of the ship. If the shiptype is user_defined, then the description provides the information for the type of the ship. The description need not be specified for a particular Ship_designation.

4.2.142.3 flag_state

The flag_state specifies the national authority with which the ship is registered.

4.2.142.4 local_units

The local_units specifies the units that are used by the definition if they differ from the units globally defined for the ship.

NOTE For Ship_designation, the local_units attribute inherited from Definition (see 4.2.80.3) has been redeclared in the ARM to be a set of zero, which is interpreted to mean that the attribute shall not be populated in the AIM.

4.2.142.5 port_of_registration

The port_of_registration specifies the national homeport of the ship. The port_of_registration lies within the jurisdiction of the flag_state.

4.2.142.6 ship_identification

The ship_identification specifies a general identifier unique to the ship assigned during the classification process.

NOTE The ship_identification may be either the IMO number or the Pennant hull number.

4.2.142.7 ship_name

The ship_name specifies the name of the ship assigned by the owner.

4.2.143 Shiptype

A Shiptype is a type of Functional_Definition (see 4.2.99) that is a description of the function, purpose, or mission for which a ship is designed. Each Shiptype is either a Carrier (see 4.2.18), a Navy_ship (see 4.2.128), a Research_ship (see 4.2.138), or a Working_ship (see 4.2.185).

NOTE A ship may have multiple functions. In this case several instances of this type may be defined and assigned to an instance of the Ship object. Only the major types are pre-defined; other Shiptypes may be assigned via the user_defined type with clarification of the function specified in the user_def_function attribute.

The data associated with a Shiptype are the following:

— defined_for;

— description.

4.2.143.1 defined_for

The defined_for specifies the ship for which the ship type is defined. There may be more than one defined_for for a Shiptype. See 4.3.115 for the application assertion.

4.2.143.2 description

The description specifies a function, purpose, or mission that a ship is designed for.

4.2.144 Shipyard_designation

A Shipyard_designation is a type of General_characteristics_definition (see 4.2.103) that specifies the identification given to the ship by the shipbuilder.

The data associated with a Shipyard_designation are the following:

— local_units;

— role;

— shipyard;

— shipyard_new_building_id;

— shipyard_project_name.

4.2.144.1 local_units

The local_units specifies the units that the Definition makes use of if they differ from the units globally defined for the ship.

NOTE For Shipyard_designation, the local_units attribute inherited from Definition (see 4.2.80.3) has been redeclared in the ARM to be a set of zero, which is interpreted to mean that the attribute shall not be populated in the AIM.

4.2.144.2 role

The role specifies the contractual obligation the shipyard has in relation to the ship.

The value of role is one of the following:

— prime;

— prime_build;

— prime_design;

— prime_repair;

— subcontractor.

NOTE See 4.2.144.2.1 - 4.2.144.2.5 for the definition of each allowable value for role.

4.2.144.2.1 prime

The shipyard is the prime contractor for the ship.

4.2.144.2.2 prime_build

The shipyard is the prime contractor with contract responsibilities for the manufacture of the ship.

4.2.144.2.3 prime_design

The shipyard is the prime contractor with contract responsibility for the design of the ship.

4.2.144.2.4 prime_repair

The shipyard is the prime contractor with contract responsibilities for the repair of the ship.

4.2.144.2.5 subcontractor

The shipyard is a subcontractor for the ship.

4.2.144.3 shipyard

The shipyard specifies the name and organizational details of the shipyard.

4.2.144.4 shipyard_new_building_id

The shipyard_new_building_id specifies an identifier for the ship that is assigned by the shipyard after an order has been confirmed. The shipyard_new_building_id need not be specified for a particular Shipyard_designation.

4.2.144.5 shipyard_project_name

The shipyard_project_name specifies an identifier for the ship that is assigned by the shipyard on receipt of an order, or tender, for a new ship.

4.2.145 Space

A Space is a type of Item (see 4.2.109) that is a defined volume on board a ship. A Space may have a functional definition, design definitions, and product structure definitions relating applicable properties to the Space. Each Space is either a Compartment (see 4.2.37), a Deck_zone (see 4.2.76), or a Zone (see 4.2.186).

4.2.146 Space_adjacency_relationship

A Space_adjacency_relationship is a type of Space_arrangement_relationship (see 4.2.147) that identifies spaces that share a common boundary. These spaces may or may not be accessible from one another and may share all or a portion of a boundary.

The data associated with a Space_adjacency_relationship are the following:

— adjacency_access;

— adjacency_type;

— adjacent_space_surface_area.

4.2.146.1 adjacency_access

The adjacency_access specifies that it is or is not intended to provide a means to allow passage of a person between the two adjacent spaces. A value of TRUE specifies the design intent of accessibility.

4.2.146.2 adjacency_type

The adjacency_type specifies whether the two adjacent spaces are completely or partially adjacent.

The value of adjacency_type is one of the following:

— complete;

— partial.

NOTE See 4.2.146.2.1 - 4.2.146.2.2 for the definition of each allowable value for adjacency_type.

4.2.146.2.1 complete

The two spaces have identical boundaries with respect to a specific orientation.

EXAMPLE Both share a common forward longitudinal extent, or a common port transverse extent.

4.2.146.2.2 partial

The boundaries are not identical.

4.2.146.3 adjacent_space_surface_area

The adjacent_space_surface_area specifies the area of that portion of the boundary between adjacent spaces that is common to both spaces. See 4.3.116 for the application assertion.

4.2.147 Space_arrangement_relationship

A Space_arrangement_relationship is a type of Item_relationship (see 4.2.110) that represents an association between two spaces. The collection of the set of any particular category of relationships defines a network of inter-related spaces. Each Space_arrangement_relationship is either a Space_enclosing_relationship (see 4.2.149), a Space_adjacency_relationship (see 4.2.146), a Space_connection_relationship (see 4.2.148), or a Space_positional_relationship (see 4.2.150).

The data associated with a Space_arrangement_relationship are the following:

— item_1;

— item_2.

4.2.147.1 item_1

The item_1 specifies the first Space participating in the relationship. See 4.3.117 for the application assertion.

4.2.147.2 item_2

The item_2 specifies the second Space participating in the relationship. See 4.3.117 for the application assertion.

4.2.148 Space_connection_relationship

A Space_connection_relationship is a type of Space_arrangement_relationship (see 4.2.147) that identifies spaces that are intended to be interconnected in some way.

EXAMPLE Two tanks may be interconnected by a piping system to allow the transfer of ballast water between the tanks.

The data associated with a Space_connection_relationship are the following:

— connecting_system.

4.2.148.1 connecting_system

The connecting_system specifies the identification of the system that connects the two spaces through the use of an External_instance_reference to a system, or a Label. The connecting_system need not

be specified for a particular Space_connection_relationship. See 4.3.118 for the application assertion.

4.2.149 Space_enclosing_relationship

A Space_enclosing_relationship is a type of Space_arrangement_relationship (see 4.2.149) that identifies a space that is contained within another space. The two spaces may share one or more boundaries, or a portion of a boundary.

EXAMPLE A lube oil storage tank located within the main engine room.

The data associated with a Space_enclosing_relationship are the following:

— item_1;

— item_2.

4.2.149.1 item_1

The item_1 specifies the space that is enclosed within the item_2 space. See 4.3.119 for the application assertion.

4.2.149.2 item_2

The item_2 specifies the space that contains the item_1 space. See 4.3.119 for the application assertion.

4.2.150 Space_positional_relationship

A Space_positional_relationship is a type of Space_arrangement_relationship (see 4.2.147) that identifies a space whose position is dependent upon another space. A variety of positional relationship types are supported that define the significant aspects of the relationship.

The data associated with a Space_positional_relationship are the following:

— relationship_type.

4.2.150.1 relationship_type

The relationship_type specifies an indicator as to the kind of space positional relationship expressed. The relationship is defined in terms of topological aspects of the two related spaces.

The value of relationship_type is one of the following:

— above;

— aft_longitudinal_extent;

— below;

— forward_longitudinal_extent;

— matched_transverse;

— port_transverse_extent;

— relative;

— starboard_transverse_extent.

NOTE See 4.2.150.1.1 - 4.2.150.1.8 for the definition of each allowable value for relationship_type.

4.2.150.1.1 above

The space in item_1 is intended to be above the space in item_2.

4.2.150.1.2 aft_longitudinal_extent

The aftmost boundaries of the spaces in item_1 and item_2 are intended to have the same longitudinal position in the ship.

4.2.150.1.3 below

The space in item_1 is intended to be below the space in item_2.

4.2.150.1.4 forward_longitudinal_extent

The forwardmost boundaries of the spaces in item_1 and item_2 are intended to have the same longitudinal position in the ship.

4.2.150.1.5 matched_transverse

The spaces in item_1 and item_2 are intended to have the same longitudinal positions in the ship and to be mirrored across the longitudinal centreline.

4.2.150.1.6 port_transverse_extent

The portmost boundaries of the spaces in item_1 and item_2 are intended to have the same transverse position in the ship.

4.2.150.1.7 relative

The locations of the two spaces are fixed relative to one another.

4.2.150.1.8 starboard_transverse_extent

The starboardmost boundaries of the spaces in item_1 and item_2 are intended to have the same transverse position in the ship.

4.2.151 Space_product_structure

A Space_product_structure is a type of Item (see 4.2.109) and Item_structure (see 4.2.111) that serves as a collection of External_instance_references to items that are contained within a compartment or zone. It may be defined to consist of references to items from any one or more disciplines. Alternatively, the Space_product_structure may contain compartment items to define a collection of spaces in an arrangement.

The data associated with a Space_product_structure are the following:

— contained_in;

— product_structure_type.

4.2.151.1 contained_in

The contained_in specifies the Space that all items are contained within, or for which they form the boundary. See 4.3.120 for the application assertion.

EXAMPLE In the case of a Space_Product_structure used to reference all of the part Items contained in a Space, the contained_in would be the appropriate Compartment Item. In the case of a Space_product_structure used to define a structure of Compartments, the contained_in would be a Zone Item, such as a design zone or arrangement zone representing the compartment and access drawing for a deck of the ship.

4.2.151.2 product_structure_type

The product_structure_type specifies whether the Product Structure identifies the items in a single compartment, or the compartments in a compartment arrangement.

The value of product_structure_type is one of the following:

— compartments_in_arrangement;

— items_in_compartment.

NOTE See 4.2.151.2.1 - 4.2.151.2.2 for the definition of each allowable value for product_structure_type.

4.2.151.2.1 compartments_in_arrangement

The Space_product_structure contains references or external instance references to items that are compartments within an arrangement zone.

4.2.151.2.2 items_in_compartment

The Space_product_structure contains external instance references to items that are parts or systems within or bounding a compartment.

4.2.152 Space_product_structure_revision

A Space_product_structure_revision is a type of Revision_with_context (see 4.2.140) that relates the versions of Items in a Space_product_structure to a particular version of the Space_product_-structure.

NOTE The versions of Items are related through a particular Design_definition for each Item in the Item_structure.external_items or .items attributes.

The data associated with a Space_product_structure_revision are the following:

— context_of_revision;

— members.

4.2.152.1 context_of_revision

The context_of_revision specifies the Space_product_structure for which the revision is defined. See 4.3.122 for the application assertion.

4.2.152.2 members

The members specifies the Design_definition objects that are related to the Space_product_-structure_revision. There may be more than one members for a Space_product_structure_revision. See 4.3.121 for the application assertion.

4.2.153 Spacing_position

A Spacing_position is a location on one of the global coordinate axes of the ship that is used as a reference point for any geometrical or structural item during the design and manufacture of the ship. Each Spacing_position is either a Longitudinal_position (see 4.2.123), a Transversal_position (see 4.2.166), a Vertical_position (see 4.2.181), a Spacing_position_with_offset (see 4.2.154), or a combination of Spacing_position_with_offset with one of the other three objects stated previously.

EXAMPLE Typically spacing positions are specified by LFR 123, TFR 10, 100, 100.1, A. In addition the distance to the global origin is defined, for instance, by 154.5 metres.

The data associated with a Spacing_position are the following:

— name;

— position;

— position_number.

4.2.153.1 name

The name specifies a label that is used to identify the reference point. The name need not be specified for a particular Spacing_position.

4.2.153.2 position

The position specifies the distance to the origin of the global coordinate system of the ship. The axis on which the distance is measured depends on the type of Spacing_position.

4.2.153.3 position_number

The position_number specifies the numerical identification that is given to the Spacing_position.

4.2.154 Spacing_position_with_offset

A Spacing_position_with_offset is a type of Spacing_position (see 4.2.153) that is a position defined by an offset to an existing Spacing_position on one of the global coordinate axes of the ship. It is used as a reference point for any geometrical or structural item during the design and manufacture of the ship.

The data associated with a Spacing_position_with_offset are the following:

— offset;

— position;

— relating_spacing_position.

4.2.154.1 offset

The offset specifies the distance to the relating Spacing_position. The axis on which the distance is measured depends on the type of the relating Spacing_position.

4.2.154.2 position

The position specifies the distance to the origin of the global coordinate system of the ship. The axis on which the distance is measured depends on the type of the relating Spacing_position.

4.2.154.3 relating_spacing_position

The relating_spacing_position specifies the Spacing_position from which the offset is taken to identify the Spacing_position_with_offset. The relating_spacing_position shall not be a reference to another Spacing_position_with_offset. See 4.3.123 for the application assertion.

4.2.155 Spacing_table

A Spacing_table is a type of Definition (see 4.2.80) that is a collection of Spacing_position objects that defines a list of reference points along one of the coordinate axes of the ship. Each Spacing_table may be one of the following: a Longitudinal_table (see 4.2.124), a Transversal_table (see 4.2.167), or a Vertical_table (see 4.2.182).

The data associated with a Spacing_table are the following:

— description;

— name;

— spacing_table_representations.

4.2.155.1 description

The description specifies the textual account of the reason for creation of the Spacing_table and any additional text that is required to describe the purpose of the Spacing_table. The description need not be specified for a particular Spacing_table.

4.2.155.2 name

The name specifies the context specific identification for the Spacing_table. The name need not be specified for a particular Spacing_table.

4.2.155.3 spacing_table_representations

The spacing_table_representations specifies the positions that make up the table on the coordinate axis that are of interest. The spacing_table_representations need not be specified for a particular Spacing_table. There may be more than one spacing_table_representations for a Spacing_table. See 4.3.124 for the application assertion.

4.2.156 Stability_definition

A Stability_definition is a type of Design_definition (see 4.2.82) that defines the stability properties for a given ship. The results are defined in a tabular form for different loading conditions and represent the righting arms and the centre of buoyancy for different heel angles. Each Stability_definition may be a Damage_stability_definition (see 4.2.72).

The data associated with a Stability_definition are the following:

— defined_for;

— representations.

4.2.156.1 defined_for

The defined_for specifies the ship for which the Stability_definition is defined. There may be more than one defined_for for a Stability_definition. See 4.3.125 for the application assertion.

4.2.156.2 representations

The representations specifies the Stability_table which the Stability_definition represents. There may be more than one representations for a Stability_definition. See 4.3.126 for the application assertion.

4.2.157 Stability_properties_for_one_floating_position

A Stability_properties_for_one_floating_position is the collection of values that represent the data specific to stability calculations for a particular Floating_position.

The data associated with a Stability_properties_for_one_floating_position are the following:

— centre_of_gravity_above_keel;

— definition_of_starting_floating_position;

— related_stability_table;

— stability_properties_for_different_angles_of_heel.

4.2.157.1 centre_of_gravity_above_keel

The centre_of_gravity_above_keel specifies the location for the centre of gravity of a ship. It will be presumed that the location for the centre of gravity of the ship will not change for different heel angles. The X-coordinate is measured from the aft perpendicular, the Y-coordinate is measured from the centreline and the Z-coordinate is measured from the baseline. See 4.3.127 for the application assertion.

4.2.157.2 definition_of_starting_floating_position

The definition_of_starting_floating_position specifies the Floating_position of the ship for specific loading conditions, which is used as the starting point for the calculations of the stability properties. The Floating_position covers the draught, trim, heel, and corresponding volume of displacement for the ship. See 4.3.128 for the application assertion.

4.2.157.3 related_stability_table

The related_stability_table specifies the Stability_table with which the Stability_properties_for_-one_floating_position is associated. There may be more than one related_stability_table for a Stability_properties_for_one_floating_position. See 4.3.130 for the application assertion.

4.2.157.4 stability_properties_for_different_angles_of_heel

The stability_properties_for_different_angles_of_heel specifies the Stability_property that is valid for the particular Definition_of_starting_floating_position. There may be more than one stability_-

properties_for_different_angles_of_heel for a Stability_properties_for_one_floating_position. See 4.3.129 for the application assertion.

4.2.158 Stability_property

A Stability_property is all the necessary properties for the stability caculations for a ship with specific loading conditions.

The data associated with a Stability_property are the following:

— angle_of_heel;

— centre_of_buoyancy;

— righting_arm.

4.2.158.1 angle_of_heel

The angle_of_heel specifies the angle of rotation of the ship around the X-axis. The angle_of_heel has positive values if the starboard side of the ship moves down.

4.2.158.2 centre_of_buoyancy

The centre_of_buoyancy specifies the location of the volumetric centre of the submerged ship moulded form volume. The X-coordinate is measured from the aft perpendicular, the Y-coordinate is measured from the centreline, and the Z-coordinate is measured from the baseline. See 4.3.131 for the application assertion.

4.2.158.3 righting_arm

The righting_arm specifies the distance between a perpendicular line through the centre_of_gravity_above_keel and a perpendicular line through the centre_of_buoyancy. Both lines are taken orthogonal to the heeled waterline.

4.2.159 Stability_table

A Stability_table is the stability properties for a given intact or damaged ship, depending on the stability definition. The results are defined in a tabular form and represent the righting arms and the centre of buoyancy for different heel angles for one starting Floating_position.

The data associated with a Stability_table are the following:

— items;

— mean_shell_thickness;

— name.

4.2.159.1 items

The items specifies the stability property that is the righting arms and the centre of buoyancy for a heel angle for one starting Floating_position for which the Stability_table is defined. There may be more than one items for a Stability_table. See 4.3.132 for the application assertion.

4.2.159.2 mean_shell_thickness

The mean_shell_thickness specifies the real value for the average thickness of the shell plating, which may be used to define the related extreme form for the stability properties for the ship including the thickness.

4.2.159.3 name

The name specifies a user or system defined name for the Stability_table.

4.2.160 Station_table

A Station_table is a type of Longitudinal_table (see 4.2.124) that has positions that reference the location of stations that are located along the global X-axis.

NOTE Stations are used in the design process of a ship and the station curves are curves of transversal sections through the hull of the ship. There are usually 20 stations, but the number can differ from shipyard to shipyard.

4.2.161 Tank_compartment_property

A Tank_compartment_property is a type of Compartment_property (see 4.2.56) that specifies properties for compartments designated for carrying fluid cargo.

The data associated with a Tank_compartment_property are the following:

— design_properties;

— design_stowage_density;

— geometric_parameters;

— liquid_capacity;

— moments_of_inertia.

4.2.161.1 design_properties

The design_properties specifies properties of the tank required for the design of piping systems supplying the tank. The design_properties need not be specified for a particular Tank_compartment_property. See 4.3.136 for the application assertion.

4.2.161.2 design_stowage_density

The design_stowage_density specifies the measure of the quantity per unit volume of the Liquid_cargo for which the tank compartment is designed.

4.2.161.3 geometric_parameters

The geometric_parameters specifies geometric properties of the tank compartment used for analysis of fluid cargo sloshing. The geometric_parameters need not be specified for a particular Tank_compartment_property. See 4.3.135 for the application assertion.

4.2.161.4 liquid_capacity

The liquid_capacity specifies volumetric characteristics of a tank compartment. The liquid_capacity need not be specified for a particular Tank_compartment_property. There may be more than one liquid_capacity for a Tank_compartment_property. See 4.3.133 for the application assertion.

4.2.161.5 moments_of_inertia

The moments_of_inertia specifies the values of the area moments of the boundary formed by the intersection of the compartment and a plane representing the cargo to non-cargo interface, which indicates inertial resistance to motion of a fluid cargo in the tank. See 4.3.134 for the application assertion.

4.2.162 Tank_geometric_parameters

A Tank_geometric_parameters is the geometric properties of the tank compartment used for analysis of fluid cargo sloshing.

The data associated with a Tank_geometric_parameters are the following:

— breadth_wash;

— length_wash.

4.2.162.1 breadth_wash

The breadth_wash specifies breadth between effective wash bulkheads at the height of the load point. The breadth_wash need not be specified for a particular Tank_geometric_parameters.

4.2.162.2 length_wash

The length_wash specifies length between effective wash bulkheads at the height of the load point. The length_wash need not be specified for a particular Tank_geometric_parameters.

4.2.163 Tank_piping_design_properties

A Tank_piping_design_properties is a collection of piping properties applicable to a tank compartment that provide requirements for the design of the applicable piping system.

The data associated with a Tank_piping_design_properties are the following:

— airpipe_height;

— filling_height;

— relief_valve_pressure_setting;

— sounding_pipe_height.

4.2.163.1 airpipe_height

The airpipe_height specifies height from the base line to the top of the air pipe, if any. The airpipe_height need not be specified for a particular Tank_piping_design_properties.

4.2.163.2 filling_height

The filling_height specifies the maximum height for filling of the tank compartment. The filling_height need not be specified for a particular Tank_piping_design_properties.

4.2.163.3 relief_valve_pressure_setting

The relief_valve_pressure_setting specifies the opening pressure of the relief valve. The relief_valve_pressure_setting need not be specified for a particular Tank_piping_design_properties.

4.2.163.4 sounding_pipe_height

The sounding_pipe_height specifies the height of a sounding pipe. The sounding_pipe_height need not be specified for a particular Tank_piping_design_properties.

4.2.164 Tonnage_definition

A Tonnage_definition is a type of Definition (see 4.2.80) that defines a method of volume calculation applied to ships.

NOTE It is used for determining charges for facilities such as berthing, docking, and passage through canals and locks.

The data associated with a Tonnage_definition are the following:

— certificate;

— compensated_gross_tonnage;

— defined_for;

— gross_tonnage;

— net_tonnage;

— spaces_excluded;

— tonnage_regulation.

4.2.164.1 certificate

The certificate specifies the document that is issued to the ship owner by the authority that carried out the tonnage calculations. See 4.3.139 for the application assertion.

4.2.164.2 compensated_gross_tonnage

The compensated_gross_tonnage specifies the Gross_tonnage value compensated for the type and complexity of the vessel. See 4.3.138 for the application assertion.

4.2.164.3 defined_for

The defined_for specifies the ship for which a Tonnage_definition is valid. There may be more than one defined_for for a Tonnage_definition. See 4.3.142 for the application assertion.

4.2.164.4 gross_tonnage

The gross_tonnage specifies the value and derivation of the Gross_tonnage calculation. See 4.3.140 for the application assertion.

4.2.164.5 net_tonnage

The net_tonnage specifies the value and derivation of the Net_tonnage calculation. See 4.3.141 for the application assertion.

4.2.164.6 spaces_excluded

The spaces_excluded specifies the set of compartments that are excluded from the tonnage calculations. The spaces_excluded need not be specified for a particular Tonnage_definition. There may be more than one spaces_excluded for a Tonnage_definition. See 4.3.137 for the application assertion.

4.2.164.7 tonnage_regulation

The tonnage_regulation specifies the regulations that are used to produce the tonnage calculations.

The value of tonnage_regulation is one of the following:

— convention1969;

— panama;

— suez.

NOTE See 4.2.164.7.1 - 4.2.164.7.3 for the definition of each allowable value for tonnage_regulation.

4.2.164.7.1 convention1969

The applicable regulation is a tonnage regulation for ships transporting cargo based on IMO I713E.

4.2.164.7.2 panama

The applicable regulation is a tonnage regulation for ships transporting cargo through the Panama canal zone.

4.2.164.7.3 suez

The applicable regulation is a tonnage regulation for ships transporting cargo through the Suez canal zone.

4.2.165 Tonnage_measurement

A Tonnage_measurement is the calculation of tonnage and the spaces that are included to obtain that measurement. Each Tonnage_measurement may be one of the following: a Net_tonnage (see 4.2.129), or a Gross_tonnage (see 4.2.107).

The data associated with a Tonnage_measurement are the following:

— date_of_measurement;

— spaces_included;

— tonnage_value.

4.2.165.1 date_of_measurement

The date_of_measurement specifies the date and time that the measurement was acquired.

4.2.165.2 spaces_included

The spaces_included specifies the Compartment_group included in the tonnage calculation. The spaces_included need not be specified for a particular Tonnage_measurement. There may be more than one spaces_included for a Tonnage_measurement. See 4.3.143 for the application assertion.

4.2.165.3 tonnage_value

The tonnage_value specifies the numerical value resulting from the tonnage calculation.

4.2.166 Transversal_position

A Transversal_position is a type of Spacing_position (see 4.2.153) that is located on the global Y-axis.

NOTE Each Transversal_position may also be combined with a Spacing_position_with_offset (see 4.2.154).

4.2.167 Transversal_table

A Transversal_table is a type of Spacing_table (see 4.2.155) that has positions that are along the transverse axis of the global coordinate system that is the global Y-axis. Each Transversal_table may be a Buttock_table (see 4.2.10).

The data associated with a Transversal_table are the following:

— spacing_table_representations.

4.2.167.1 spacing_table_representations

The spacing_table_representations specifies the transversal position for which the transversal table is defined. There may be more than one spacing_table_representations for a Transversal_table. See 4.3.144 for the application assertion.

4.2.168 Unit_cargo

A Unit_cargo is a type of Dry_cargo (see 4.2.89) that is cargo that is packed or comprises discrete units that can be loaded and stored individually on the ship.

The data associated with an Unit_cargo are the following:

— bounding_space;

— cargo_type;

— footprints;

— lashing_points;

— shape_description;

— stack_limit;

— volume;

— weight_and_centre_of_gravity.

4.2.168.1 bounding_space

The bounding_space specifies the representation of the total space needed for stowage of the Unit_cargo. This may be the same as the shape_description but may also include the surrounding space required for inspection and maintenance. The bounding_space may be either a non-manifold surface shape representation or a bounding box representation. The bounding_space need not be specified for a particular Unit_cargo. See 4.3.146 and 4.3.147 for the application assertions.

4.2.168.2 cargo_type

The cargo_type specifies the type of Unit_cargo that can be loaded into the ship.

The value of cargo_type is one of the following:

— aircraft;

— boat;

— cable;

— container;

— drums;

— livestock;

— pallet;

— trailer;

— undefined;

— user_defined;

— vehicle.

NOTE See 4.2.168.2.1 - 4.2.168.2.11 for the definition of each allowable value for cargo_type.

4.2.168.2.1 aircraft

The Unit_cargo is aircraft.

4.2.168.2.2 boat

The Unit_cargo is a boat.

4.2.168.2.3 cable

The Unit_cargo is cable.

4.2.168.2.4 container

The Unit_cargo is a container.

4.2.168.2.5 drums

The Unit_cargo is contained in drums.

4.2.168.2.6 livestock

The Unit_cargo is livestock.

4.2.168.2.7 pallet

The Unit_cargo is palletized.

4.2.168.2.8 trailer

The Unit_cargo is a trailer.

4.2.168.2.9 undefined

The Unit_cargo is not specified.

4.2.168.2.10 user_defined

The Unit_cargo is defined by the user.

4.2.168.2.11 vehicle

The Unit_cargo is a vehicle.

4.2.168.3 footprints

The footprints specifies the areas of the Unit_cargo that are in contact with the ship deck or hanging point. The footprints need not be specified for a particular Unit_cargo. There may be more than one footprints for a Unit_cargo. See 4.3.145 for the application assertion.

EXAMPLE There may be more than one footprints for a Unit_cargo; a car has a footprint for each wheel.

4.2.168.4 lashing_points

The lashing_points specifies the points at which lashings are secured to the Unit_cargo. These points are specified in the local coordinate system of the Unit_cargo. There may be more than one lashing_points for a Unit_cargo. The lashing_points need not be specified for a Unit_cargo.

4.2.168.5 shape_description

The shape_description specifies a non-manifold surface shape representation of the true shape of the Unit_cargo. The shape_description need not be specified for a particular Unit_cargo. See 4.3.146 for the application assertion.

4.2.168.6 stack_limit

The stack_limit specifies the maximum number of this type of Unit_cargo that can be stacked on top of each other. The stack_limit need not be specified for a particular Unit_cargo.

4.2.168.7 volume

The volume specifies the volume of the Unit_cargo. The volume need not be specified for a particular Unit_cargo.

4.2.168.8 weight_and_centre_of_gravity

The weight_and_centre_of_gravity specifies the definition of the Unit_cargo weight and centre of gravity with respect to that local coordinate system, whose origin is at the base of the centre of plane area of the cargo. The weight_and_centre_of_gravity need not be specified for a particular Unit_cargo. See 4.3.148 for the application assertion.

4.2.169 Unit_cargo_assignment

A Unit_cargo_assignment is a type of Cargo_assignment (see 4.2.13) that is an allocation of Unit_cargo to a compartment or a zone on a deck for loading analysis during the design phase or to identify an actual cargo loading during the operating phase.

The data associated with a Unit_cargo_assignment are the following:

— assigned_to;

— position.

4.2.169.1 assigned_to

The assigned_to specifies the type of location to which the cargo is assigned. The assigned_to need not be specified for a particular Unit_cargo_assignment. See 4.3.150 and 4.3.151 for the application assertions.

4.2.169.2 position

The position specifies the position of the Unit_cargo within the compartment or on the deck where it has been loaded. See 4.3.149 for the application assertion.

4.2.170 Unit_cargo_bounding_box

A Unit_cargo_bounding_box is the cube circumscribing a Unit_cargo. It is defined by two cartesian points for opposite corners of the bounding box. The edges of the cube are parallel to the coordinate axes.

The data associated with an Unit_cargo_bounding_box are the following:

— point_max;

— point_min.

4.2.170.1 point_max

The point_max specifies the opposite corner from the point_min that has maximum coordinate values.

4.2.170.2 point_min

The point_min specifies the corner of the bounding box that has minimum coordinate values.

4.2.171 Unit_cargo_group

A Unit_cargo_group is a collector of unit cargoes that may be aggregated for use in analysis. Each Unit_cargo_group may be a Person_group (see 4.2.132).

The data associated with a Unit_cargo_group are the following:

— cargo;

— footprint;

— volume;

— weight_and_centre_of_gravity.

4.2.171.1 cargo

The cargo specifies the set of Unit_cargo objects being grouped. There may be more than one cargo for a Unit_cargo_group. See 4.3.153 for the application assertion.

4.2.171.2 footprint

The footprint specifies the combined area of space taken up by the group. See 4.3.152 for the application assertion.

4.2.171.3 volume

The volume specifies the combined volume of space taken up by the unit cargoes in the group.

4.2.171.4 weight_and_centre_of_gravity

The weight_and_centre_of_gravity specifies the combined weight of all the unit cargoes in the group. See 4.3.154 for the application assertion.

4.2.172 Universal_resource_locator

A Universal_resource_locator is the location of an electronic data source as an Internet address.

EXAMPLE http://www.w3.org.

The data associated with a Universal_resource_locator is the following:

— location.

4.2.172.1 location

The location specifies the storage place where a document or other information is located on the Internet.

NOTE Such a location is typically composed of several components such as ftp or http protocol, machine address name or IP number, an optional port number, and a local path.

EXAMPLE 1 The location of a text document is: http://www.w3.org/Addressing/rfc1738.txt.

EXAMPLE 2 The location of a spreadsheet document is: ftp://ftp.atlantec-es.com/pub/out/AP218/post-ws/mapping status.xls.

4.2.173 Vehicle_load_description

A Vehicle_load_description is all of the properties that are required to estimate the impact of a vehicle on a deck.

The data associated with a Vehicle_load_description are the following:

— load_handling;

— load_per_wheel;

— max_tyre_pressure;

— number_of_wheels;

— print_area;

— type_of_vehicle.

4.2.173.1 load_handling

The load_handling specifies whether or not the vehicle is for load handling only or will stay onboard when at sea.

NOTE A value of true indicates that the vehicle is for load handling only.

4.2.173.2 load_per_wheel

The load_per_wheel specifies the maximum permitted load per wheel.

4.2.173.3 max_tyre_pressure

The max_tyre_pressure specifies the maximum tyre pressure; only required if print_area is not known. The max_tyre_pressure need not be specified for a particular Vehicle_load_description.

4.2.173.4 number_of_wheels

The number_of_wheels specifies minimum number of wheels per vehicle.

4.2.173.5 print_area

The print_area specifies the minimum area of a wheel that touches the deck. The print_area need not be specified for a particular Vehicle_load_description.

4.2.173.6 type_of_vehicle

The type_of_vehicle specifies the name of the type of vehicle.

4.2.174 Version_creation

A Version_creation is a type of Versionable_object_change_event (see 4.2.180) that represents the event leading to a new Definition, Item_structure, or Item_relationship.

The data associated with a Version_creation are the following:

— base;

— subject.

4.2.174.1 base

The base specifies the Versionable_object objects from which the subject Versionable_object is derived, if one or more exists. The base need not be the immediately preceding version of the Versionable_object referenced in the subject attribute, but it may refer to any previous version in the version history of the same Item, or to any Versionable_object of another Item which contributes to the creation of the subject. The base need not be specified for a particular Version_creation. There may be more than one base for a Version_creation. See 4.3.155 for the application assertion.

4.2.174.2 subject

The subject specifies the Versionable_object created by the change event. See 4.3.155 for the application assertion.

4.2.175 Version_deletion

A Version_deletion is a type of Versionable_object_change_event (see 4.2.180) that identifies the Event leading to the deletion of a Definition, an Item_structure, or an Item_relationship.

The data associated with a Version_deletion are the following:

— subject.

4.2.175.1 subject

The subject specifies the Versionable_object deleted or to be deleted by the change event. See 4.3.156 for the application assertion.

4.2.176 Version_history

A Version_history is an identification of Versionable_object objects and their Version_relationship objects in terms of their role as predecessors, successors, and with respect to each other.

NOTE The Version_history is a directed acyclic graph. Consequently the Version_history may contain Versionable_object objects considered alternatives with respect to each other, i.e., a Versionable_object having more than one successor, and merged Versionable_object objects, i.e., a Versionable_object having more than one predecessor.

The data associated with a Version_history are the following:

— current_version;

— relationships;

— versions.

4.2.176.1 current_version

The current_version specifies the Versionable_object that plays the role of the current version in this Version_history. See 4.3.158 for the application assertion.

4.2.176.2 relationships

The relationships specifies the Version_relationship for which the Version_history is defined. The relationships need not be specified for a particular Version_history. There may be more than one relationships for a Version_history. See 4.3.157 for the application assertion.

4.2.176.3 versions

The versions specifies the Versionable_object for which the Version_history is defined. There may be more than one versions for a Version_history. See 4.3.158 for the application assertion.

4.2.177 Version_modification

A Version_modification is a type of Versionable_object_change_event (see 4.2.180) that identifies the event leading to a change of a Versionable_object. The base Versionable_object need not be the immediately preceding version of the subject Versionable_object, but may refer to any previous version in the Version_history of the same Item.

EXAMPLE A Version_modification may be the creation of a new version for an existing thing.

The data associated with a Version_modification are the following:

— base;

— subject.

4.2.177.1 base

The base specifies the Versionable_objects from which the subject is derived. There may be more than one base for a Version_modification. See 4.3.159 for the application assertion.

4.2.177.2 subject

The subject specifies the Versionable_object modified or to be modified by the Change Event. See 4.3.159 for the application assertion.

4.2.178 Version_relationship

A Version_relationship defines the relationship of two Versionable_object objects of the same type in terms of a Version_history.

The data associated with a Version_relationship are the following:

— predecessor;

— reason;

— successor.

4.2.178.1 predecessor

The predecessor specifies the Versionable_object from which the successor is derived. See 4.3.160 for the application assertion.

4.2.178.2 reason

The reason specifies the purpose for a new version.

4.2.178.3 successor

The successor specifies the Versionable_object of which the predecessor is the preceding version. See 4.3.160 for the application assertion.

4.2.179 Versionable_object

A Versionable_object is any object that may be versioned. Each Versionable_object is either a Definition (see 4.2.80), a Document (see 4.2.85), an Item_relationship (see 4.2.110), an Item_structure (see 4.2.111), or a Revision (see 4.2.139).

The data associated with a Versionable_object are the following:

— description;

— version_id.

4.2.179.1 description

The description specifies additional information that identifies or describes the object. The description need not be specified for a particular Versionable_object.

4.2.179.2 version_id

The version_id specifies the version identifier of the Versionable_object.

4.2.180 Versionable_object_change_event

A Versionable_object_change_event is a type of Event (see 4.2.91) that is the generalization of the events changing a Definition, Item_structure, or Item_relationship. Each Versionable_-object_change_event is either an Envisaged_version_creation (see 4.2.90), a Version_creation (see 4.2.174), a Version_modification (see 4.2.177), or a Version_deletion (see 4.2.175).

4.2.181 Vertical_position

A Vertical_position is a type of Spacing_position (see 4.2.153) that is located on the global Z-axis.

NOTE Each Vertical_position may also be combined with a Spacing_position_with_offset (see 4.2.154).

4.2.182 Vertical_table

A Vertical_table is a type of Spacing_table (see 4.2.155) that has positions that are along the vertical axis of the global coordinate system, which is the global Z-axis. A Vertical_table may be a Waterline_table (see 4.2.183).

The data associated with a Vertical_table are the following:

— spacing_table_representations.

4.2.182.1 spacing_table_representations

The spacing_table_representations specifies the vertical positions that make up the vertical table. The spacing_table_representations need not be specified for a particular Vertical_table. There may be more than one spacing_table_representations for a Vertical_table. See 4.3.161 for the application assertion.

4.2.183 Waterline_table

A Waterline_table is a type of Vertical_table (see 4.2.182) that has positions that reference the location of waterlines that are located along the global Z-axis.

4.2.184 Weight_and_centre_of_gravity

A Weight_and_centre_of_gravity is the weight and the centre of gravity of a ship part. Each Weight_and_centre_of_gravity may be a Lightship_weight_item (see 4.2.114).

The data associated with a Weight_and_centre_of_gravity are the following:

— centre_of_gravity;

— mass;

— moment.

4.2.184.1 centre_of_gravity

The centre_of_gravity specifies the centre of gravity of a ship part. See 4.3.162 for the application assertion.

4.2.184.2 mass

The mass specifies the weight of a ship part.

4.2.184.3 moment

The moment specifies the Moment_3d based on the centre_of_gravity and weight of a ship part. The moment need not be specified for a particular Weight_and_centre_of_gravity. See 4.3.163 for the application assertion.

4.2.185 Working_ship

A Working_ship is a type of Shiptype (see 4.2.143) that is a vessel designed to perform a particular commercial or industrial task.

The data associated with a Working_ship are the following:

— has_type.

4.2.185.1 has_type

The has_type specifies the type of the Working_ship.

The value of has_type is one of the following:

— crane_vessel;

— dredger;

— drilling_vessel;

— fire_fighter;

— fishing_vessel;

— floating_dock;

— floating_hotel;

— FPGO;

— FPSO;

— ice_breaker;

— offshore_supply_vessel;

— oil_production_and_storage_vessel;

— oil_production_vessel;

— oil_storage_vessel;

— pilot_boat;

— pipe_laying_vessel;

— pusher;

— reefer;

— sealer;

— shuttle_tanker;

— stern_trawler;

— supply_vessel;

— tug;

— user_defined;

— well_stimulation_vessel.

NOTE See 4.2.185.1.1 - 4.2.185.1.25 for the definition of each allowable value for has_type.

4.2.185.1.1 crane_vessel

The ship is constructed for lifting purposes.

4.2.185.1.2 dredger

The ship is constructed for dredging channels or harbor entrances.

4.2.185.1.3 drilling_vessel

The ship is constructed for drilling purposes.

4.2.185.1.4 fire_fighter

The ship is constructed for fire fighting purposes.

4.2.185.1.5 fishing_vessel

The ship is constructed for fishing.

4.2.185.1.6 floating_dock

The ship is constructed for lifting ships for repair purposes.

4.2.185.1.7 floating_hotel

The ship is constructed for hotel purposes.

4.2.185.1.8 FPGO

The ship is a floating platform used for production and storage of gas.

4.2.185.1.9 FPSO

The ship is a floating platform used for production and storage of oil.

4.2.185.1.10 ice_breaker

The ship is constructed for breaking ice.

4.2.185.1.11 offshore_supply_vessel

The ship is constructed for supplying offshore platforms.

4.2.185.1.12 oil_production_and_storage_vessel

The ship is used for production and storage of oil.

4.2.185.1.13 oil_production_vessel

The ship is constructed for production of oil.

4.2.185.1.14 oil_storage_vessel

The ship is used for storage of oil.

4.2.185.1.15 pilot_boat

The ship is constructed for carrying the pilot to the ship that he navigates in and out of the harbor.

4.2.185.1.16 pipe_laying_vessel

The ship is constructed for pipe laying purposes.

4.2.185.1.17 pusher

The ship is constructed to push other unpropelled ships.

4.2.185.1.18 reefer

The ship is constructed for transporting refrigerated cargo.

4.2.185.1.19 sealer

The ship is constructed for the purpose of seal hunting.

4.2.185.1.20 shuttle_tanker

The ship is equipped for offshore oil loading. It may be used for transporting oil from a platform to shore.

4.2.185.1.21 stern_trawler

The ship is constructed for trawling the sea from the stern of the ship.

4.2.185.1.22 supply_vessel

The ship is constructed for for the purpose of transporting supplies.

4.2.185.1.23 tug

The ship is constructed for towing purposes.

4.2.185.1.24 user_defined

The ship type is not one of the pre-defined types.

NOTE Details may be found in the Shiptype description attribute.

4.2.185.1.25 well_stimulation_vessel

The ship is constructed for the purposes of stimulating a well.

4.2.186 Zone

A Zone is a type of Space (see 4.2.145) that represents an abstract bounded volume identifying a region of a ship with unique requirements or characteristics that must be specially treated in the design and manufacturing processes.

4.2.187 Zone_design_definition

A Zone_design_definition is a type of Design_definition (see 4.2.82) that is the abstract definition of a version of a Zone from a design perspective.

The data associated with a Zone_design_definition are the following:

— boundaries;

— constituent_compartments;

— defined_for;

— properties;

— representations.

4.2.187.1 boundaries

The boundaries specifies External_instance_references to the ISO 10303-216 Moulded_form objects or ISO 10303-218 Structural_system objects that bound the Zone. The boundaries need not be specified for a particular Zone_design_definition. There may be more than one boundaries for a Zone_design_definition. See 4.3.166 for the application assertion.

4.2.187.2 constituent_compartments

The constituent_compartments specifies the Compartments, or portions of Compartments, which consitute the Zone. The constituent_compartments need not be specified for a particular Zone_design_definition. There may be more than one constituent_compartments for a Zone_-design_definition. See 4.3.166 and 4.3.164 for the application assertion.

4.2.187.3 defined_for

The defined_for specifies the Zones for which the Zone_design_definition is applicable. There may be more than one defined_for for a Zone_design_definition. See 4.3.168 for the application assertion.

4.2.187.4 properties

The properties specifies a collection of properties applicable to or derived from the design of a Zone. The properties need not be specified for a particular Zone_design_definition. There may be more than one properties for a Zone_design_definition. See 4.3.165 for the application assertion.

4.2.187.5 representations

The representations specifies the Compartment_shape_representations for the Zone_design_-definition. The representations need not be specified for a particular Zone_design_definition. There may be more than one representations for a Zone_design_definition. See 4.3.167 for the application assertion.

4.2.188 Zone_functional_definition

A Zone_functional_definition is a type of Functional_definition (see 4.2.99) that defines the functional role of a Zone. The role may be a pre-defined one or may be user-defined.

The data associated with a Zone_functional_definition are the following:

— defined_for;

— used_for.

4.2.188.1 defined_for

The defined_for specifies the Zones for which the Zone_functional_definition is applicable. There may be more than one defined_for for a Zone_functional_definition. See 4.3.169 for the application assertion.

4.2.188.2 used_for

The used_for specifies the name of a function that a specific Zone may have in a ship.

The value of used_for is one of the following:

— arrangement_zone;

— bottom_external_zone;

— bow_external_zone;

— bridging_structure_zone;

— damage_control_zone;

— deckhouse_zone;

— design_zone;

— double_bottom_zone;

— external_zone;

— fire_zone;

— hatch_zone;

— pressure_zone;

— ship_side_external_zone;

— stern_external_zone;

— subsafe_zone;

— superstructure_zone;

— tank_top_zone;

— tween_deck_zone;

— user_defined.

NOTE See 4.2.188.2.1 - 4.2.188.2.19 for the definition of each allowable value for used_for.

4.2.188.2.1 arrangement_zone

The zone defines an arrangement of compartments.

4.2.188.2.2 bottom_external_zone

The zone is a space external to the bottom of the ship.

4.2.188.2.3 bow_external_zone

The zone is a space external to the bow of the ship.

4.2.188.2.4 bridging_structure_zone

The zone is a space between the wet deck and the main deck between hulls on multi-hulled vessels.

4.2.188.2.5 damage_control_zone

The zone defines the boundaries of a damage control area.

4.2.188.2.6 deckhouse_zone

The zone is a space above main deck intended for passengers, crew, stores or equipment.

4.2.188.2.7 design_zone

The zone defines the boundaries of an area managed as a unit during design development.

4.2.188.2.8 double_bottom_zone

The zone consists of compartments at the bottom of a ship between the inner bottom and the bottom shell plating.

4.2.188.2.9 external_zone

The zone is any space external to the structure of the ship.

4.2.188.2.10 fire_zone

The zone defines the boundaries of a fire protection area.

4.2.188.2.11 hatch_zone

The zone is a space limited by the hatch cover, deck level and hatch coamings.

4.2.188.2.12 pressure_zone

The zone defines the boundaries of a positive pressure zone.

4.2.188.2.13 ship_side_external_zone

The zone is a space external to the ship side.

4.2.188.2.14 stern_external_zone

The zone is a space external to the stern of the ship.

4.2.188.2.15 subsafe_zone

The zone defines the boundaries of a submarine safety zone.

4.2.188.2.16 superstructure_zone

The zone is a space above main deck intended for passengers, crew, stores or equipment.

4.2.188.2.17 tank_top_zone

The zone consists of compartments immediately above the inner bottom of the ship.

4.2.188.2.18 tween_deck_zone

The zone is the space between any two adjacent decks intended for carriage of cargo.

4.2.188.2.19 user_defined

The Zone function is other than one of the pre-defined values and is specified by the user_def_function attribute.

4.3 Application assertions

This subclause specifies the application assertions for the ship arrangement application protocol. Application assertions specify the relationships between application objects, the cardinality of the relationships, and the rules required for the integrity and validity of the application objects and UoFs. The application assertions and their definitions are given below.

4.3.1 Absolute_cargo_position to Global_axis_placement

Each Absolute_cargo_position has parent_coordinate_system defined by exactly one Global_axis_placement object. Each Global_axis_placement defines the parent_coordinate_system for zero, one, or many Absolute_cargo_position objects.

4.3.2 Alternative_version_relationship to Versionable_object

Each Alternative_version_relationship has alternative_1 defined by exactly one Versionable_object objects. Each Versionable_object defines alternative_1 for zero, one, or many Alternative_version_relationship objects.

Each Alternative_version_relationship has alternative_2 defined by exactly one Versionable_object objects. Each Versionable_object defines alternative_2 for zero, one, or many Alternative_version_relationship objects.

4.3.3 Approval_event to Approval_history

Each Approval_event has approval_reference defined by exactly one Approval_history object. Each Approval_history defines the approval_reference for one, or many Approval_event objects.

4.3.4 Approval_history to Definition

Each Approval_history has subject defined by exactly one Definition object. Each Definition defines the subject for zero, one, or many Approval_history objects.

4.3.5 Bay_cell_position to Longitudinal_position

Each Bay_cell_position has row defined by exactly one Longitudinal_position object. Each Longitudinal_position defines the row for zero, one, or many Bay_cell_position objects.

4.3.6 Bay_cell_position to Transversal_position

Each Bay_cell_position has bay_number defined by exactly one Transversal_position object. Each Transversal_position defines the bay_number for zero, one, or many Bay_cell_position objects.

4.3.7 Bay_cell_position to Vertical_position

Each Bay_cell_position has tier defined by exactly one Vertical_position object. Each Vertical_position defines the tier for zero, one, or many Bay_cell_position objects.

4.3.8 Bay_position to Cargo_bay_definition

Each Bay_position has relating_to defined by exactly one Cargo_bay_definition object. Each Cargo_bay_definition defines the relating_to for zero, one, or many Bay_position objects.

4.3.9 Capacity_properties to Centre_location

Each Capacity_properties has capacity_centre defined by exactly one Centre_location object. Each Centre_location defines the capacity_centre for zero, one, or many Capacity_properties objects.

Each Capacity_properties has capacity_level_origin defined by exactly one Centre_location object. Each Centre_location defines the capacity_level_origin for zero, one, or many Capacity_properties objects.

4.3.10 Cargo to Dangerous_goods_code

Each Cargo has cargo_hazard defined by zero or one Dangerous_goods_code objects. Each Dangerous_goods_code defines the cargo_hazard for zero, one, or many Cargo objects.

4.3.11 Cargo to Document_reference_with_address

Each Cargo has references defined by zero, one, or many Document_reference_with_address objects. Each Document_reference_with_address defines the references for zero, one, or many Cargo objects.

4.3.12 Cargo to General_cargo_material_properties

Each Cargo has material_properties defined by zero or one General_cargo_material_properties objects. Each General_cargo_material_properties defines the material_properties for zero, one, or many Cargo objects.

4.3.13 Cargo_assignment to Weight_and_centre_of_gravity

Each Cargo_assignment has allocated_weight defined by zero or one Weight_and_centre_of_gravity objects. Each Weight_and_centre_of_gravity defines the allocated_weight for zero, one, or many Cargo_assignment objects.

4.3.14 Cargo_bay_definition to Compartment

Each Cargo_bay_definition has defined_for defined by one or many Compartment objects. Each Compartment defines the defined_for for zero, one, or many Cargo_bay_definition objects.

4.3.15 Cargo_bay_definition to Spacing_table

Each Cargo_bay_definition has cargo_positions defined by one or many Spacing_table objects. Each Spacing_table defines the cargo_positions for zero, one, or many Cargo_bay_definition objects.

4.3.16 Cargo_compartment_property to Capacity_properties

Each Cargo_compartment_property has bulk_cargo_capacity defined by zero or one Capacity_properties objects. Each Capacity_properties defines the bulk_cargo_capacity for zero, one, or many Cargo_compartment_property objects.

4.3.17 Change_definition to Change

Each Change_definition has defined_for defined by one or many Change objects. Each Change defines the defined_for for zero, one, or many Change_definition objects.

4.3.18 Change_impact to Versionable_object_change_event

Each Change_impact has impact defined by one or many Versionable_object_change_event objects. Each Versionable_object_change_event defines the impact for zero, one, or many Change_impact objects.

4.3.19 Change_plan to Change_impact

Each Change_plan has planned_impact defined by exactly one Change_impact object. Each Change_impact defines the planned_impact for zero, one, or many Change_plan objects.

4.3.20 Change_plan to Change_request

Each Change_plan has chosen_solution_for defined by exactly one Change_request object. Each Change_request defines the chosen_solution_for for zero, one, or many Change_plan objects.

4.3.21 Change_plan to Check

Each Change_plan has checks defined by zero, one, or many Check objects. Each Check defines the checks for zero, one, or many Change_plan objects.

4.3.22 Change_realization to Change_impact

Each Change_realization has impact defined by exactly one Change_impact object. Each Change_impact defines the impact for zero, one, or many Change_realization objects.

4.3.23 Change_realization to Change_plan

Each Change_realization has realization_of defined by exactly one Change_plan object. Each Change_plan defines the realization_of for zero, one, or many Change_realization objects.

4.3.24 Change_realization to Check

Each Change_realization has checks defined by zero, one, or many Check objects. Each Check defines the checks for zero, one, or many Change_realization objects.

4.3.25 Change_request to Change_impact

Each Change_request has solution_alternatives defined by zero, one, or many Change_impact objects. Each Change_impact defines the solution_alternatives for zero, one, or many Change_request objects.

4.3.26 Class_and_statutory_designation to Class_notation

Each Class_and_statutory_designation has the_class defined by exactly one Class_notation object. Each Class_notation defines the the_class for zero, one, or many Class_and_statutory_designation objects.

4.3.27 Class_and_statutory_designation to Regulation

Each Class_and_statutory_designation has the_statutory defined by exactly one Regulation object. Each Regulation defines the the_statutory for zero, one, or many Class_and_statutory_designation objects.

4.3.28 Class_compartment_requirement_definition to Compartment

Each Class_compartment_requirement_definition has defined_for defined by one or many Compartment objects. Each Compartment defines the defined_for for zero, one, or many Class_compartment_requirement_definition objects.

4.3.29 Class_deck_load_requirement_definition to Deck_zone

Each Class_deck_load_requirement_definition has defined_for defined by one or many Deck_zone objects. Each Deck_zone defines the defined_for for zero, one, or many Class_deck_load_-requirement_definition objects.

4.3.30 Class_deck_load_requirement_definition to Vehicle_load_description

Each Class_deck_load_requirement_definition has vehicle_load defined by zero or one Vehicle_load_description objects. Each Vehicle_load_description defines the vehicle_load for zero, one, or many Class_deck_load_requirement_definition objects.

4.3.31 Coating to Coating_certification

Each Coating has certification defined by zero, one, or many Coating_certification objects. Each Coating_certification defines the certification for zero, one, or many Coating objects.

4.3.32 Compartment_cargo_assignment to Cargo

Each Compartment_cargo_assignment has cargo defined by zero or one Cargo object. Each Cargo defines the cargo for zero, one, or many Compartment_cargo_assignment objects.

4.3.33 Compartment_cargo_assignment to Compartment

Each Compartment_cargo_assignment has compartment defined by exactly one Compartment object. Each Compartment defines the compartment for zero, one, or many Compartment_cargo_assignment objects.

4.3.34 Compartment_cargo_assignment to Unit_cargo_group

Each Compartment_cargo_assignment has cargo defined by zero or one Unit_cargo_group objects. Each Unit_cargo_group defines the cargo for zero, one, or many Compartment_cargo_assignment objects.

4.3.35 Compartment_coating to Corrosion_protection

Each Compartment_coating has corrosion_protection defined by exactly one Corrosion_protection object. Each Corrosion_protection defines the corrosion_protection for zero, one, or many Compartment_coating objects.

4.3.36 Compartment_design_definition to Compartment

Each Compartment_design_definition has defined_for defined by one or many Compartment objects. Each Compartment defines the defined_for for zero, one, or many Compartment_design_definition objects.

4.3.37 Compartment_design_definition to Compartment_property

Each Compartment_design_definition has properties defined by zero, one, or many Compartment_property objects. Each Compartment_property defines the properties for zero, one, or many Compartment_design_definition objects.

4.3.38 Compartment_design_definition to External_instance_reference

Each Compartment_design_definition has boundaries defined by zero, one, or many External_-instance_reference objects. Each External_instance_reference defines the boundaries for zero, one, or many Compartment_design_definition objects.

4.3.39 Compartment_design_definition to Non_manifold_surface_shape

Each Compartment_design_definition has representations defined by zero, one, or many Non_-manifold_surface_shape objects. Each Non_manifold_surface_shape defines the representations for zero, one, or many Compartment_design_definition objects.

4.3.40 Compartment_design_requirement to Space

Each Compartment_design_requirement has defined_for defined by one or many Space objects. Each Space defines the defined_for for zero, one, or many Compartment_design_requirement objects.

4.3.41 Compartment_functional_definition to Compartment

Each Compartment_functional_definition has defined_for defined by one or many Compartment objects. Each Compartment defines the defined_for for zero, one, or many Compartment_-functional_definition objects.

4.3.42 Compartment_group to Compartment

Each Compartment_group has compartment defined by zero, one, or many Compartment objects. Each Compartment defines the compartment for zero, one, or many Compartment_group objects.

4.3.43 Compartment_volume_property to Centre_location

Each Compartment_volume_property has centre_of_volume defined by exactly one Centre_location object. Each Centre_location defines the centre_of_volume for zero, one, or many Compartment_-volume_property objects.

4.3.44 Compensated_gross_tonnage to Gross_tonnage

Each Compensated_gross_tonnage has gross_tonnage_measurement defined by exactly one Gross_tonnage object. Each Gross_tonnage defines the gross_tonnage_measurement for zero, one, or many Compensated_gross_tonnage objects.

4.3.45 Corrosion_control_coating to Primer_coating

Each Corrosion_control_coating has primer defined by exactly one Primer_coating object. Each Primer_coating defines the primer for zero, one, or many Corrosion_control_coating objects.

4.3.46 Corrosion_protection to Coating

Each Corrosion_protection has coating_material defined by exactly one Coating object. Each Coating defines the coating_material for zero, one, or many Corrosion_protection objects.

4.3.47 Corrosion_protection to Coating_level

Each Corrosion_protection has coating_height defined by exactly one Coating_level object. Each Coating_level defines the coating_height for zero, one, or many Corrosion_protection objects.

4.3.48 Damage_case to Compartment_design_definition

Each Damage_case has damaged_compartments defined by one or many Compartment_design_-definition objects. Each Compartment_design_definition defines the damaged_compartments for zero, one, or many Damage_case objects.

4.3.49 Damage_case to Damage_position

Each Damage_case has position_of_damage defined by exactly one Damage_position object. Each Damage_position defines the position_of_damage for zero, one, or many Damage_case objects.

4.3.50 Damage_case to Loading_condition_definition

Each Damage_case has original_loads defined by exactly one Loading_condition_definition object. Each Loading_condition_definition defines the original_loads for zero, one, or many Damage_case objects.

4.3.51 Damage_position to Centre_location

Each Damage_position has centre_of_damage defined by exactly one Centre_location object. Each Centre_location defines the centre_of_damage for zero, one, or many Damage_position objects.

4.3.52 Damage_stability_definition to Damage_case

Each Damage_stability_definition has extent_of_damage defined by one or many Damage_case objects. Each Damage_case defines the extent_of_damage for zero, one, or many Damage_stability_-definition objects.

4.3.53 Damage_stability_definition to Ship

Each Damage_stability_definition has defined_for defined by exactly one Ship object. Each Ship defines the defined_for for zero, one, or many Damage_stability_definition objects.

4.3.54 Damage_stability_definition to Stability_table

Each Damage_stability_definition has representations defined by one or many Stability_table objects. Each Stability_table defines the representations for zero, one, or many Damage_stability_definition objects.

4.3.55 Deadweight to Cargo_assignment

Each Deadweight has deadweight_items defined by one or many Cargo_assignment objects. Each Cargo_assignment defines the deadweight_items for zero, one, or many Deadweight objects.

4.3.56 Deck_cargo_assignment to Cargo_position

Each Deck_cargo_assignment has position defined by exactly one Cargo_position object. Each Cargo_position defines the position for zero, one, or many Deck_cargo_assignment objects.

4.3.57 Deck_cargo_assignment to Deck_zone

Each Deck_cargo_assignment has deck_zone defined by exactly one Deck_zone object. Each Deck_zone defines the deck_zone for zero, one, or many Deck_cargo_assignment objects.

4.3.58 Deck_cargo_assignment to Unit_cargo_group

Each Deck_cargo_assignment has cargo defined by exactly one Unit_cargo_group object. Each Unit_cargo_group defines the cargo for zero, one, or many Deck_cargo_assignment objects.

4.3.59 Deck_zone_design_definition to Compartment

Each Deck_zone_design_definition has constituent_compartments defined by zero, one, or many Compartment objects. Each Compartment defines the constituent_compartments for zero, one, or many Deck_zone_design_definition objects.

4.3.60 Deck_zone_design_definition to Compartment_property

Each Deck_zone_design_definition has properties defined by zero, one, or many Compartment_-property objects. Each Compartment_property defines the properties for zero, one, or many Deck_zone_design_definition objects.

4.3.61 Deck_zone_design_definition to Deck_zone

Each Deck_zone_design_definition has defined_for defined by one or many Deck_zone objects. Each Deck_zone defines the defined_for for zero, one, or many Deck_zone_design_definition objects.

4.3.62 Deck_zone_design_definition to External_instance_reference

Each Deck_zone_design_definition has deck_for_zone defined by exactly one External_instance_-reference object. Each External_instance_reference defines the deck_for_zone for zero, one, or many Deck_zone_design_definition objects.

Each Deck_zone_design_definition has constituent_compartments defined by zero, one, or many External_instance_reference objects. Each External_instance_reference defines the constituent_-compartments for zero, one, or many Deck_zone_design_definition objects.

4.3.63 Deck_zone_design_definition to Non_manifold_surface_shape

Each Deck_zone_design_definition has representations defined by zero, one, or many Non_-manifold_surface_shape objects. Each Non_manifold_surface_shape defines the representations for zero, one, or many Deck_zone_design_definition objects.

4.3.64 Deck_zone_functional_definition to Deck_zone

Each Deck_zone_functional_definition has defined_for defined by one or many Deck_zone objects. Each Deck_zone defines the defined_for for zero, one, or many Deck_zone_functional_definition objects.

4.3.65 Definable_object to Global_id

Each Definable_object has id defined by exactly one Global_id object. Each Global_id defines the id for zero, one, or many Definable_object objects.

4.3.66 Definition to Definable_object

Each Definition has defined_for defined by one or many Definable_object objects. Each Definable_-object defines the defined_for for zero, one, or many Definition objects.

4.3.67 Definition to Derived_unit

Each Definition has local_units defined by zero, one, or many Derived_unit objects. Each Derived_-unit defines the local_units for zero, one, or many Definition objects.

4.3.68 Definition to Global_id

Each Definition has id defined by exactly one Global_id object. Each Global_id defines the id for zero, one, or many Definition objects.

4.3.69 Definition to Named_unit

Each Definition has local_units defined by zero, one, or many Named_unit objects. Each Named_unit defines the local_units for zero, one, or many Definition objects.

4.3.70 Design_requirement to Document_reference_with_address

Each Design_requirement has specification defined by zero, one, or many Document_-reference_with_address objects. Each Document_reference_with_address defines the specification for zero, one, or many Design_requirement.

4.3.71 Document_portion to Document

Each Document_portion has source defined by exactly one Document object. A Document defines the source for a Document_portion.

4.3.72 Document_reference to Document

Each Document_reference has assigned_document defined by exactly one Document object. Each Document defines the assigned_document for zero, one, or many Document_reference objects.

4.3.73 Document_reference to document_portion

Each Document_reference has assigned_document defined by exactly one Document_portion object. A document_portion defines the assigned_document for a Document_reference.

4.3.74 Envisaged_version_creation to Versionable_object

Each Envisaged_version_creation has base defined by zero, one, or many Versionable_object objects. Each Versionable_object defines the base for zero, one, or many Envisaged_version_creation objects.

4.3.75 External_instance_reference to Global_id

Each External_instance_reference has target_GUID defined by exactly one Global_id object. Each Global_id defines the target_GUID for zero, one, or many External_instance_reference objects.

4.3.76 External_reference to External_storage

Each External_reference has location defined by exactly one External_storage object. Each External_storage defines the location for zero, one, or many External_reference objects.

4.3.77 External_reference to Universal_resource_locator

Each External_reference has location defined by exactly one Universal_resource_locator object. Each Universal_resource_locator defines the location for zero, one, or many External_reference objects.

4.3.78 Fire_safe_coating to Primer_coating

Each Fire_safe_coating has primer defined by exactly one Primer_coating object. Each Primer_-coating defines the primer for zero, one, or many Fire_safe_coating objects.

4.3.79 Freeboard_characteristics to Loadline

Each Freeboard_characteristics has applicable_loadline defined by exactly one Loadline object. Each Loadline defines the applicable_loadline for zero, one, or many Freeboard_characteristics objects.

4.3.80 General_cargo_material_properties to Document_reference_with_address

Each General_cargo_material_properties has material_reference defined by zero or one Document_reference_with_address objects. Each Document_reference_with_address defines the material_reference for zero, one, or many General_cargo_material_properties objects.

4.3.81 General_characteristics_definition to Ship

Each General_characteristics_definition has defined_for defined by one or many Ship objects. Each Ship defines the defined_for for zero, one, or many General_characteristics_definition objects.

4.3.82 Hull_applicability to Definition

Each Hull_applicability may apply to zero, one, or many Definition objects. Each Definition may be applicable to zero, one, or many Hull_applicability objects.

4.3.83 Hull_applicability to Item

Each Hull_applicability may apply to zero, one, or many Item objects. Each Item may be applicable to zero, one, or many Hull_applicability objects.

4.3.84 Item to External_reference

Each Item has documentation defined by zero, one, or many External_reference objects. Each External_reference defines the documentation for zero, one, or many Item objects.

4.3.85 Item to Ship

Each Item has ship_context defined by zero or one Ship objects. Each Ship defines the ship_context for zero, one, or many Item objects.

4.3.86 Item_relationship to External_instance_reference

Each Item_relationship has external_item_1 defined by zero or one External_instance_reference objects. Each External_instance_reference defines the external_item_1 for zero, one, or many Item_relationship objects.

Each Item_relationship has external_item_2 defined by zero or one External_instance_reference objects. Each External_instance_reference defines the external_item_2 for zero, one, or many Item_relationship objects.

4.3.87 Item_relationship to Item

Each Item_relationship has item_1 defined by zero or one Item objects. Each Item defines the item_1 for zero, one, or many Item_relationship objects.

Each Item_relationship has item_2 defined by zero or one Item objects. Each Item defines the item_2 for zero, one, or many Item_relationship objects.

4.3.88 Item_structure to External_instance_reference

Each Item_structure has external_items defined by zero, one, or many External_instance_reference objects. Each External_instance_reference defines the external_items for zero, one, or many Item_structure objects.

Each Item_structure has external_relationships defined by zero, one, or many External_-instance_reference objects. Each External_instance_reference defines the external_relationships for zero, one, or many Item_structure objects.

4.3.89 Item_structure to Item

Each Item_structure has items defined by zero, one, or many Item objects. Each Item defines the items for zero, one, or many Item_structure objects.

4.3.90 Item_structure to Item_relationship

Each Item_structure has relationships defined by zero, one, or many Item_relationship objects. Each Item_relationship defines the relationships for zero, one, or many Item_structure objects.

4.3.91 Lane_position to Longitudinal_position

Each Lane_position has frame_number defined by exactly one Longitudinal_position object. Each Longitudinal_position defines the frame_number for zero, one, or many Lane_position objects.

4.3.92 Lane_position to Transversal_position

Each Lane_position has lane_number defined by exactly one Transversal_position object. Each Transversal_position defines the lane_number for zero, one, or many Lane_position objects.

4.3.93 Lane_position to Vertical_position

Each Lane_position has deck_number defined by exactly one Vertical_position object. Each Vertical_position defines the deck_number for zero, one, or many Lane_position objects.

4.3.94 Lightship_definition to Centre_location

Each Lightship_definition has lightship_centre_of_gravity defined by exactly one Centre_location object. Each Centre_location defines the centre of gravity for zero, one, or many Lightship_-definition objects.

4.3.95 Lightship_definition to Lightship_weight_item

Each Lightship_definition has lightship_items defined by zero, one, or many Lightship_weight_item objects. Each Lightship_weight_item defines the lightship_items for zero, one, or many Lightship_-definition objects.

4.3.96 Lightship_definition to Ship

Each Lightship_definition has defined_for defined by one or many Ship objects. Each Ship defines the defined_for for zero, one, or many Lightship_definition objects.

4.3.97 Loading_condition_definition to Cargo_assignment

Each Loading_condition_definition has cargo_loads defined by one or many Cargo_assignment objects. Each Cargo_assignment defines the cargo_loads for zero, one, or many Loading_condition_-definition objects.

4.3.98 Loading_condition_definition to Deadweight

Each Loading_condition_definition has deadweight defined by zero or one Deadweight objects. Each Deadweight defines the deadweight for zero, one, or many Loading_condition_definition objects.

4.3.99 Loading_condition_definition to Floating_position

Each Loading_condition_definition has floating_position defined by exactly one Floating_position object. Each Floating_position defines the floating_position for zero, one, or many Loading_-condition_definition objects.

4.3.100 Loading_condition_definition to Ship

Each Loading_condition_definition has defined_for defined by one or many Ship objects. Each Ship defines the defined_for for zero, one, or many Loading_condition_definition objects.

4.3.101 Local_co_ordinate_system to Global_axis_placement

Each Local_co_ordinate_system has parent defined by exactly one Global_axis_placement object. Each Global_axis_placement defines the parent for zero, one, or many Local_co_ordinate_system objects.

4.3.102 Local_co_ordinate_system to Local_co_ordinate_system

Each Local_co_ordinate_system has parent defined by exactly one Local_co_ordinate_system object. Each Local_co_ordinate_system defines the parent for zero, one, or many Local_co_ordinate_system objects.

4.3.103 Local_co_ordinate_system_with_position_reference to Spacing_position

Each Local_co_ordinate_system_with_position_reference has longitudinal_ref defined by zero or one Spacing_position objects. Each Spacing_position defines the longitudinal_ref for zero, one, or many Local_co_ordinate_system_with_position_reference objects.

Each Local_co_ordinate_system_with_position_reference has transversal_ref defined by zero or one Spacing_position objects. Each Spacing_position defines the transversal_ref for zero, one, or many Local_co_ordinate_system_with_position_reference objects.

Each Local_co_ordinate_system_with_position_reference has vertical_ref defined by zero or one Spacing_position objects. Each Spacing_position defines the vertical_ref for zero, one, or many Local_co_ordinate_system_with_position_reference objects.

4.3.104 Longitudinal_table to Longitudinal_position

Each Longitudinal_table has spacing_table_representations defined by zero, one, or many Longitudinal_position objects. Each list of Longitudinal_position defines the spacing_table_-representations for zero, one, or many Longitudinal_table objects.

4.3.105 Moment_3d to Centre_location

Each Moment_3d has origin defined by exactly one Centre_location object. Each Centre_location defines the origin for zero, one, or many Moment_3d objects.

4.3.106 Regulation to External_reference

Each Regulation has international_regulations defined by zero, one, or many External_reference objects. Each External_reference defines the international_regulations for zero, one, or many Regulation objects.

Each Regulation has national_regulations defined by zero, one, or many External_reference objects. Each External_reference defines the national_regulations for zero, one, or many Regulation objects.

Each Regulation has standards defined by zero, one, or many External_reference objects. Each External_reference defines the standards for zero, one, or many Regulation objects.

4.3.107 Relative_cargo_position to Longitudinal_position

Each Relative_cargo_position has longitudinal_location defined by exactly one Longitudinal_-position object. Each Longitudinal_position defines the longitudinal_location for zero, one, or many Relative_cargo_position objects.

4.3.108 Relative_cargo_position to Transversal_position

Each Relative_cargo_position has transverse_location defined by exactly one Transversal_position object. Each Transversal_position defines the transverse_location for zero, one, or many Relative_-cargo_position objects.

4.3.109 Relative_cargo_position to Vertical_position

Each Relative_cargo_position has vertical_location defined by exactly one Vertical_position object. Each Vertical_position defines the vertical_location for zero, one, or many Relative_cargo_position objects.

4.3.110 Revision to Versionable_object

Each Revision has members defined by one or many Versionable_object objects. Each Versionable_-object defines the members for zero, one, or many Revision objects.

4.3.111 Revision_with_context to Definable_object

Each Revision_with_context has context_of_revision defined by exactly one Definable_object object. Each Definable_object defines the context_of_revision for zero, one, or many Revision_with_context objects.

4.3.112 Ship to Derived_unit

Each Ship has units defined by zero, one, or many Derived_unit objects. Each Derived_unit defines the units for zero, one, or many Ship objects.

4.3.113 Ship to Item

Each Ship has ship_items defined by zero, one, or many Item objects. Each Item defines the ship_items for zero, one, or many Ship objects.

4.3.114 Ship to Named_unit

Each Ship has units defined by zero, one, or many Named_unit objects. Each Named_unit defines the units for zero, one, or many Ship objects.

4.3.115 Shiptype to Ship

Each Shiptype has defined_for defined by one or many Ship objects. Each Ship defines the defined_for for zero, one, or many Shiptype objects.

4.3.116 Space_adjacency_relationship to Adjacent_space_surface_area

Each Space_adjacency_relationship has adjacent_space_surface_area defined by exactly one Adjacent_space_surface_area object. Each Adjacent_space_surface_area defines the adjacent_-space_surface_area for zero, one, or many Space_adjacency_relationship objects.

4.3.117 Space_arrangement_relationship to Space

Each Space_arrangement_relationship has item_1 defined by exactly one Space object. Each Space defines the item_1 for zero, one, or many Space_arrangement_relationship objects.

Each Space_arrangement_relationship has item_2 defined by exactly one Space object. Each Space defines the item_2 for zero, one, or many Space_arrangement_relationship objects.

4.3.118 Space_connection_relationship to External_instance_reference

Each Space_connection_relationship has connecting_system defined by zero or one External_-instance_reference objects. Each External_instance_reference defines the connecting_system for zero, one, or many Space_connection_relationship objects.

4.3.119 Space_enclosing_relationship to Space

Each Space_enclosing_relationship has item_1 defined by exactly one Space object. Each Space defines the item_1 for zero, one, or many Space_enclosing_relationship objects.

Each Space_enclosing_relationship has item_2 defined by exactly one Space object. Each Space defines the item_2 for zero, one, or many Space_enclosing_relationship objects.

4.3.120 Space_product_structure to Space

Each Space_product_structure has contained_in defined by exactly one Space object. Each Space defines the contained_in for zero, one, or many Space_product_structure objects.

4.3.121 Space_product_structure_revision to Design_definition

Each Space_product_structure_revision has members defined by one or many Design_definition objects. Each Design_definition defines the members for zero, one, or many Space_product_-structure_revision objects.

4.3.122 Space_product_structure_revision to Space_product_structure

Each Space_product_structure_revision has context_of_revision defined by exactly one Space_-product_structure object. Each Space_product_structure defines the context_of_revision for zero, one, or many Space_product_structure_revision objects.

4.3.123 Spacing_position_with_offset to Spacing_position

Each Spacing_position_with_offset has relating_spacing_position defined by exactly one Spacing_-position object. Each Spacing_position defines the relating_spacing_position for zero, one, or many Spacing_position_with_offset objects.

4.3.124 Spacing_table to Spacing_position

Each Spacing_table has spacing_table_representations defined by zero, one, or many Spacing_-position objects. Each list of Spacing_position defines the spacing_table_representations for zero, one, or many Spacing_table objects.

4.3.125 Stability_definition to Ship

Each Stability_definition has defined_for defined by one or many Ship objects. Each Ship defines the defined_for for zero, one, or many Stability_definition objects.

4.3.126 Stability_definition to Stability_table

Each Stability_definition has representations defined by one or many Stability_table objects. Each Stability_table defines the representations for zero, one, or many Stability_definition objects.

4.3.127 Stability_properties_for_one_floating_position to Centre_location

Each Stability_properties_for_one_floating_position has centre_of_gravity_above_keel defined by exactly one Centre_location object. Each Centre_location defines the centre_of_gravity_above_keel for zero, one, or many Stability_properties_for_one_floating_position objects.

4.3.128 Stability_properties_for_one_floating_position to Floating_position

Each Stability_properties_for_one_floating_position has definition_of_starting_floating_position defined by exactly one Floating_position object. Each Floating_position defines the definition_of_-starting_floating_position for zero, one, or many Stability_properties_for_one_floating_position objects.

4.3.129 Stability_properties_for_one_floating_position to Stability_property

Each Stability_properties_for_one_floating_position has stability_properties_for_different_angles_-of_heel defined by one or many Stability_property objects. Each list of Stability_property defines the stability_properties_for_different_angles_of_heel for zero, one, or many Stability_properties_-for_one_floating_position objects.

4.3.130 Stability_properties_for_one_floating_position to Stability_table

Each Stability_properties_for_one_floating_position has related_stability_table defined by one or many Stability_table objects. Each Stability_table defines the related_stability_table for zero, one, or many Stability_properties_for_one_floating_position objects.

4.3.131 Stability_property to Centre_location

Each Stability_property has centre_of_buoyancy defined by exactly one Centre_location object. Each Centre_location defines the centre_of_buoyancy for zero, one, or many Stability_property objects.

4.3.132 Stability_table to Stability_properties_for_one_floating_position

Each Stability_table has items defined by one or many Stability_properties_for_one_floating_-position objects. Each Stability_properties_for_one_floating_position defines the items for zero, one, or many Stability_table objects.

4.3.133 Tank_compartment_property to Capacity_properties

Each Tank_compartment_property has liquid_capacity defined by zero, one, or many Capacity_-properties objects. Each Capacity_properties defines the liquid_capacity for zero, one, or many Tank_compartment_property objects.

4.3.134 Tank_compartment_property to Moments_of_inertia

Each Tank_compartment_property has moments_of_inertia defined by exactly one Moments_of_-inertia object. Each Moments_of_inertia defines the moments_of_inertia for zero, one, or many Tank_compartment_property objects.

4.3.135 Tank_compartment_property to Tank_geometric_parameters

Each Tank_compartment_property has geometric_parameters defined by zero or one Tank_-geometric_parameters objects. Each Tank_geometric_parameters defines the geometric_parameters for zero, one, or many Tank_compartment_property objects.

4.3.136 Tank_compartment_property to Tank_piping_design_properties

Each Tank_compartment_property has design_properties defined by zero or one Tank_piping_-design_properties objects. Each Tank_piping_design_properties defines the design_properties for zero, one, or many Tank_compartment_property objects.

4.3.137 Tonnage_definition to Compartment

Each Tonnage_definition has spaces_excluded defined by zero, one, or many Compartment objects. Each Compartment defines the spaces_excluded for zero, one, or many Tonnage_definition objects.

4.3.138 Tonnage_definition to Compensated_gross_tonnage

Each Tonnage_definition has compensated_gross_tonnage defined by exactly one Compensated_-gross_tonnage object. Each Compensated_gross_tonnage defines the compensated_gross_tonnage for zero, one, or many Tonnage_definition objects.

4.3.139 Tonnage_definition to Document_reference_with_address

Each Tonnage_definition has certificate defined by exactly one Document_reference_with_address object. Each Document_reference_with_address defines the certificate for zero, one, or many Tonnage_definition objects.

4.3.140 Tonnage_definition to Gross_tonnage

Each Tonnage_definition has gross_tonnage defined by exactly one Gross_tonnage object. Each Gross_tonnage defines the gross_tonnage for zero, one, or many Tonnage_definition objects.

4.3.141 Tonnage_definition to Net_tonnage

Each Tonnage_definition has net_tonnage defined by exactly one Net_tonnage object. Each Net_-tonnage defines the net_tonnage for zero, one, or many Tonnage_definition objects.

4.3.142 Tonnage_definition to Ship

Each Tonnage_definition is defined for one or many Ship objects. Each Ship has zero, one, or many Tonnage_definition objects.

4.3.143 Tonnage_measurement to Compartment_group

Each Tonnage_measurement has spaces_included defined by zero, one, or many Compartment_group objects. Each Compartment_group defines the spaces_included for zero, one, or many Tonnage_-measurement objects.

4.3.144 Transversal_table to Transversal_position

Each Transversal_table has spacing_table_representations defined by zero, one, or many Transversal_position objects. Each list of Transversal_position defines the spacing_table_-representations for zero, one, or many Transversal_table objects.

4.3.145 Unit_cargo to Cargo_footprint

Each Unit_cargo has footprints defined by zero, one, or many Cargo_footprint objects. Each Cargo_footprint defines the footprints for zero, one, or many Unit_cargo objects.

4.3.146 Unit_cargo to Non_manifold_surface_shape

Each Unit_cargo has bounding_space defined by zero or one Non_manifold_surface_shape objects. Each Non_manifold_surface_shape defines the bounding_space for zero, one, or many Unit_cargo objects.

Each Unit_cargo has shape_description defined by zero or one Non_manifold_surface_shape objects. Each Non_manifold_surface_shape defines the shape for zero, one, or many Unit_cargo objects.

4.3.147 Unit_cargo to Unit_cargo_bounding_box

Each Unit_cargo has bounding_space defined by zero or one Unit_cargo_bounding_box objects. Each Unit_cargo_bounding_box defines the bounding_space for zero, one, or many Unit_cargo objects.

4.3.148 Unit_cargo to Weight_and_centre_of_gravity

Each Unit_cargo has weight_and_centre_of_gravity defined by zero or one Weight_and_centre_-of_gravity objects. Each Weight_and_centre_of_gravity defines the weight_and_centre_of_gravity for zero, one, or many Unit_cargo objects.

4.3.149 Unit_cargo_assignment to Cargo_position

Each Unit_cargo_assignment has position defined by exactly one Cargo_position object. Each Cargo_position defines the position for zero, one, or many Unit_cargo_assignment objects.

4.3.150 Unit_cargo_assignment to Compartment

Each Unit_cargo_assignment has assigned_to defined by zero or one Compartment object. Each Compartment defines the assigned_to for zero, one, or many Unit_cargo_assignment objects.

4.3.151 Unit_cargo_assignment to Deck_zone

Each Unit_cargo_assignment has assigned_to defined by zero or one Deck_zone object. Each Deck_zone defines the assigned_to for zero, one, or many Unit_cargo_assignment objects.

4.3.152 Unit_cargo_group to Cargo_footprint

Each Unit_cargo_group has footprint defined by exactly one Cargo_footprint object. Each Cargo_footprint defines the footprint for zero, one, or many Unit_cargo_group objects.

4.3.153 Unit_cargo_group to Unit_cargo

Each Unit_cargo_group has cargo defined by one, or many Unit_cargo objects. Each Unit_cargo defines the cargo for zero, one, or many Unit_cargo_group objects.

4.3.154 Unit_cargo_group to Weight_and_centre_of_gravity

Each Unit_cargo_group has weight_and_centre_of_gravity defined by exactly one Weight_and_-centre_of_gravity object. Each Weight_and_centre_of_gravity defines the weight_and_centre_-of_gravity for zero, one, or many Unit_cargo_group objects.

4.3.155 Version_creation to Versionable_object

Each Version_creation has base defined by zero, one, or many Versionable_object objects. Each Versionable_object defines the base for zero, one, or many Version_creation objects.

Each Version_creation has subject defined by exactly one Versionable_object object. Each Versionable_object defines the subject for zero, one, or many Version_creation objects.

4.3.156 Version_deletion to Versionable_object

Each Version_deletion has subject defined by exactly one Versionable_object object. Each Versionable_object defines the subject for zero, one, or many Version_deletion objects.

4.3.157 Version_history to Version_relationship

Each Version_history has relationships defined by zero, one, or many Version_relationship objects. Each Version_relationship defines the relationships for zero, one, or many Version_history objects.

4.3.158 Version_history to Versionable_object

Each Version_history has versions defined by one or many Versionable_object objects. Each Versionable_object defines the versions for zero, one, or many Version_history objects.

Each Version_history has current_version defined by exactly one Versionable_object object. Each Versionable_object defines the current_version for zero, one, or many Version_history objects.

4.3.159 Version_modification to Versionable_object

Each Version_modification has base defined by one or many Versionable_object objects. Each Versionable_object defines the base for zero, one, or many Version_modification objects.

Each Version_modification has subject defined by exactly one Versionable_object object. Each Versionable_object defines the subject for zero, one, or many Version_modification objects.

4.3.160 Version_relationship to Versionable_object

Each Version_relationship has predecessor defined by exactly one Versionable_object object. Each Versionable_object defines the predecessor for zero, one, or many Version_relationship objects.

Each Version_relationship has successor defined by exactly one Versionable_object object. Each Versionable_object defines the successor for zero, one, or many Version_relationship objects.

4.3.161 Vertical_table to Vertical_position

Each Vertical_table has spacing_table_representations defined by zero, one, or many Vertical_position objects. Each list of Vertical_position defines the spacing_table_representations for zero, one, or many Vertical_table objects.

4.3.162 Weight_and_centre_of_gravity to Centre_location

Each Weight_and_centre_of_gravity has center_of_gravity defined by exactly one Centre_location object. Each Centre_location defines the centre of gravity for zero, one, or many Weight_and_-centre_of_gravity objects.

4.3.163 Weight_and_centre_of_gravity to Moment_3d

Each Weight_and_centre_of_gravity has moment defined by zero or one Moment_3d objects. Each Moment_3d defines the moment for zero, one, or many Weight_and_centre_of_gravity objects.

4.3.164 Zone_design_definition to Compartment

Each Zone_design_definition has constituent_compartments defined by zero, one, or many Compartment objects. Each Compartment defines the constituent_compartments for zero, one, or many Zone_design_definition objects.

4.3.165 Zone_design_definition to Compartment_property

Each Zone_design_definition has properties defined by zero, one, or many Compartment_property objects. Each Compartment_property defines the properties for zero, one, or many Zone_design_-definition objects.

4.3.166 Zone_design_definition to External_instance_reference

Each Zone_design_definition has boundaries defined by zero, one, or many External_instance_-reference objects. Each External_instance_reference defines the boundaries for zero, one, or many Zone_design_definition objects.

Each Zone_design_definition has constituent_compartments defined by zero, one, or many External_-instance_reference objects. Each External_instance_reference defines the constituent_compartments for zero, one, or many Zone_design_definition objects.

4.3.167 Zone_design_definition to Non_manifold_surface_shape

Each Zone_design_definition has representations defined by zero, one, or many Non_manifold_-surface_shape objects. Each Non_manifold_surface_shape defines the representations for zero, one, or many Zone_design_definition objects.

4.3.168 Zone_design_definition to Zone

Each Zone_design_definition has defined_for defined by one or many Zone objects. Each Zone defines the defined_for for zero, one, or many Zone_design_definition objects.

4.3.169 Zone_functional_definition to Zone

Each Zone_functional_definition has defined_for defined by one or many Zone objects. Each Zone defines the defined_for for zero, one, or many Zone_functional_definition objects.

5 Application interpreted model

5.1 Mapping specification

This clause contains the mapping specification that shows how each UoF and application object of this part of ISO 10303 (see clause 4) maps to one or more AIM constructs (see annex A). Each mapping specifies up to five elements.

Application element: The mapping for each application element is specified in a separate subclause below. Application object names are given in title case. Attribute names and assertions are listed after the application object to which they belong and are given in lower case.

AIM element: The name of one or more AIM entity data types (see annex A), the term "IDENTICAL MAPPING", or the term "PATH". AIM entity data type names are given in lower case. Attributes of AIM entity data types are referred to as <entity name>.<attribute name>. The mapping of an application element may involve more than one AIM element. Each of these AIM elements is presented on a separate line in the mapping specification. The term "IDENTICAL MAPPING" indicates that both application objects involved in an application assertion map to the same instance of an AIM entity data type. The term "PATH" indicates that the application assertion maps to a collection of related AIM entity instances specified by the entire reference path.

Source: For those AIM elements that are interpreted from any common resource, this is the ISO standard number and part number in which the resource is defined. For those AIM elements that are created for the purpose of this part of ISO 10303, this is "ISO 10303-" followed by the number of this part.

Rules: One or more global rules may be specified that apply to the population of the AIM entity data types specified as the AIM element or in the reference path. For rules that are derived from relationships between application objects, the same rule is referred to by the mapping entries of all the involved AIM elements. A reference to a global rule may be accompanied by a reference to the subclause in which the rule is defined.

Reference path: To describe fully the mapping of an application object, it may be necessary to specify a reference path involving several related AIM elements. Each line in the reference path documents the role of an AIM element relative to the AIM element in the line following it. Two or more such related AIM elements define the interpretation of the integrated resources that satisfies the requirement specified by the application object. For each AIM element that has been created for use within this part of ISO 10303, a reference path to its supertype from an integrated resource is specified. For the expression of reference paths and the relationships between AIM elements the following notational conventions apply:

[] enclosed section constrains multiple AIM elements or sections of the reference path are required to satisfy an information requirement;

() enclosed section constrains multiple AIM elements or sections of the reference path are identified as alternatives within the mapping to satisfy an information requirement;

{} enclosed section constrains the reference path to satisfy an information requirement;

<> enclosed section constrains at one or more required reference path;

|| enclosed section constrains the supertype entity;

-> attribute references the entity or select type given in the following row;

<- entity or select type is referenced by the attribute in the following row;

[i] attribute is an aggregation of which a single member is given in the following row;

[n] attribute is an aggregation of which member n is given in the following row;

=> entity is a supertype of the entity given in the following row;

<= entity is a subtype of the entity given in the following row;

= the string, select, or enumeration type is constrained to a choice or value;

\ the reference path expression continues on the next line;

* used in conjunction with braces to indicate that any number of relationship entity data types may be assembled in a relationship tree structure

// enclosed section is an application of one of the mapping templates defined in clause 5.1.1;

-- the text following is a comment (normally a clause reference).

For the purposes of defining mapping templates, the following abbreviations apply:

ACT	action
ASSGN	assignment
CART	cartesian
CAT	category
CD	context_dependent
DEF	definition
DESC	description
DO	definable_object
DOC	document
EXT	external
FUNC	function
GEO	geometry
INST	instance
ORG	organization
PD	product_definition
PDCD	product_definition to characterized_definition
PDR	product_definition_representation
PERS	person
PROD	product
PROP	property
REF	reference
REL	relationship
REP	representation
SA	shape_aspect
SAR	shape_aspect_relationship
SDR	shape_definition_representation
SRC	source
VAL	value
VO	versionable_object

5.1.1 Mapping templates

This mapping specification includes mapping templates. A mapping template is a reusable portion of a reference path that defines a commonly used part of the structure of the application interpreted model. A mapping template is similar to a programming language macro. The mapping templates used in this part of ISO 10303 are defined in this subclause. Each mapping template definition has three components, as follows:

— the mapping signature that specifies the name of the template and may also specify the names and the order of the formal parameters of the template;

— descriptions of the formal parameters of the template, if any;

— the template body that defines the reusable portion of a reference path and may indicate, through the use of the formal parameter names included in the mapping signature, the points at which the value parameters are supplied in each template application.

Each mapping template is used at least once in the reference paths specified in 5.1.2.1 to 5.1.22.2. Each such template application is a reference to the template definition, based on the pattern established by the mapping signature, and supplies the value parameters that are to substituted for the formal parameters specified in the template definition. The full reference path can be derived by replacing any formal parameters in the template body by the value parameters specified in the template application and then substituting the completed template body for the template application.

The non-blank characters following the first "/" define the name of the mapping template. The name of the mapping template is given in upper case. The name of the template is followed by a list of value parameters, separated by commas, enclosed in parentheses. Parameter values are given in lower case except in the case that the value parameter is a string literal that includes upper case characters.

The following notational conventions apply to the definitions and applications of templates:

/ marks the beginning and end of a mapping signature or a template application;

& prefixes the name of a formal parameter within the definition of a template body;

( ) enclose the formal parameters in a mapping signature or the value parameters in a template application;

, separates formal parameters in a mapping signature or value parameters in a template application;

' ' denotes a string literal that is used as a value parameter in a template application.

Value parameters that are not enclosed by quotes are EXPRESS data type identifiers.

This part of ISO 10303 uses the templates that are specified in the following subclauses.

5.1.1.1 APPROVES

The APPROVES mapping template specifies a reference path constraint in which instances of type ENTITY are the **approval_item**s within instances of type **applied_approval_assignment**, where the role name specified for assignment is ARM_ROLE.

Mapping signature:

/APPROVES(ENTITY, ARM_ROLE)/

Parameter definitions:

ARM_ROLE: the value of the name attribute for object_role.

ENTITY: the identifier of the AIM entity data type to which an approval is assigned

Template body:

```
approval_assignment.assigned_approval
approval_assignment =>
{/ROLE_ASSGN(approval_assignment)/
[object_role.name = &ARM_ROLE]}
applied_approval_assignment
applied_approval_assignment.items[i] ->
approval_item
approval_item = &ENTITY
```

5.1.1.2 CLASS

The CLASS mapping template specifies a reference path constraint in which instances of type T are the **classification_item**s within instances of **applied_classification_assignment**, where the role name specified for the assignment is 'class membership', and the assigned classification is a **group** whose name attribute is assigned the value ID. The **group** instance is also related to a parent **group** (via a **group_relationship** instance) whose name attribute is assigned the value S_ID.

Mapping signature:

/CLASS(T, ID, S_ID)/

Parameter definitions:

T: type of the instance that is classified.

ID: group name of the instance that is classified.

S_ID: group name of the parent

Template body:

```
&T
classification_item = &T
classification_item <-
applied_classification_assignment.items[i]
applied_classification_assignment <=
classification_assignment
{classification_assignment.role ->
classification_role
classification_role.name = 'class membership'}
classification_assignment
classification_assignment.assigned_class ->
group
{[group.name = &ID]
[group <-
group_relationship.related_group
group_relationship
{group_relationship.name = 'specialisation'}
group_relationship
group_relationship.relating_group ->
group
{group.name = &S_ID}]}
```



```
group =>
class
```

5.1.1.3 CLASS_HELP

The CLASS_HELP mapping template specifies a reference path constraint in which instances of type T are the **classification_item**s within instances of **applied_classification_assignment**, where the role name specified for the assignment is ‘class membership’, and the assigned classification is a supertype of **group**.

Mapping signature:

CLASS_HELP(T)

Parameter definitions:

T: type of the instance that is classified

Template body:

```
{&T
classification_item = &T
classification_item <-
applied_classification_assignment.items[i]
applied_classification_assignment <=
classification_assignment
{classification_assignment.role ->
classification_role
classification_role.name = 'class membership'}
classification_assignment.assigned_class ->
group =>
```

5.1.1.4 CLASS_ID

The CLASS_ID mapping template specifies a reference path constraint in which instances of type T are the **classification_item**s within instances of **applied_classification_assignment**, where the role name specified for the assignment is ‘class membership’, and the assigned classification is a **group** whose name attribute has the value ID.

NOTE - The semantics of the template is to select an instance with a group name of ID

Mapping signature:

/CLASS_ID(T, ID)/

Parameter definitions:

T: type of the instance that is classified

ID: group name

Template body:

```
&T
classification_item = &T
classification_item <-
applied_classification_assignment.items[i]
applied_classification_assignment <=
classification_assignment
{classification_assignment.role ->
```

```
classification_role
classification_role.name = 'class membership'}
classification_assignment.assigned_class ->
group =>
{group.name = &ID}
class
```

5.1.1.5 COMPOUND

The COMPOUND mapping template specifies a reference path constraint in which a path from a **compound_representation_item** to **representation_item** assigns a value to the **representation_item.name** attribute.

Mapping signature:

/COMPOUND(NAME)/

Parameter definitions:

NAME: The value for the name attribute for **representation_item**

Template body:

```
compound_representation_item
compound_representation_item.item_element ->
compound_item_definition = list_representation_item
list_representation_item[i] ->
representation_item =>
{representation_item.name = &NAME}
```

5.1.1.6 DAT_TIME_ASSGN

The DAT_TIME_ASSGN mapping template specifies a reference path constraint in which an instance of type T is assigned a **date_and_time**, and the assignment has the role ROLE.

Mapping signature:

/DAT_TIME_ASSGN(T, ROLE)/

Parameter definitions:

T: type of the instance that has a **date_and_time** assignment

ROLE: the value of the name attribute for **date_time_role**.

Template body:

```
&T
{date_and_time_item = &T}
date_and_time_item <-
applied_date_and_time_assignment.items[i]
applied_date_and_time_assignment <=
date_and_time_assignment
{date_and_time_assignment.role ->
date_time_role
[date_time_role.name = &ROLE]}
date_and_time_assignment.assigned_date_and_time ->
date_and_time
```


5.1.1.7 DESCRIPTION_ASSGN

The DESCRIPTION_ASSGN mapping template specifies a reference path constraint in which the assignment of a **description_attribute** to an instance of an entity that contains a derived description attribute and uses the Basic_attribute_schema of ISO 10303-41.

NOTE The following entity data types use the DESCRIPTION_ASSGN for population of their description attribute:
— action_request_solution;
— application_context;
— approval_role;
— configuration_design;
— date_role;
— date_time_role;
— context_dependent_shape_representation;
— effectivity;
— external_source;
— organization_role;
— person_and_organization_role;
— person_and_organization;
— person_role;
— property_definition_representation;
— representation;
— time_role.

Mapping signature:

/DESCRIPTION_ASSGN(ENTITY)/

Parameter definitions:

ENTITY: the entity type to which the **description_attribute** is assigned

Template body:

```
&ENTITY
description_attribute_select = &ENTITY
description_attribute_select <-
description_attribute.described_item
description_attribute
description_attribute.attribute_value
```

5.1.1.8 DESCRIPTION_ASSGN_WITH_VAL

The DESCRIPTION_ASSGN_WITH_VAL mapping template specifies a reference path constraint in which a **description_attribute** is assigned to an instance of ENTITY that contains a derived description attribute, and the value of the description attribute is specified by DESC_VALUE.

NOTE The following Entity data types use of the DESCRIPTION_ASSGN_WITH_VAL for population of their description attribute:
— action_request_solution;
— application_context;
— approval_role;
— configuration_design;
— date_role;
— date_time_role;
— context_dependent_shape_representation;

— effectivity;
— external_source;
— organization_role;
— person_and_organization_role;
— person_and_organization;
— person_role;
— property_definition_representation;
— representation;
— time_role.

Mapping signature:

/DESCRIPTION_ASSGN_WITH_VAL(ENTITY, DESC_VALUE)/

Parameter definitions:

ENTITY: the entity type to which the description_attribute is assigned

DESC_VALUE: the value assigned to **description_attribute.attribute_value**

Template body:

```
&ENTITY
description_attribute_select = &ENTITY
description_attribute_select <-
description_attribute.described_item
description_attribute
description_attribute.attribute_value
{description_attribute.attribute_value = &DESC_VALUE}
```

5.1.1.9 DOC_REF

The DOC_REF mapping template specifies a reference path constraint in which an instance of type T references a **document** or **document_usage_constraint** that plays the role ID.

Mapping signature:

/DOC_REF(T, ID)/

Parameter definitions:

T: type of the instance that references a **document**

ID: the value for the name attribute for **object_role**

Template body:

```
&T
document_reference_item = &T
document_reference_item <-
applied_document_reference.items[i]
applied_document_reference <=
document_reference
{ROLE_ASSGN(document_reference)
[object_role.name = &ID]}
document_reference
document_reference.assigned_document ->
(document)
```



```
(document <-
document_usage_constraint.source
document_usage_constraint)
```

5.1.1.10 EXT_INST_REF

The EXT_INST_REF mapping template specifies a reference path constraint in which an entity of type LT is assigned an **external_identification_assignment** in the role of 'external instance reference'. The **external_identification_assignment** refers to an **external_source** entity whose source_id attribute is assgined the value SN. This **external_source** entity will be related to another **external_source** entity whose source_id attribute is assgined the value ET.

Mapping signature:

/EXT_INST_REF(LT, SN, ET)/

Parameter definitions:

LT: type whose instance represents the external instance in the local instance model

SN: external schema name

ET: external entity type

Template body:

```
[&LT.description = 'external instance reference target']
[&LT = external_identification_item
external_identification_item <-
applied_external_identification_assignment.items[i]
applied_external_identification_assignment <=
external_identification_assignment <=
/ID_ROLE('external instance reference')/
external_identification_assignment.source ->
external_source
[external_source.source_id->
source_item = identifier
{identifier = &SN}]
[{/DESCRIPTION_ASSGN_WITH_VAL(external_source,'schema name')/}]
[/EXT_SRC_REL('entity type')/
{identifier = &ET}]]
```

5.1.1.11 EXT_SRC_REL

The EXT_SRC_REL mapping template specifies a reference path constraint in which an **external_source** entity is related to another **external_source** entity, and the latter entity will have a description of DESCR.

NOTE The concept can be used mainly in two places:
— for relating different components of an external_source to a single instance on the next higher level;
— for mapping of External_instance_reference.

Mapping signature:

/EXT_SRC_REL(DESCR)/

Parameter definitions:

DESCR: The value of the **description** attribute for the **external_source**.

Template body:

```
external_source <-
external_source_relationship.relating_source
external_source_relationship
{external_source_relationship.name = 'composition'}
external_source_relationship.related_source
external_source
{/DESCRIPTION_ASSGN_WITH_VAL(external_source, &DESCR)/}
external_source
external_source.source_id ->
source_item
source_item = identifier
```

5.1.1.12 GEO_REP_ITEM

The GEO_REP_ITEM specifies a geometric_representation_item with an attribute name.

Mapping signature:

/GEO_REP_ITEM(ID, GEO)/

Parameter definitions:

ID: The value for the **name** attribute for **representation_item**.

GEO: Type of **geometric_representation_item**.

Template body:

```
representation_item =>
{representation_item.name = &ID}
geometric_representation_item =>
&GEO
```

5.1.1.13 GROUPS

The GROUPS mapping template specifies a reference path constraint in which instances of type ENTITY are the **group_item**s within an **applied_group_assignment**, and the role of the assignment is ARM_ROLE.

Mapping signature:

/GROUPS(ENTITY, ARM_ROLE)/

Parameter definitions:

ARM_ROLE: the value of the **name** attribute for **object_role**

ENTITY: the entities being grouped that play an ARM_ROLE

Template body:

```
group_assignment.assigned_group
group_assignment =>
{/ROLE_ASSGN(group_assignment)/
[object_role.name = &ARM_ROLE]}
applied_group_assignment
applied_group_assignment.items[i] ->
```



```
group_item
group_item = &ENTITY
```

5.1.1.14 HAS_ID_1_ROLE

The HAS_ID_1_ROLE mapping template specifies a reference path constraint in which an instance of entity type T is given an identifier that plays a single role ROLE.

Mapping signature:

/HAS_ID_1_ROLE(T, ROLE)/

Parameter definitions:

T: type of the instance that is assigned an **identification_assignment**

ROLE: the value of the **name** attribute for **identification_role**

Template body:

```
/IDENTIFICATION(&T)/
{identification_role.name = &ROLE}
identification_assignment
identification_assignment.assigned_id
```

5.1.1.15 HAS_ID_2_ROLES

The HAS_ID_2_ROLES mapping template specifies a reference path constraint in which an instance of entity type T is given an identifier and plays one of two specified roles.

Mapping signature:

/HAS_ID_2_ROLES(T, ROLE1, ROLE2)/

Parameter definitions:

T: type of the instance that is assigned an **identification_assignment**

ROLE1: the value of the **name** attribute for **identification_role**

ROLE2: the value of the **name** attribute for **identification_role**

Template body:

```
/IDENTIFICATION(&T)/
{(identification_role.name = &ROLE1)
(identification_role.name = &ROLE2)}
identification_assignment
identification_assignment.assigned_id
```

5.1.1.16 IDENTIFICATION

The IDENTIFICATION mapping template specifies a reference path constraint in which an entity of type T is assigned an **identification_assignment**. This mapping template is primarily used in other macros.

Mapping signature:

/IDENTIFICATION(T)/

Parameter definitions:

T: type of the instance that is assigned an **identification_assignment**

Template body:

```
&T
identification_item = &T
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment <=
identification_assignment
identification_assignment.role ->
identification_role
```

5.1.1.17 ID_ASSGN

The ID_ASSGN mapping template specifies a reference path constraint in which an **id_attribute** is assigned to an instance of an ENTITY that contains a derived id attribute. The derivation of the id attribute is based on the Basic_attribute_schema of ISO 10303-41.

NOTE The following entity data types use ID_ASSGN for population of their id attribute:
— action;
— address;
— product_category;
— property_definition;
— shape_aspect;
— shape_aspect_relationship;
— application_context;
— group;
— organizational_project;
— representation.

Mapping signature:

/ID_ASSGN(ENTITY)/

Parameter definitions:

ENTITY: the entity type to which the **id_attribute** is assigned

Template body:

```
&ENTITY
id_attribute_select = &ENTITY
id_attribute_select <-
id_attribute.identified_item
id_attribute
id_attribute.attribute_value
```

5.1.1.18 ID_ROLE

The ID_ROLE mapping template specifies a reference path constraint in which an assignment of an **identification_assignment** has the role ROLE.

Mapping signature:

/ID_ROLE(ROLE)/

Parameter definitions:

ROLE: the value of the name attribute for **identification_role**

Template body:

```
identification_assignment
{identification_assignment.role ->
identificaton_role
identification_role.name = &ROLE}
```

5.1.1.19 LINK_TO_GROUP

The LINK_TO_GROUP mapping template specifies a reference path constraint in which an instance of type T is linked to a **group** instance via an **applied_group_assignment**.

NOTE Used for all application object types that are subtypes of both Item (see 4.2.109) and Item_structure (see 4.2.111).

Mapping signature:

/LINK_TO_GROUP(T)/

Parameter definitions:

T: the instance to be grouped

Template body:

```
group_item = &T <-
applied_group_assignment.items[i]
applied_group_assignment <=
group_assignment
{/ROLE_ASSGN(group_assignment)/
object_role.name = 'equivalence'}
group_assignment
group_assignment.assigned_group ->
group
group.name = ‘item and item_structure’
```

5.1.1.20 NAME_ASSGN

The NAME_ASSGN mapping template specifies a reference path constraint in which a **name_attribute** is assigned to an instance of an ENTITY that contains a derived name attribute. The derivation of the name attribute is based on the Basic_attribute_schema of ISO 10303-41.

NOTE The following entity data types use NAME_ASSGN for population of their name attribute:
— action_request_solution;
— address;
— configuration_design;

— context_dependent_shape_representation;
— derived_unit;
— effectivity;
— person_and_organization;
— product_definition;
— product_definition_substitute;
— property_definition_representation.

Mapping signature:

/NAME_ASSGN(ENTITY)/

Parameter definitions:

ENTITY: the entity type to which the **name_attribute** is assigned

Template body:

```
&ENTITY
name_attribute_select = &ENTITY
name_attribute_select <-
name_attribute.named_item
name_attribute
name_attribute.attribute_value
```

5.1.1.21 NAME_ASSGN_WITH_VAL

The NAME_ASSGN_WITH_VAL mapping template specifies a reference path constraint in which a **name_attribute** is assigned to an instance of ENTITY that contains a derived name attribute, and the value of the name attribute is specified by NAME_VALUE.

NOTE The following entity data types use NAME_ASSGN for population of their name attribute:
— action_request_solution;
— address;
— configuration_design;
— context_dependent_shape_representation;
— derived_unit;
— effectivity;
— person_and_organization;
— product_definition;
— product_definition_substitute;
— property_definition_representation.

Mapping signature:

/NAME_ASSGN_WITH_VAL(ENTITY, NAME_VALUE)/

Parameter definitions:

ENTITY: the entity type to which the **name_attribute** is assigned

NAME_VALUE: the value assigned to **name_attribute.attribute_value**

Template body:

```
&ENTITY
name_attribute_select = &ENTITY
```



```
name_attribute_select <-
name_attribute.named_item
name_attribute
{name_attribute.attribute_value = &NAME_VALUE}
```

5.1.1.22 ORG_ASSGN_PART

The ORG_ASSGN_PART mapping template specifies a reference path constraint in which an instance of type T is assigned an **organization_assignment**.

Mapping signature:

/ORG_ASSGN_PART(T)/

Parameter definitions:

T: type of the instance that is assigned an **organization_assignment**.

Template body:

```
&T
organization_item = &T
organization_item <-
applied_organization_assignment.items [i]
applied_organization_assignment <=
organization_assignment
```

5.1.1.23 ORG_ASSGN

The ORG_ASSGN mapping template specifies a reference path constraint in which an entity of type T is assigned an **organization**, where the role of the assignment is ROLE.

Mapping signature:

/ORG_ASSGN(T, ROLE)/

Parameter definitions:

T: the entity assigned to **organization**

ROLE: the value of the **name** attribute for **organization_role**

Template body:

```
&T
organization_item = &T
organization_item <-
applied_organization_assignment.items[i]
applied_organization_assignment <=
organization_assignment
{organization_assignment.role ->
organization_role
organization_role.name = &ROLE}
organization_assignment
organization_assignment.assigned_organization ->
organization
```

5.1.1.24 PDR_NAME

The PDR_NAME mapping template specifies a reference path constraint in which a **property_definition_representation.name** is assigned the value NAME.

Mapping signature:

/PDR_NAME(NAME)/

Parameter definitions:

NAME: the value for the **name** attribute for **property_definition_representation**

Template body:

```
property_definition_representation.definition
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, &NAME)/}
property_definition_representation.used_representation
```

5.1.1.25 PERS_ASSGN

The PERS_ASSGN mapping template specifies a reference path constraint in which an entity of type T is assigned a person with the role ROLE.

Mapping signature:

/PERS_ASSGN(T, ROLE)/

Parameter definitions:

T: the entity assigned a **person**

ROLE: the value of the **name** attribute for **person_role**

Template body:

```
&T
person_item = &T
person_item <-
applied_person_assignment.items[i]
applied_person_assignment <=
person_assignment
{person_assignment.role ->
person_role
person_role.name = &ROLE}
person_assignment
person_assignment.assigned_person ->
person
```

5.1.1.26 PERS_ORG_ASSGN

The PERS_ORG_ASSGN mapping template specifies a reference path constraint in which an entity of type T is assigned a person and organization with the role ROLE.

Mapping signature:

/PERS_ORG_ASSGN(T, ROLE)/

Parameter definitions:

T: the entity assigned a **person_and_organization**

ROLE: the value of the **name** attribute for **person_and_organization_role**

Template body:

```
&T
person_and_organization_item = &T
person_and_organisation_item <-
applied_person_and_organization_assignment.items[i]
applied_person_and_organization_assignment <=
person_and_organization_assignment
{person_and_organization_assignment.role ->
person_and_organization_role
person_and_organization_role.name = &ROLE}
person_and_organization_assignment
person_and_organization_assignment.assigned_person_and_organization
person_and_organization
```

5.1.1.27 PROD_CAT_NAME

The PROD_CAT_NAME mapping template specifies a reference path constraint in which a **product_category.name** is assigned the value NAME.

Mapping signature:

/PROD_CAT_NAME(NAME)/

Parameter definitions:

NAME: value given the **name** attribute for **product_category**

Template body:

```
product_related_product_category <=
product_category
{product_category.name = &NAME}
```

5.1.1.28 PROD_DEF_PRODUCT

The PROD_DEF_PRODUCT mapping template specifies a reference path constraint in which a **product_definition** is linked with a **product**.

Mapping signature:

/PROD_DEF_PRODUCT/

Parameter definitions:

none

Template body:

```
product_definition
product_definition.formation ->
product_definition_formation
product_definition_formation.of_product ->
product
```

5.1.1.29 PROD_DEF_PROP_DEF

The PROD_DEF_PROP_DEF mapping template specifies a reference path constraint in which a **product_definition** is linked with a **property_definition**.

Mapping signature:

/PROD_DEF_PROP_DEF/

Parameter definitions:

none

Template body:

```
/PROD_DEF_PROP_DEF_HELP/
property_definition
```

5.1.1.30 PROD_DEF_PROP_DEF_HELP

The PROD_DEF_PROP_DEF_HELP mapping template specifies a reference path constraint in which a **product_definition** is linked with a **property_definition.definition**. PROD_DEF_PROP_-DEF_HELP is intended to be used by other mapping templates.

Mapping signature:

/PROD_DEF_PROP_DEF_HELP/

Parameter definitions:

none

Template body:

```
product_definition
characterized_product_definition = product_definition
characterized_product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
```

5.1.1.31 PROD_DEF_TO_DESC_REP_ITEM

The PROD_DEF_TO_DESC_REP_ITEM mapping template specifies a reference path constraint in which a **product_definition** is linked with a **descriptive_represenation_item**. The **descriptive_representation_item.name** will be assigned the value ID2. This link is through a **property_definition_representation** that has an attribute **name** which will be given the value ID1. PROD_DEF_TO_DESC_REP_ITEM is intended to be used by other mapping templates.

Mapping signature:

/PROD_DEF_TO_DESC_REP_ITEM(ID1, ID2)/

Parameter definitions:

ID1: the value of the **name** attribute for **property_definition_representation**

ID2: the value of the **name** attribute for **descriptive_representation_item**

Template body:

```
/PROD_DEF_PROP_DEF_HELP/
/PROP_DEF_TO_DESC_REP_ITEM(&ID1, &ID2)/
```

5.1.1.32 PROD_DEF_TO_REP

The PROD_DEF_TO_REP mapping template specifies a reference path constraint in which that links a **product_definition** is linked with a **property_definition_representation** with an attribute **name** that has the value ID. PROD_DEF_TO_REP is intended to be used by other mapping templates.

Mapping signature:

/PROD_DEF_TO_REP(ID)/

Parameter definitions:

ID: the value of the **name** attribute for **property_definition_representation**

Template body:

```
/PROD_DEF_PROP_DEF_HELP/
/PROP_DEF_REP_HELP(&ID)/
```

5.1.1.33 PROD_DEF_TO_SPECIAL_VAL_REP_ITEM

The PROD_DEF_TO_SPECIAL_VAL_REP_ITEM mapping template specifies a reference path constraint in which a **product_definition** is linked with a **value_representation_item** that refers to **measure_value** that is a **context_dependent_measure**. This link is through a **property_-definition_representation** whose **name** attribute will be assigned the value ID1. The name attribute of the **value_representation_item** will be assigned the value ID2. The units for the **context_dependent_measure** will be defined by a **derived_unit** that has a **name** attribute with the value DER_UNIT_NAME. PROD_DEF_TO_SPECIAL_VAL_REP_ITEM is intended to be used by other mapping templates.

Mapping signature:

/PROD_DEF_TO_SPECIAL_VAL_REP_ITEM(ID1, ID2, DER_UNIT_NAME)/

Parameter definitions:

ID1: the value of the **name** attribute for **property_definition_representation**

ID2: the value of the **name** attribute for **value_representation_item**

DER_UNIT_NAME: the value of the **name** attribute for **derived_unit**

Template body:

```
/PROD_DEF_PROP_DEF_HELP/
/PROP_DEF_TO_SPECIAL_VAL_REP_ITEM(&ID1, &ID2, &DER_UNIT_NAME)/
```

5.1.1.34 PROD_DEF_TO_UNITS

The PROD_DEF_TO_UNITS mapping template specifies a reference path constraint in which a **product_definition** is linked with a **global_unit_assigned_context**. This link is through a **property_definition_representation** with an attribute **name** that has the value ID. PROD_DEF_-TO_UNITS is intended to be used by other mapping templates.

Mapping signature:

/PROD_DEF_TO_UNITS(ID)/

Parameter definitions:

ID: the value of the **name** attribute for **property_definition_representation**

Template body:

```
/PROD_DEF_PROP_DEF_HELP/
/PROP_DEF_TO_UNITS(&ID)/
```

5.1.1.35 PROD_DEF_TO_VAL_REP_ITEM

The PROD_DEF_TO_VAL_REP_ITEM mapping template specifies a reference path constraint in which a **product_definition** is linked with a **value_representation_item** that refers to a **measure_-value** whose type is specified by MEAS. This link is through a **property_definition_representation** with an attribute **name** that has the value ID1. The name attribute of the **value_representation_item** will be assigned the value ID2. PROD_DEF_TO_VAL_REP_ITEM is intended to be used by other mapping templates.

Mapping signature:

/PROD_DEF_TO_VAL_REP_ITEM(ID1, ID2, MEAS)/

Parameter definitions:

ID1: the value of the **name** attribute for **property_definition_representation**

ID2: the value of the **name** attribute for **value_representation_item**

MEAS: type of **measure_value** that is specifed

Template body:

```
/PROD_DEF_PROP_DEF_HELP/
/PROP_DEF_TO_VAL_REP_ITEM(&ID1, &ID2, &MEAS)/
```

5.1.1.36 PROP_DEF_REP_HELP

The PROP_DEF_REP_HELP mapping template specifies a reference path constraint in which a **property_definition** is linked with a **property_definition_representation**. The **name** attribute for property_definition_representation is given the value ID. PROP_DEF_REP_HELP is intended to be used by other mapping templates.

Mapping signature:

/PROP_DEF_REP_HELP(ID)/

Parameter definitions:

ID: the value of the **name** attribute for **property_definition_representation**

Template body:

```
property_definition
represented_definition = property_definition
represented_definition <-
{/PDR_NAME(&ID)/}
```

5.1.1.37 PROP_DEF_TO_CART_POINT

The PROP_DEF_TO_CART_POINT mapping template specifies a reference path constraint in which a **property_definition** is linked with a **geometric_representation_item** that has an attribute **name** with the specific value ID, and is a **cartesian_point**.

Mapping signature:

/PROP_DEF_TO_CART_POINT(ID)/

Parameter definitions:

ID: the value of the **name** attribute for the **geometric_representation_item**

Template body:

```
/PROP_DEF_TO_REP/
representation.items [i] ->
representation_item =>
{representation_item.name = &ID}
geometric_representation_item =>
point =>
cartesian_point
```

5.1.1.38 PROP_DEF_TO_DESC_REP_ITEM

The PROP_DEF_TO_DESC_REP_ITEM mapping template specifies a reference path constraint in which a **property_definition** is linked with a **descriptive_representation_item** that has an attribute **name** with the specific value ID2. This link is through a **property_definition_representation** that has an attribute **name** with the specific value ID1.

Mapping signature:

/PROP_DEF_TO_DESC_REP_ITEM(ID1, ID2)/

Parameter definitions:

ID1: the value of the **name** attribute for **property_definition_representation**

ID2: the value of the **name** attribute for **descriptive_representation_item**

Template body:

```
/PROP_DEF_REP_HELP(&ID1)/
/REP_ITEM(&ID2)/
descriptive_representation_item
descriptive_representation_item.description
```

5.1.1.39 PROP_DEF_TO_REP

The PROP_DEF_TO_REP mapping template specifies a reference path constraint in which a **property_definition** is linked with a **representation** via a **property_definition_representation**.

Mapping signature:

/PROP_DEF_TO_REP/

Parameter definitions:

none

Template body:

```
property_definition
represented_definition = property_definition
represented_definition <-
property_definition_representation.definition
property_definition_representation
property_definition_representation.used_representation ->
representation
```

5.1.1.40 PROP_DEF_TO_SPECIAL_VAL_REP_ITEM

The PROP_DEF_TO_SPECIAL_VAL_REP_ITEM mapping template specifies a reference path constraint in which a **property_definition** is linked with a **value_representation_item** that refers to a **measure_value** that is a **context_dependent_measure**. This link is through a **property_-definition_representation** whose **name** attribute will be assigned the value ID1. The name attribute of the **value_representation_item** will be assigned the value ID2. The units for the **context_dependent_measure** will be defined by a **derived_unit** that has a **name** attribute with the value DER_UNIT_NAME.

Mapping signature:

/PROP_DEF_TO_SPECIAL_VAL_REP_ITEM(ID1, ID2, DER_UNIT_NAME)/

Parameter definitions:

ID1: the value of the **name** attribute for **property_definition_representation**

ID2: the value of the **name** attribute for **value_representation_item**

DER_UNIT_NAME: the value of the **name** attribute for **derived_unit**

Template body:

```
/PROP_DEF_REP_HELP(&ID1)/
/REP_TO_SPECIAL_VAL_REP_ITEM(&ID2, &DER_UNIT_NAME)/
```

5.1.1.41 PROP_DEF_TO_UNITS

The PROP_DEF_TO_UNITS mapping template specifies a reference path constraint in which that links a **property_definition** is linked with a **global_unit_assigned_context** via a **property_-definition_representation** whose attribute **name** has the value ID. The property_definition_representation refers to the **global_unit_assigned_context** via a **representation** instance.

Mapping signature:

/PROP_DEF_TO_UNITS(ID)/

Parameter definitions:

ID: the value of the **name** attribute for **property_definition_representation**

Template body:

```
/PROP_DEF_REP_HELP(&ID)/
representation
representation.context_of_items ->
representation_context =>
global_unit_assigned_context
global_unit_assigned_context.units
```

5.1.1.42 PROP_DEF_TO_VAL_REP_ITEM

The PROP_DEF_TO_VAL_REP_ITEM mapping template specifies a reference path constraint in which a **property_definition** is linked with a **value_representation_item** that refers to a **measure_-value** whose type is specified by MEAS. This link is through a **property_definition_representation** with an attribute **name** that has the value ID1. The name attribute of the **value_representation_item** will be assigned the value ID2.

Mapping signature:

/PROP_DEF_TO_VAL_REP_ITEM(ID1, ID2, MEAS)/

Parameter definitions:

ID1: the value of the **name** attribute for **property_definition_representation**

ID2: the value of the **name** attribute for **value_representation_item**

MEAS: type of **measure_value** that is specifed

Template body:

```
/PROP_DEF_REP_HELP(&ID1)/
/REP_ITEM(&ID2)/
value_representation_item
value_representation_item.value_component ->
measure_value
{measure_value = &MEAS}
```

5.1.1.43 PROP_TO_PROD_DEF

The PROP_TO_PROD_DEF mapping template specifies a reference path constraint in which a **property_definition** is linked to a **product_definition**.

Mapping signature:

/PROP_TO_PROD_DEF/

Parameter definitions:

none

Template body:

```
property_definition
property_defintion.definition ->
characterized_definition
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
```

5.1.1.44 RELATE_ACT_2_VO

The RELATE_ACT_2_VO mapping template specifies a reference path constraint in which an entity of type T that is assigned an **action** must have an assigned classification of type 'versionable object'.

Mapping signature:

/RELATE_ACT_2_VO(T)/

Parameter definitions:

T: type of the instance that is assigned an **action**

Template body:

```
applied_action_assignment
applied_action_assignment.items[i] ->
action_item = &T
{/CLASS_ID(&T, 'versionable object')/}
```

5.1.1.45 RELATE_GROUP_2_DO

The RELATE_GROUP_2_DO mapping template specifies a reference path constraint in which an entity of type T is assigned a **group** with the role ROLE, and the entity T must have an assigned classification of type 'definable object'.

Mapping signature:

/RELATE_GROUP_2_DO(T, ROLE)/

Parameter definitions:

T: type of object being assigned to the **group**

ROLE: the name given to the role of the assignment

Template body:

```
/GROUPS(&T, &ROLE)/
{/CLASS_ID(&T,'definable object')/}
```

5.1.1.46 RELATE_GROUP_2_VO

The RELATE_GROUP_2_VO mapping template specifies a reference path constraint in which an entity of type T is assigned a **group** with the role ROLE, and the entity T must have an assigned classification of type 'versionable object'.

Mapping signature:

/RELATE_GROUP_2_VO(T, ROLE)/

Parameter definitions:

T: type of object being assigned to the **group**

ROLE: the name given to the role of the assignment

Template body:

```
/GROUPS(&T, &ROLE)/
{/CLASS_ID(&T,'versionable object')/}
```

5.1.1.47 RELATE_ID_2_VO

The RELATE_ID_2_VO mapping template specifies a reference path constraint in which an entity of type T is assigned an **identification_assignment** with the role name of 'version identifier', and the entity T must have an assigned classification of type 'versionable object'.

NOTE This mapping template can be understood as the "inverse" mapping template to VERSION_ID.

Mapping signature:

/RELATE_ID_2_VO(T)/

Parameter definitions:

T: type of object that has an **identification_assignment**

Template body:

```
identification_assignment
identification_assignment.role ->
identification_role
{identification_role.name = 'version identifier'}
identification_role =>
applied_identification_assignment
applied_identification_assignment.items[i] ->
identification_item = &T
{/CLASS_ID(&T, 'versionable object')/}
```

5.1.1.48 REP_ITEM

The REP_ITEM mapping template specifies a reference path constraint in which a **representation** is linked with a **representation_item**. The **representation_item.name** is assigned the value ID. REP_ITEM is intended to be used by other mapping templates.

Mapping signature:

/REP_ITEM(ID)/

Parameter definitions:

ID: value given the **name** attribute for **representation_item**

Template body:

```
representation
representation.items [i] ->
representation_item
{representation_item.name = &ID}
```

```
representation_item =>
```

5.1.1.49 REP_TO_SPECIAL_VAL_REP_ITEM

The REP_TO_SPECIAL_VAL_REP_ITEM mapping template specifies a reference path constraint in which a **representation** is linked with a **value_representation_item** that refers to a **measure_-value** that is a **context_dependent_measure**. The name attribute of the **value_representation_item** will be assigned the value ID. The units for the **context_dependent_measure** will be defined by a **derived_unit** that has a **name** attribute with the value DER_UNIT_NAME.

Mapping signature:

/REP_TO_SPECIAL_VAL_REP_ITEM(ID, DER_UNIT_NAME)/

Parameter definitions:

ID: the value of the **name** attribute for **value_representation_item**

DER_UNIT_NAME: the value of the **name** attribute for **derived_unit**

Template body:

```
representation
{representation.context_of_items ->
representation_context =>
global_unit_assigned_context
global_unit_assigned_context.units[i] ->
unit
unit = derived_unit
/NAME_ASSGN_WITH_VAL(derived_unit,&DER_UNIT_NAME)/}
representation.items [i] ->
representation_item
{representation_item.name = &ID}
representation_item =>
value_representation_item
value_representation_item.value_component ->
measure_value
{measure_value = context_dependent_measure}
```

5.1.1.50 REP_TO_VAL_REP_ITEM

The REP_TO_VAL_REP_ITEM mapping template specifies a reference path constraint in which a **representation** is linked with a **value_representation_item**. The **value_representation_item** attribute **name** is given the value ID, and the **value_component** is a **measure_value** specified by MEAS.

Mapping signature:

/REP_TO_VAL_REP_ITEM(ID, MEAS)/

Parameter definitions:

ID: value given the **name** attribute for **value_representation_item**

MEAS: type of **measure_value** that is specifed

Template body:


```
/REP_ITEM(&ID)/
value_representation_item
value_representation_item.value_component ->
measure_value
{measure_value = &MEAS}
```

5.1.1.51 ROLE_ASSGN

The ROLE_ASSGN mapping template specifies a reference path constraint in which a **role_association** is assigned to an instance of ENTITY that contains a derived role attribute. The derivation of the role attribute is based on the Basic_attribute_schema of ISO 10303-41.

NOTE The following entity data types use the ROLE_ASSGN for population of their role attribute:
— action_assignment;
— action_request_assignment;
— approval_assignment;
— approval_date_time;
— certification_assignment;
— contract_assignment;
— document_reference;
— effectivity_assignment;
— external_referent_assignment;
— group_assignment;
— name_assignment;
— security_classification_assignment.

Mapping signature:

/ROLE_ASSGN(ENTITY)/

Parameter definitions:

ENTITY: the entity type to which the role_association is assigned

Template body:

```
&ENTITY
role_select = &ENTITY
role_select <-
role_association.item_with_role
role_association
role_association.role ->
object_role
```

5.1.1.52 ROOT_CLASS

The ROOT_CLASS mapping template specifies a reference path constraint in which instances of type T are the **classification_item**s within instances of **applied_classification_assignment**, where the role name specified for the assignment is ‘class membership’, and the assigned classification is a **group** whose name attribute is assigned the value ID.

NOTE The semantics of the template is to assign a class name to an instance.

Mapping signature:

/ROOT_CLASS(T, ID)/

Parameter definitions:

T: type of the instance that is classified

ID: class name

Template body:

```
&T
classification_item = &T
classification_item <-
applied_classification_assignment.items[i]
applied_classification_assignment <=
classification_assignment
{classification_assignment.role ->
classification_role
classification_role.name = 'class membership'}
classification_assignment.assigned_class ->
group
{group.name = &ID}
group =>
class
```

5.1.1.53 SDR_NAME

The SDR_NAME mapping template specifies a reference path constraint in which a **shape_definition_representation**.**name** is assigned the value NAME.

Mapping signature:

/SDR_NAME(NAME)/

Parameter definitions:

NAME: the value for the **name** attribute for **shape_definition_representation**

Template body:

```
property_definition_representation.definition
property_definition_representation
{property_definition_representation =>
shape_definition_representation}
{/NAME_ASSGN_WITH_VAL(property_definition_representation, &NAME)/}
property_definition_representation.used_representation
```

5.1.1.54 SUBTYPE

The SUBTYPE mapping template specifies a reference to the mapping of a subtype of the current application object. Several such references may be included for one supertype application object.

NOTE This template definition only consists of a template signature, there is no matching template body. The template is included to ease the automatic processing of the mapping specification.

Mapping signature:

/SUBTYPE(application_object)/

Parameter definitions:

application_object: the application object that is a subtype of the current supertype application

object and that has the entire or a part of the mapping specification of this supertype.

5.1.1.55 SUPERTYPE

The SUPERTYPE mapping template specifies a reference to the mapping of a supertype of the current application object. Several such references may be included for the subtype application object.

NOTE This template only consists of a signature, there is no matching body. The template is included to ease the automatic processing of the mapping specification.

Mapping signature:

/SUPERTYPE(application_object)/

Parameter definition:

application_object: the application object that is a supertype of the current subtype application object and that has the entire or a part of the mapping specification of this subtype.

5.1.1.56 VAL_REP_ITEM

The VAL_REP_ITEM mapping template specifies the **name** for a **value_representation_item**. The **value_representation_item** attribute **name** is given the value ID, and the **value_component** is a **measure_value** specified by MEAS.

Mapping signature:

/VAL_REP_ITEM(ID, MEAS)/

Parameter definitions:

ID: the value for the **name** attribute for a **value_representation_item**

MEAS: type of **measure_value** that is specifed for the **value_component** attribute of the **value_-representation_item**

Template body:

```
representation_item
{representation_item.name = &ID}
representation_item =>
value_representation_item
value_representation_item.value_component
{value_representation_item.value_component = &MEAS}
```

5.1.1.57 VERSION_ID

The VERSION_ID mapping template specifies a reference path constraint in which an entity of type T is given a version identifier.

Mapping signature:

/VERSION_ID(T)/

Parameter definitions:

T: type of instance that is given a version identifier.

Template body:

```
/IDENTIFICATION(&T)/
identification_role.name = 'version identifier'}
identification_assignment
identification_assignment.assigned_id
```

5.1.2 Arrangement_relationships UoF

5.1.2.1 ADJACENT_SPACE_SURFACE_AREA

AIM element: property_definition
Source: ISO 10303-41
Reference path: {/ROOT_CLASS(property_definition, 'adjacent space surface area')/}

5.1.2.1.1 surface_area

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.101
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('adjacent space surface area parameters', 'surface area', area_measure)/

5.1.2.2 SPACE_ADJACENCY_RELATIONSHIP

AIM element: product_definition_relationship
Source: ISO 10303-41
Reference path: product_definition_relationship
{[product_definition_relationship.name = ‘space adjacency relationship’]
[product_definition_relationship.id = ‘.UNUSED.’]}
product_definition_relationship
{/CLASS(product_definition_relationship, 'space adjacency relationship', 'space arrangement relationship')/]
[/CLASS(product_definition_relationship, 'space arrangement relationship', 'item relationship')/]
[/CLASS(product_definition_relationship, 'item relationship', 'versionable object')/]
[/ROOT_CLASS(product_definition_relationship, 'versionable object')/]
[/CLASS(product_definition_relationship, 'item relationship', 'definable object')/]
[/ROOT_CLASS(product_definition_relationship, 'definable object')/]}

5.1.2.2.1 adjacency_access

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.169
Reference path: product_definition_relationship
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘space adjacency relationship parameters’, 'adjacency access')/

{(descriptive_representation_item.description = 'TRUE')
(descriptive_representation_item.description = ‘FALSE')}

5.1.2.2.2 adjacency_type

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.169
Reference path: product_definition_relationship
characterized_product_definition = product_definition_relationship
characterized_product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘space adjacency relationship parameters’,
'adjacency type')/
{(descriptive_representation_item.description = 'complete')
(descriptive_representation_item.description = ‘partial')}

5.1.2.2.3 description

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.1)

5.1.2.2.4 version_id

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.2)

5.1.2.2.5 space_adjacency_relationship to adjacent_space_surface_area (as adjacent_space_surface_area)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'space adjacency relationship')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘adjacent space surface area’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, 'adjacent space surface area')/}

5.1.2.2.6 space_adjacency_relationship to external_instance_reference (as external_item_1)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.3)

5.1.2.2.7 space_adjacency_relationship to external_instance_reference (as external_item_2)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.4)

5.1.2.2.8 space_adjacency_relationship to global_id (as id)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.5)

5.1.2.2.9 space_adjacency_relationship to space (as item_1)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.6)

5.1.2.2.10 space_adjacency_relationship to space (as item_2)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.7)

5.1.2.3 SPACE_ARRANGEMENT_RELATIONSHIP

AIM element: product_definition_relationship
Source: ISO 10303-41
Reference path: product_definition_relationship
{[/CLASS(product_definition_relationship, 'space arrangement relationship', 'item relationship')/]
[/CLASS(product_definition_relationship, 'item relationship', 'versionable object')/]
[/ROOT_CLASS(product_definition_relationship, 'versionable object')/]
[/CLASS(product_definition_relationship, 'item relationship', 'definable object')/]
[/ROOT_CLASS(product_definition_relationship, 'definable object')/]}

5.1.2.3.1 description

AIM element: product_definition_relationship.description
Source: ISO 10303-41

5.1.2.3.2 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition_relationship)/

5.1.2.3.3 space_arrangement_relationship to external_instance_reference (as external_item_1)

AIM element: product_definition_relationship.relating_product_definition
Source: ISO 10303-41
Reference path: {product_definition_relationship.relating_product_definition ->
product_definition
([/CLASS_ID(product_definition, 'compartment')/]
[/EXT_INST_REF(product_definition, 'ship arrangement schema', 'compartment')/])
([/CLASS_ID(product_definition, 'zone')/]
[/EXT_INST_REF(product_definition, 'ship arrangement schema', 'zone')/])
([/CLASS_ID(product_definition, 'deck zone')/]
[/EXT_INST_REF(product_definition, 'ship arrangement schema', 'deck zone')/])}

5.1.2.3.4 space_arrangement_relationship to external_instance_reference (as external_item_2)

AIM element: product_definition_relationship.related_product_definition
Source: ISO 10303-41
Reference path: {product_definition_relationship.related_product_definition ->

product_definition
([/CLASS_ID(product_definition, 'compartment')/]
[/EXT_INST_REF(product_definition, 'ship arrangement schema',
'compartment')/])
([/CLASS_ID(product_definition, 'zone')/]
[/EXT_INST_REF(product_definition, 'ship arrangement schema', 'zone')/])
([/CLASS_ID(product_definition, 'deck zone')/]
[/EXT_INST_REF(product_definition, 'ship arrangement schema', 'deck zone')/])}

5.1.2.3.5 space_arrangement_relationship to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.66
Reference path: product_definition_relationship
identification_item = product_definition_relationship <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.2.3.6 space_arrangement_relationship to space (as item_1)

AIM element: product_definition_relationship.relating_product_definition
Source: ISO 10303-41
Reference path: {product_definition_relationship.relating_product_definition ->
product_definition
([/CLASS_ID(product_definition, 'compartment')/])
([/CLASS_ID(product_definition, 'zone')/])
([/CLASS_ID(product_definition, 'deck zone')/])}

5.1.2.3.7 space_arrangement_relationshipto space (as item_2)

AIM element: product_definition_relationship.related_product_definition
Source: ISO 10303-41
Reference path: {product_definition_relationship.related_product_definition ->
product_definition
([/CLASS_ID(product_definition, 'compartment')/])
([/CLASS_ID(product_definition, 'zone')/])
([/CLASS_ID(product_definition, 'deck zone')/])}

5.1.2.4 SPACE_CONNECTION_RELATIONSHIP

AIM element: product_definition_relationship
Source: ISO 10303-41
Reference path: product_definition_relationship
{/CLASS(product_definition_relationship, 'space connection relationship', 'space arrangement relationship')/]
[/CLASS(product_definition_relationship, 'space arrangement relationship', 'item relationship')/]
[/CLASS(product_definition_relationship, 'item relationship', 'versionable object')/]
[/ROOT_CLASS(product_definition_relationship, 'versionable object')/]
[/CLASS(product_definition_relationship, 'item relationship', 'definable object')/]
[/ROOT_CLASS(product_definition_relationship, 'definable object')/]}

5.1.2.4.1 connecting_system

AIM element: descriptive_representation_item.description

Source: ISO 10303-45
Rules: 5.2.4.208
Reference path: product_definition_relationship
characterized_product_definition = product_definition_relationship
characterized_product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘space connection relationship parameters’, 'connecting system')/

5.1.2.4.2 description

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.1)

5.1.2.4.3 version_id

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.2)

5.1.2.4.4 space_connection_relationship to external_instance_reference (as connecting_system)

AIM element: PATH
Reference path: product_definition_relationship
{[/CLASS_ID(product_definition_relationship, 'space connection relationship')/]
[/EXT_INST_REF(product_definition, 'plant spatial configuration', 'plant item')/]}

5.1.2.4.5 space_connection_relationship to external_instance_reference (as external_item_1)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.3)

5.1.2.4.6 space_connection_relationship to external_instance_reference (as external_item_2)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.4)

5.1.2.4.7 space_connection_relationship to global_id (as id)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.5)

5.1.2.4.8 space_connection_relationship to space (as item_1)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.6)

5.1.2.4.9 space_connection_relationship to space (as item_2)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.7)

5.1.2.5 SPACE_ENCLOSING_RELATIONSHIP

AIM element: product_definition_relationship
Source: ISO 10303-41
Reference path: product_definition_relationship
{[/CLASS(product_definition_relationship, 'space enclosing relationship', 'space arrangement relationship')/]
[/CLASS(product_definition_relationship, 'space arrangement relationship', 'item relationship')/]

[/CLASS(product_definition_relationship, 'item relationship', 'versionable object')/]
[/ROOT_CLASS(product_definition_relationship, 'versionable object')/]
[/CLASS(product_definition_relationship, 'item relationship', 'definable object')/]
[/ROOT_CLASS(product_definition_relationship, 'definable object')/]}

5.1.2.5.1 description

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.1)

5.1.2.5.2 version_id

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.2)

5.1.2.5.3 space_enclosing_relationship to external_instance_reference (as external_item_1)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.3)

5.1.2.5.4 space_enclosing_relationship to external_instance_reference (as external_item_2)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.4)

5.1.2.5.5 space_enclosing_relationship to global_id (as id)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.5)

5.1.2.5.6 space_enclosing_relationship to space (as item_1)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.6)

5.1.2.5.7 space_enclosing_relationship to space (as item_2)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.7)

5.1.2.6 SPACE_POSITIONAL_RELATIONSHIP

AIM element: product_definition_relationship
Source: ISO 10303-41
Reference path: product_definition_relationship
{[/CLASS(product_definition_relationship, 'space positional relationship', 'space arrangement relationship')/]
[/CLASS(product_definition_relationship, 'space arrangement relationship', 'item relationship')/]
[/CLASS(product_definition_relationship, 'item relationship', 'versionable object')/]
[/ROOT_CLASS(product_definition_relationship, 'versionable object')/]
[/CLASS(product_definition_relationship, 'item relationship', 'definable object')/]
[/ROOT_CLASS(product_definition_relationship, 'definable object')/]}

5.1.2.6.1 description

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.1)

5.1.2.6.2 relationship_type

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.170

Reference path: product_definition_relationship
characterized_product_definition = product_definition_relationship
characterized_product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘space positional relationship parameters’,
'relationship type')/
{(descriptive_representation_item.description = 'above')
(descriptive_representation_item.description = ‘aft longitudinal extent')
(descriptive_representation_item.description = ‘below')
(descriptive_representation_item.description = ‘forward longitudinal extent')
(descriptive_representation_item.description = ‘matched transverse')
(descriptive_representation_item.description = ‘port transverse extent')
(descriptive_representation_item.description = ‘relative')
(descriptive_representation_item.description = ‘starboard transverse extent')}

5.1.2.6.3 version_id

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.2)

5.1.2.6.4 space_positional_relationship to external_instance_reference (as external_item_1)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.3)

5.1.2.6.5 space_positional_relationship to external_instance_reference (as external_item_2)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.4)

5.1.2.6.6 space_positional_relationship to global_id (as id)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.5)

5.1.2.6.7 space_positional_relationship to space (as item_1)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.6)

5.1.2.6.8 space_positional_relationship to space (as item_2)

AIM element: /SUPERTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.7)

5.1.3 Cargoes UoF

5.1.3.1 ABSOLUTE_CARGO_POSITION

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.100
Reference path: {[/CLASS(property_definition, ’absolute cargo position’, ‘cargo position’)/]
[/ROOT_CLASS(property_definition, ’cargo position’)/]}

5.1.3.1.1 orientation

AIM element: value_representation_item.value_component
Source: ISO 10303-43

Reference path: /PROP_DEF_TO_VAL_REP_ITEM('absolute cargo position parameters', 'orientation', plane_angle_measure)/

5.1.3.1.2 position

AIM element: cartesian_point
Source: ISO 10303-42
Reference path: property_definition = represented_definition
represented_definition <-
/PDR_NAME('absolute cargo position parameters')/ ->
representation
representation.items[i] ->
/GEO_REP_ITEM('position', cartesian_point)/

5.1.3.1.3 absolute_cargo_position to global_axis_placement (as parent_coordinate_system)

AIM element: PATH
Reference path: property_definition
property_definition.definition ->
characterized_definition
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
product_definition <-
product_definition_relationship.relating_product_definition
product_definition_relationship
{product_definition_relationship.name = ‘parent coordinate system’}
product_definition_relationship.related_product_definition ->
product_definition
{/CLASS_ID(product_definition, ‘global axis placement’)/}

5.1.3.2 BAY_CELL_POSITION

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, ’bay cell position’, ‘bay position’)/]
[/CLASS(property_definition, ’bay position’, ‘cargo position’)/]
[/ROOT_CLASS(property_definition, ’cargo position’)/]}

5.1.3.2.1 Bay_cell_position to cargo_bay_definition (as relating_to)

AIM element: PATH
Reference path: property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘relating to’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, ‘cargo bay definition’)/}

5.1.3.2.2 Bay_cell_position to longitudinal_position (as row)

AIM element: PATH
Reference path: property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship

{property_definition_relationship.name = ‘relating to’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, ‘cargo bay definition’)/}
property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
{/CLASS_ID(product_definition, 'compartment')/}
product_definition <-
product_definition_relationship.relating_definition
product_definition_relationship
product_definition_relationship.related_definition ->
product_definition
product_definition = characterized_product_definition
characterized_product_definition = characterized_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, 'longitudinal table')/}
/PROP_DEF_TO_REP/
representation.items [i] ->
representation_item =>
compound_representation_item
{representation_item.name = 'row’}
{/CLASS_ID(compound_representation_item, ‘longitudinal position’)/}

5.1.3.2.3 Bay_cell_position to transversal_position (as bay_number)

AIM element: PATH
Reference path: property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘relating to’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, ‘cargo bay definition’)/}
property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
{/CLASS_ID(product_definition, 'compartment')/}
product_definition <-
product_definition_relationship.relating_definition
product_definition_relationship
product_definition_relationship.related_definition ->
product_definition
product_definition = characterized_product_definition
characterized_product_definition = characterized_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, 'transversal table')/}

/PROP_DEF_TO_REP/
representation.items [i] ->
representation_item =>
compound_representation_item
{representation_item.name = 'bay number'}
{/CLASS_ID(compound_representation_item, 'transversal position')/}

5.1.3.2.4 Bay_cell_position to vertical_position (as tier)

AIM element: PATH
Reference path: property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = 'relating to'}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, 'cargo bay definition')/}
property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
{/CLASS_ID(product_definition, 'compartment')/}
product_definition <-
product_definition_relationship.relating_definition
product_definition_relationship
product_definition_relationship.related_definition ->
product_definition
product_definition = characterized_product_definition
characterized_product_definition = characterized_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, 'vertical table')/}
/PROP_DEF_TO_REP/
representation.items [i] ->
representation_item =>
compound_representation_item
{representation_item.name = 'tier'}
{/CLASS_ID(compound_representation_item, 'vertical position')/}

5.1.3.3 BAY_POSITION

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, 'bay position', 'cargo position')/]
[/ROOT_CLASS(property_definition, 'cargo position')/]}

5.1.3.3.1 Bay_position to cargo_bay_definition (as relating_to)

AIM element: PATH
Reference path: property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = 'relating to'}

property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, ‘cargo bay definition’)/}

5.1.3.4 BULK_CARGO

AIM element: product
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.102, 5.2.4.185
Reference path: {/ROOT_CLASS(product, ’bulk cargo’)/}

5.1.3.4.1 description

AIM element: product.description
Source: ISO 10303-41

5.1.3.4.2 flash_point

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'bulk cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('bulk cargo parameters', 'flash point',
thermodynamic_temperature_measure)/

5.1.3.4.3 natural_angle_of_repose

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'bulk cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('bulk cargo parameters', 'natural angle of
repose', plane_angle_measure)/

5.1.3.4.4 permeability

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'bulk cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('bulk cargo parameters', 'permeability',
ratio_measure)/

5.1.3.4.5 pollution_code

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product
{/CLASS_ID(product, 'bulk cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘bulk cargo parameters’, 'pollution code')/
{(descriptive_representation_item.description = 'code A')
(descriptive_representation_item.description = 'code B')
(descriptive_representation_item.description = 'code C')
(descriptive_representation_item.description = 'code D')}

5.1.3.4.6 required_carriage_temperature

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'bulk cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('bulk cargo parameters', 'required carriage temperature', thermodynamic_temperature_measure)/

5.1.3.4.7 stowage_factor

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'bulk cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('bulk cargo parameters', 'stowage factor', volume_measure)/

5.1.3.4.8 type_of

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.229
Reference path: product
{/CLASS_ID(product, 'bulk cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘bulk cargo parameters’, 'type of')/
{(descriptive_representation_item.description = ‘cement')
(descriptive_representation_item.description = 'coal')
(descriptive_representation_item.description = 'fish')
(descriptive_representation_item.description = 'general')

(descriptive_representation_item.description = 'grain')
(descriptive_representation_item.description = 'mud')
(descriptive_representation_item.description = 'ore')
(descriptive_representation_item.description = 'sugar')
(descriptive_representation_item.description = 'timber')
(descriptive_representation_item.description = 'unspecified')
(descriptive_representation_item.description = 'user defined')}

5.1.3.4.9 un_type_code

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'bulk cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('bulk cargo parameters', 'un type code',
numeric_measure)/

5.1.3.4.10 user_def_cargo

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.229
Reference path: product
{/CLASS_ID(product, 'bulk cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘bulk cargo parameters’, 'user def cargo')/

5.1.3.4.11 bulk_cargo to dangerous_goods_code (as cargo_hazard)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'bulk cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, 'dangerous goods code')/}

5.1.3.4.12 bulk_cargo to document_reference_with_address (as references)

AIM element: PATH
Reference path: product
/DOC_REF(product,’references’)/
document

{/CLASS_ID(document, ‘document reference with address’)/}

5.1.3.4.13 bulk_cargo to general_cargo_material_properties (as material_properties)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'bulk cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{(/CLASS_ID(property_definition, 'general cargo material properties')/)
(/CLASS_ID(property_definition, 'detailed cargo material properties')/)}

5.1.3.5 BULK_CARGO_ASSIGNMENT

AIM element: product_definition_relationship
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.103
Reference path: {[/CLASS(product_definition_relationship, 'bulk cargo assignment', 'compartment cargo assignment')/]
[/CLASS(product_definition_relationship, 'compartment cargo assignment', 'cargo assignment')/]
[/ROOT_CLASS(product_definition_relationship, 'cargo assignment')/]}

5.1.3.5.1 actual_angle_of_repose

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'bulk cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_VAL_REP_ITEM('bulk cargo assignment parameters', 'actual angle of repose', plane_angle_measure)/

5.1.3.5.2 assignment_context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'bulk cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘bulk cargo assignment parameters’, 'assignment context')/

{(descriptive_representation_item.description = 'accomodation')
(descriptive_representation_item.description = 'cargo load')
(descriptive_representation_item.description = 'consumables')
(descriptive_representation_item.description = 'stores')
(descriptive_representation_item.description = 'unspecified')}

5.1.3.5.3 cargo_height

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'bulk cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_VAL_REP_ITEM('bulk cargo assignment parameters', 'cargo height', length_measure)/

5.1.3.5.4 cargo_identifier

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'bulk cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘bulk cargo assignment parameters’, 'cargo identifier')/

5.1.3.5.5 trimmed

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'bulk cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘bulk cargo assignment parameters’, 'trimmed')/
{(descriptive_representation_item.description = 'true')
(descriptive_representation_item.description = 'false')}

5.1.3.5.6 bulk_cargo_assignment to cargo (as cargo)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'bulk cargo assignment')/}
product_definition_relationship.related_product_definition ->
product_definition

product_definition.formation ->
product_definition_formation
product_definition_formation.of_product ->
product
{/CLASS_ID(product, ‘bulk cargo’)/}

5.1.3.5.7 bulk_cargo_assignment to compartment (as compartment)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'bulk cargo assignment')/}
product_definition_relationship.relating_product_definition ->
product_definition
{/CLASS_ID(product_definition, 'compartment')/}

5.1.3.5.8 bulk_cargo_assignment to weight_and_centre_of_gravity (as allocated_weight)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'bulk cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘allocated weight’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, ‘weight and centre of gravity’/}

5.1.3.6 CARGO

#1: If the cargo is a dry_cargo
#2: If the cargo is a gaseous_cargo
#3: If the cargo is a liquid_cargo
#4: If the cargo is a bulk_cargo
#5: If the cargot is a unit_cargo

AIM element: #1: /SUBTYPE(dry_cargo)/ -- (see 5.1.3.14)
#2: /SUBTYPE(gaseous_cargo)/ -- (see 5.1.3.15)
#3: /SUBTYPE(liquid_cargo)/ -- (see 5.1.3.19)
#4: /SUBTYPE(bulk_cargo)/ -- (see 5.1.3.4)
#5: /SUBTYPE(unit_cargo)/ -- (see 5.1.3.23)

5.1.3.6.1 description

AIM element: #1: /SUBTYPE(dry_cargo)/ -- (see 5.1.3.14.1)
#2: /SUBTYPE(gaseous_cargo)/ -- (see 5.1.3.15.3)
#3: /SUBTYPE(liquid_cargo)/ -- (see 5.1.3.19.2)
#4: /SUBTYPE(bulk_cargo)/ -- (see 5.1.3.4.1)
#5: /SUBTYPE(unit_cargo)/ -- (see 5.1.3.23.2)

5.1.3.6.2 flash_point

AIM element: #1: /SUBTYPE(dry_cargo)/ -- (see 5.1.3.14.2)
#2: /SUBTYPE(gaseous_cargo)/ -- (see 5.1.3.15.4)
#3: /SUBTYPE(liquid_cargo)/ -- (see 5.1.3.19.3)
#4: /SUBTYPE(bulk_cargo)/ -- (see 5.1.3.4.2)
#5: /SUBTYPE(unit_cargo)/ -- (see 5.1.3.23.3)

5.1.3.6.3 pollution_code

AIM element: #1: /SUBTYPE(dry_cargo)/ -- (see 5.1.3.14.4)
#2: /SUBTYPE(gaseous_cargo)/ -- (see 5.1.3.15.5)
#3: /SUBTYPE(liquid_cargo)/ -- (see 5.1.3.19.4)
#4: /SUBTYPE(bulk_cargo)/ -- (see 5.1.3.4.5)
#5: /SUBTYPE(unit_cargo)/ -- (see 5.1.3.23.6)

5.1.3.6.4 required_carriage_temperature

AIM element: #1: /SUBTYPE(dry_cargo)/ -- (see 5.1.3.14.5)
#2: /SUBTYPE(gaseous_cargo)/ -- (see 5.1.3.15.7)
#3: /SUBTYPE(liquid_cargo)/ -- (see 5.1.3.19.6)
#4: /SUBTYPE(bulk_cargo)/ -- (see 5.1.3.4.6)
#5: /SUBTYPE(unit_cargo)/ -- (see 5.1.3.23.7)

5.1.3.6.5 un_type_code

AIM element: #1: /SUBTYPE(dry_cargo)/ -- (see 5.1.3.14.7)
#2: /SUBTYPE(gaseous_cargo)/ -- (see 5.1.3.15.8)
#3: /SUBTYPE(liquid_cargo)/ -- (see 5.1.3.19.7)
#4: /SUBTYPE(bulk_cargo)/ -- (see 5.1.3.4.9)
#5: /SUBTYPE(unit_cargo)/ -- (see 5.1.3.23.10)

5.1.3.6.6 user_def_cargo

AIM element: #1: /SUBTYPE(dry_cargo)/ -- (see 5.1.3.14.8)
#2: /SUBTYPE(gaseous_cargo)/ -- (see 5.1.3.15.9)
#3: /SUBTYPE(liquid_cargo)/ -- (see 5.1.3.19.8)
#4: /SUBTYPE(bulk_cargo)/ -- (see 5.1.3.4.10)
#5: /SUBTYPE(unit_cargo)/ -- (see 5.1.3.23.11)

5.1.3.6.7 cargo to dangerous_goods_code (as cargo_hazard)

AIM element: #1: /SUBTYPE(dry_cargo)/ -- (see 5.1.3.14.9)
#2: /SUBTYPE(gaseous_cargo)/ -- (see 5.1.3.15.10)
#3: /SUBTYPE(liquid_cargo)/ -- (see 5.1.3.19.9)
#4: /SUBTYPE(bulk_cargo)/ -- (see 5.1.3.4.11)
#5: /SUBTYPE(unit_cargo)/ -- (see 5.1.3.23.14)

5.1.3.6.8 cargo to document_reference_with_address (as references)

AIM element: #1: /SUBTYPE(dry_cargo)/ -- (see 5.1.3.14.10)
#2: /SUBTYPE(gaseous_cargo)/ -- (see 5.1.3.15.11)
#3: /SUBTYPE(liquid_cargo)/ -- (see 5.1.3.19.10)
#4: /SUBTYPE(bulk_cargo)/ -- (see 5.1.3.4.12)

#5: /SUBTYPE(unit_cargo)/ -- (see 5.1.3.23.15)

5.1.3.6.9 cargo to general_cargo_material_properties (as material_properties)

AIM element: #1: /SUBTYPE(dry_cargo)/ -- (see 5.1.3.14.11)
#2: /SUBTYPE(gaseous_cargo)/ -- (see 5.1.3.15.12)
#3: /SUBTYPE(liquid_cargo)/ -- (see 5.1.3.19.11)
#4: /SUBTYPE(bulk_cargo)/ -- (see 5.1.3.4.13)
#5: /SUBTYPE(unit_cargo)/ -- (see 5.1.3.23.16)

5.1.3.7 CARGO_ASSIGNMENT

#1: If the cargo_assignment is a bulk_cargo_assignment
#2: If the cargo_assignment is a compartment_cargo_assignment
#3: If the cargo_assignment is a deck_cargo_assignment
#4: If the cargo_assignment is a gaseous_cargo_assignment
#5: If the cargo_assignment is a liquid_cargo_assignment
#6: If the cargo_assignment is a unit_cargo_assignment

AIM element: #1: /SUBTYPE(bulk_cargo_assignment)/ -- (see 5.1.3.5)
#2: /SUBTYPE(compartment_cargo_assignment)/ -- (see 5.1.3.10)
#3: /SUBTYPE(deck_cargo_assignment)/ -- (see 5.1.3.12)
#4: /SUBTYPE(gaseous_cargo_assignment)/ -- (see 5.1.3.16)
#5: /SUBTYPE(liquid_cargo_assignment)/ -- (see 5.1.3.20)
#6: /SUBTYPE(unit_cargo_assignment)/ -- (see 5.1.3.24)

5.1.3.7.1 assignment_context

AIM element: #1: /SUBTYPE(bulk_cargo_assignment)/ -- (see 5.1.3.5.2)
#2: /SUBTYPE(compartment_cargo_assignment)/ -- (see 5.1.3.10.1)
#3: /SUBTYPE(deck_cargo_assignment)/ -- (see 5.1.3.12.1)
#4: /SUBTYPE(gaseous_cargo_assignment)/ -- (see 5.1.3.16.1)
#5: /SUBTYPE(liquid_cargo_assignment)/ -- (see 5.1.3.20.1)
#6: /SUBTYPE(unit_cargo_assignment)/ -- (see 5.1.3.24.1)

5.1.3.7.2 cargo_identifier

AIM element: #1: /SUBTYPE(bulk_cargo_assignment)/ -- (see 5.1.3.5.4)
#2: /SUBTYPE(compartment_cargo_assignment)/ -- (see 5.1.3.10.2)
#3: /SUBTYPE(deck_cargo_assignment)/ -- (see 5.1.3.12.2)
#4: /SUBTYPE(gaseous_cargo_assignment)/ -- (see 5.1.3.16.2)
#5: /SUBTYPE(liquid_cargo_assignment)/ -- (see 5.1.3.20.3)
#6: /SUBTYPE(unit_cargo_assignment)/ -- (see 5.1.3.24.2)

5.1.3.7.3 cargo_assignment to weight_and_centre_of_gravity (as allocated_weight)

AIM element: #1: /SUBTYPE(bulk_cargo_assignment)/ -- (see 5.1.3.5.8)
#2: /SUBTYPE(compartment_cargo_assignment)/ -- (see 5.1.3.10.5)
#3: /SUBTYPE(deck_cargo_assignment)/ -- (see 5.1.3.12.6)
#4: /SUBTYPE(gaseous_cargo_assignment)/ -- (see 5.1.3.16.5)
#5: /SUBTYPE(liquid_cargo_assignment)/ -- (see 5.1.3.20.6)
#6: /SUBTYPE(unit_cargo_assignment)/ -- (see 5.1.3.24.7)

5.1.3.8 CARGO_FOOTPRINT

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.106
Reference path: {/ROOT_CLASS(property_definition_representation, 'cargo footprint')/}

5.1.3.8.1 contact_material

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, 'cargo footprint')/}
property_definition_representation.used_representation ->
representation
representation.items[i] ->
{representation_item.name = 'contact material'}
representation_item => descriptive_representation_item
{(descriptive_representation_item.description = 'metal')
(descriptive_representation_item.description = 'other')
(descriptive_representation_item.description = 'pneumatic')
(descriptive_representation_item.description = 'rubber')}

5.1.3.8.2 shape

AIM element: bounded_curve
Source: ISO 10303-42
Reference path: property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, 'cargo footprint')/}
property_definition_representation.used_representation ->
representation
representation.items[i] ->
/GEO_REP_ITEM(‘shape', bounded_curve)/

5.1.3.8.3 transferred_mass

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, 'cargo footprint')/}
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('transferred mass', mass_measure)

5.1.3.9 CARGO_POSITION

#1: If the cargo_position is an absolute_cargo_position
#2: If the cargo_position is a bay_position
#3: If the cargo_position is a relative_cargo_position

AIM element: #1: /SUBTYPE(absolute_cargo_position)/ -- (see 5.1.3.1)
#2: /SUBTYPE(bay_position)/ -- (see 5.1.3.3)

#3: /SUBTYPE(relative_cargo_position)/ -- (see 5.1.3.22)

5.1.3.10 COMPARTMENT_CARGO_ASSIGNMENT

AIM element: product_definition_relationship
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.119
Reference path: {[/CLASS(product_definition_relationship, 'compartment cargo assignment', 'cargo assignment')/]
[/ROOT_CLASS(product_definition_relationship, 'cargo assignment')/]}

5.1.3.10.1 assignment_context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'compartment cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘compartment cargo assignment parameters’, 'assignment context')/
{(descriptive_representation_item.description = 'accomodation')
(descriptive_representation_item.description = 'cargo load')
(descriptive_representation_item.description = 'consumables')
(descriptive_representation_item.description = 'stores')
(descriptive_representation_item.description = 'unspecified')}

5.1.3.10.2 cargo_identifier

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'compartment cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘compartment cargo assignment parameters’, 'cargo identifier')/

5.1.3.10.3 compartment_cargo_assignment to cargo (as cargo)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'compartment cargo assignment')/}
product_definition_relationship.related_product_definition ->
product_definition
product_definition.formation ->
product_definition_formation
product_definition_formation.of_product ->
product
{(/CLASS_ID(product, ‘bulk cargo’)/)
(/CLASS_ID(product, ‘gaseous cargo’)/)

(/CLASS_ID(product, ‘liquid cargo’)/)}

5.1.3.10.4 compartment_cargo_assignment to compartment (as compartment)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'compartment cargo assignment')/}
product_definition_relationship.relating_product_definition ->
product_definition
{/CLASS_ID(product_definition, 'compartment')/}

5.1.3.10.5 compartment_cargo_assignment to weight_and_centre_of_gravity (as allocated_weight)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'compartment cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘allocated weight’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, ‘weight and centre of gravity’/}

5.1.3.11 DANGEROUS_GOODS_CODE

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.145
Reference path: {/ROOT_CLASS(property_definition, 'dangerous goods code')/}

5.1.3.11.1 class

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, 'dangerous goods code parameters')/}
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'class'}
representation_item => descriptive_representation_item
{(descriptive_representation_item.description = ‘class 1’)
(descriptive_representation_item.description = ‘class 21’)
(descriptive_representation_item.description = ‘class 22’)
(descriptive_representation_item.description = ‘class 23’)
(descriptive_representation_item.description = ‘class 3’)

(descriptive_representation_item.description = ‘class 41’)
(descriptive_representation_item.description = ‘class 42’)
(descriptive_representation_item.description = ‘class 43’)
(descriptive_representation_item.description = ‘class 51’)
(descriptive_representation_item.description = ‘class 52’)
(descriptive_representation_item.description = ‘class 61’)
(descriptive_representation_item.description = ‘class 62’)
(descriptive_representation_item.description = ‘class 71’)
(descriptive_representation_item.description = ‘class 72’)
(descriptive_representation_item.description = ‘class 73’)
(descriptive_representation_item.description = ‘class 8’)
(descriptive_representation_item.description = ‘class 9’)}

5.1.3.11.2 subsidiary_risks

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, 'dangerous goods code parameters')/}
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'subsidiary risks'}
representation_item => descriptive_representation_item
{(descriptive_representation_item.description = ‘class 1’)
(descriptive_representation_item.description = ‘class 21’)
(descriptive_representation_item.description = ‘class 22’)
(descriptive_representation_item.description = ‘class 23’)
(descriptive_representation_item.description = ‘class 3’)
(descriptive_representation_item.description = ‘class 41’)
(descriptive_representation_item.description = ‘class 42’)
(descriptive_representation_item.description = ‘class 43’)
(descriptive_representation_item.description = ‘class 51’)
(descriptive_representation_item.description = ‘class 52’)
(descriptive_representation_item.description = ‘class 61’)
(descriptive_representation_item.description = ‘class 62’)
(descriptive_representation_item.description = ‘class 71’)
(descriptive_representation_item.description = ‘class 72’)
(descriptive_representation_item.description = ‘class 73’)
(descriptive_representation_item.description = ‘class 8’)
(descriptive_representation_item.description = ‘class 9’)}

5.1.3.12 DECK_CARGO_ASSIGNMENT

AIM element: product_definition_relationship
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.146
Reference path: {[/CLASS(product_definition_relationship, 'deck cargo assignment', 'cargo assignment')/]
[/ROOT_CLASS(product_definition_relationship, 'cargo assignment')/]}

5.1.3.12.1 assignment_context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'deck cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘deck cargo assignment parameters’, 'assignment context')/
{(descriptive_representation_item.description = 'accomodation')
(descriptive_representation_item.description = 'cargo load')
(descriptive_representation_item.description = 'consumables')
(descriptive_representation_item.description = 'stores')
(descriptive_representation_item.description = 'unspecified')}

5.1.3.12.2 cargo_identifier

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'deck cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘deck cargo assignment parameters’, 'cargo identifier')/

5.1.3.12.3 deck_cargo_assignment to cargo_position (as position)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'deck cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘position’}
property_definition_relationship.related_property_definition ->
property_definition
{(/CLASS_ID(property_definition, ‘absolute cargo position’/)
(/CLASS_ID(property_definition, ‘relative cargo position’/)
(/CLASS_ID(property_definition, ‘bay position’/)}

5.1.3.12.4 deck_cargo_assignment to deck_zone (as deck_zone)

AIM element: PATH

Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'deck cargo assignment')/}
product_definition_relationship.relating_product_definition ->
product_definition
{/CLASS_ID(product_definition, 'deck zone')/}

5.1.3.12.5 deck_cargo_assignment to unit_cargo_group (as cargo)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'deck cargo assignment')/}
product_definition_relationship.related_product_definition ->
product_definition
product_definition.formation ->
product_definition_formation
product_definition_formation.of_product ->
product
{/CLASS_ID(product, ‘unit cargo group’)/}

5.1.3.12.6 deck_cargo_assignment to weight_and_centre_of_gravity (as allocated_weight)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'deck cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘allocated weight’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, ‘weight and centre of gravity’/}

5.1.3.13 DETAILED_CARGO_MATERIAL_PROPERTIES

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.200
Reference path: {/ROOT_CLASS(property_definition, 'detailed cargo material properties')/}

5.1.3.13.1 density

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('detailed cargo material properties parameters', 'density', 'density unit')/

5.1.3.13.2 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.3.13.3 expansion_coefficient

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('detailed cargo material properties parameters', 'expansion coefficient', 'coefficient of thermal expansion unit')/

5.1.3.13.4 specific_heat_capacity

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('detailed cargo material properties parameters', 'specific heat capacity', 'specific heat capacity unit')/

5.1.3.13.5 thermal_conductivity

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('detailed cargo material properties parameters', 'thermal conductivity', 'thermal conductivity unit')/

5.1.3.13.6 viscosity

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('detailed cargo material properties parameters', 'viscosity', 'coefficient of viscosity unit')/

5.1.3.13.7 detailed_cargo_material_properties to document_reference_with_address (as material_reference)

AIM element: PATH
Reference path: property_definition
/DOC_REF(property_definition,'material reference')/
document
{/CLASS_ID(document, 'document reference with address')/}

5.1.3.14 DRY_CARGO

AIM element: product
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.148, 5.2.4.201
Reference path: {/ROOT_CLASS(product, 'dry cargo')/}

5.1.3.14.1 description

AIM element: product.description
Source: ISO 10303-41

5.1.3.14.2 flash_point

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'dry cargo')/}

product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('dry cargo parameters', 'flash point',
thermodynamic_temperature_measure)/

5.1.3.14.3 permeability

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'dry cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('dry cargo parameters', 'permeability',
ratio_measure)/

5.1.3.14.4 pollution_code

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product
{/CLASS_ID(product, 'dry cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘dry cargo parameters’, 'pollution code')/
{(descriptive_representation_item.description = 'code A')
(descriptive_representation_item.description = 'code B')
(descriptive_representation_item.description = 'code C')
(descriptive_representation_item.description = 'code D')}

5.1.3.14.5 required_carriage_temperature

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'dry cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('dry cargo parameters', 'required carriage
temperature', thermodynamic_temperature_measure)/

5.1.3.14.6 stowage_factor

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'dry cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-

/PROD_DEF_TO_VAL_REP_ITEM('dry cargo parameters', 'stowage factor', volume_measure)/

5.1.3.14.7 un_type_code

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'dry cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('dry cargo parameters', 'un type code', numeric_measure)/

5.1.3.14.8 user_def_cargo

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product
{/CLASS_ID(product, 'dry cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘dry cargo parameters’, 'user def cargo')/

5.1.3.14.9 dry_cargo to dangerous_goods_code (as cargo_hazard)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'dry cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, 'dangerous goods code')/}

5.1.3.14.10 dry_cargo to document_reference_with_address (as references)

AIM element: PATH
Reference path: product
/DOC_REF(product,’references’)/
document
{/CLASS_ID(document, ‘document reference with address’)/}

5.1.3.14.11 dry_cargo to general_cargo_material_properties (as material_properties)

AIM element: PATH
Reference path: product

{/CLASS_ID(product, 'dry cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{(/CLASS_ID(property_definition, 'general cargo material properties')/)
(/CLASS_ID(property_definition, 'detailed cargo material properties')/)}

5.1.3.15 GASEOUS_CARGO

AIM element: product
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.151, 5.2.4.202
Reference path: {/ROOT_CLASS(product, ’gaseous cargo’)/}

5.1.3.15.1 cargo_type

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.228
Reference path: product
{/CLASS_ID(product, 'gaseous cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘gaseous cargo parameters’, 'cargo type')/
{(descriptive_representation_item.description = 'acetaldehyde')
(descriptive_representation_item.description = 'anhydrous ammonia')
(descriptive_representation_item.description = 'avcat')
(descriptive_representation_item.description = 'butane')
(descriptive_representation_item.description = 'butadiene')
(descriptive_representation_item.description = 'butylene')
(descriptive_representation_item.description = ‘chlorine')
(descriptive_representation_item.description = 'diethyl ether')
(descriptive_representation_item.description = 'dimethylamine')
(descriptive_representation_item.description = 'ethylene')
(descriptive_representation_item.description = 'ethyl chlorine')
(descriptive_representation_item.description = 'ethylene oxide')
(descriptive_representation_item.description = 'inert gas')
(descriptive_representation_item.description = 'isoprene')
(descriptive_representation_item.description = 'isopropylamine')
(descriptive_representation_item.description = 'liquified natural gas')
(descriptive_representation_item.description = 'liquified petroleum gas')
(descriptive_representation_item.description = 'methane')
(descriptive_representation_item.description = 'methyl chloride')
(descriptive_representation_item.description = 'monoethylamine')
(descriptive_representation_item.description = 'naptha')
(descriptive_representation_item.description = 'propane')
(descriptive_representation_item.description = 'propane butane mix')

(descriptive_representation_item.description = 'propylene oxide')
(descriptive_representation_item.description = 'propylene')
(descriptive_representation_item.description = 'user defined')
(descriptive_representation_item.description = ‘vinyl chloride monomer')
(descriptive_representation_item.description = 'vinyl ethyl ether')}

5.1.3.15.2 carried_in_liquid_state

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product
{/CLASS_ID(product, 'gaseous cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘gaseous cargo parameters’, 'carried in liquid state')/
{(descriptive_representation_item.description = 'true')
(descriptive_representation_item.description = 'false')}

5.1.3.15.3 description

AIM element: product.description
Source: ISO 10303-41

5.1.3.15.4 flash_point

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'gaseous cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('gaseous cargo parameters', 'flash point', thermodynamic_temperature_measure)/

5.1.3.15.5 pollution_code

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product
{/CLASS_ID(product, 'gaseous cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘gaseous cargo parameters’, 'pollution code')/
{(descriptive_representation_item.description = 'code A')
(descriptive_representation_item.description = 'code B')
(descriptive_representation_item.description = 'code C')
(descriptive_representation_item.description = 'code D')}

5.1.3.15.6 required_carriage_pressure

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'gaseous cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_SPECIAL_VAL_REP_ITEM('gaseous cargo parameters',
'required carriage pressure', 'pressure unit')/

5.1.3.15.7 required_carriage_temperature

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'gaseous cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('gaseous cargo parameters', 'required carriage temperature', thermodynamic_temperature_measure)/

5.1.3.15.8 un_type_code

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'gaseous cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('gaseous cargo parameters', 'un type code', numeric_measure)/

5.1.3.15.9 user_def_cargo

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.228
Reference path: product
{/CLASS_ID(product, 'gaseous cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘gaseous cargo parameters’, 'user def cargo')/

5.1.3.15.10 gaseous_cargo to dangerous_goods_code (as cargo_hazard)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'gaseous cargo')/}
product <-

product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, 'dangerous goods code')/}

5.1.3.15.11 gaseous_cargo to document_reference_with_address (as references)

AIM element: PATH
Reference path: product
/DOC_REF(product,'references')/
document
{/CLASS_ID(document, 'document reference with address')/}

5.1.3.15.12 gaseous_cargo to general_cargo_material_properties (as material_properties)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'gaseous cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{(/CLASS_ID(property_definition, 'general cargo material properties')/)
(/CLASS_ID(property_definition, 'detailed cargo material properties')/)}

5.1.3.16 GASEOUS_CARGO_ASSIGNMENT

AIM element: product_definition_relationship
Source: ISO 10303-41
Rules: 5.2.4.152
Reference path: {[/CLASS(product_definition_relationship, 'gaseous cargo assignment',
'compartment cargo assignment')/]
[/CLASS(product_definition_relationship, 'compartment cargo assignment', 'cargo assignment')/]
[/ROOT_CLASS(product_definition_relationship, 'cargo assignment')/]}

5.1.3.16.1 assignment_context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'gaseous cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition

characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘gaseous cargo assignment parameters’,
'assignment context')/
{(descriptive_representation_item.description = 'accomodation')
(descriptive_representation_item.description = 'cargo load')
(descriptive_representation_item.description = 'consumables')
(descriptive_representation_item.description = 'stores')
(descriptive_representation_item.description = 'unspecified')}

5.1.3.16.2 cargo_identifier

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'gaseous cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘gaseous cargo assignment parameters’,
'cargo identifier')/

5.1.3.16.3 gaseous_cargo_assignment to cargo (as cargo)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'gaseous cargo assignment')/}
product_definition_relationship.related_product_definition ->
product_definition
product_definition.formation ->
product_definition_formation
product_definition_formation.of_product ->
product
{/CLASS_ID(product, ‘gaseous cargo’)/}

5.1.3.16.4 gaseous_cargo_assignment to compartment (as compartment)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'gaseous cargo assignment')/}
product_definition_relationship.relating_product_definition ->
product_definition
{/CLASS_ID(product_definition, 'compartment')/}

5.1.3.16.5 gaseous_cargo_assignment to weight_and_centre_of_gravity (as allocated_weight)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'gaseous cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition <-

property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘allocated weight’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, ‘weight and centre of gravity’/}

5.1.3.17 GENERAL_CARGO_MATERIAL_PROPERTIES

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.203
Reference path: {/ROOT_CLASS(property_definition, 'general cargo material properties')/}

5.1.3.17.1 density

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('general cargo material properties parameters', 'density', 'density unit')/

5.1.3.17.2 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.3.17.3 general_cargo_material_properties to document_reference_with_address (as material_reference)

AIM element: PATH
Reference path: property_definition
/DOC_REF(property_definition,’material reference’)/
document
{/CLASS_ID(document, ‘document reference with address’)/}

5.1.3.18 LANE_POSITION

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, ’lane position’, ‘bay position’)/]
[/CLASS(property_definition, ’bay position’, ‘cargo position’)/]
[/ROOT_CLASS(property_definition, ’cargo position’)/]}

5.1.3.18.1 lane_position to cargo_bay_definition (as relating_to)

AIM element: PATH
Reference path: property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘relating to’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, ‘cargo bay definition’)/}

5.1.3.18.2 lane_position to longitudinal_position (as frame_number)

AIM element: PATH
Reference path: property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘relating to’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, ‘cargo bay definition’)/}
property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
{/CLASS_ID(product_definition, 'compartment')/}
product_definition <-
product_definition_relationship.relating_definition
product_definition_relationship
product_definition_relationship.related_definition ->
product_definition
product_definition = characterized_product_definition
characterized_product_definition = characterized_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, 'longitudinal table')/}
/PROP_DEF_TO_REP/
representation.items [i] ->
representation_item =>
compound_representation_item
{representation_item.name = 'frame number’}
{/CLASS_ID(compound_representation_item, ‘longitudinal position’)/}

5.1.3.18.3 lane_position to transversal_position (as lane_number)

AIM element: PATH
Reference path: property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘relating to’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, ‘cargo bay definition’)/}
property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
{/CLASS_ID(product_definition, 'compartment')/}
product_definition <-
product_definition_relationship.relating_definition
product_definition_relationship
product_definition_relationship.related_definition ->
product_definition
product_definition = characterized_product_definition

characterized_product_definition = characterized_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, 'transversal table')/}
/PROP_DEF_TO_REP/
representation.items [i] ->
representation_item =>
compound_representation_item
{representation_item.name = 'lane number'}
{/CLASS_ID(compound_representation_item, 'transversal position')/}

5.1.3.18.4 lane_position to vertical_position (as deck_number)

AIM element: PATH
Reference path: property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = 'relating to'}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, 'cargo bay definition')/}
property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
{/CLASS_ID(product_definition, 'compartment')/}
product_definition <-
product_definition_relationship.relating_definition
product_definition_relationship
product_definition_relationship.related_definition ->
product_definition
product_definition = characterized_product_definition
characterized_product_definition = characterized_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, 'vertical table')/}
/PROP_DEF_TO_REP/
representation.items [i] ->
representation_item =>
compound_representation_item
{representation_item.name = 'deck number'}
{/CLASS_ID(compound_representation_item, 'vertical position')/}

5.1.3.19 LIQUID_CARGO

AIM element: product
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.157, 5.2.4.204
Reference path: {/ROOT_CLASS(product, 'liquid cargo')/}

5.1.3.19.1 cargo_type

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.228
Reference path: product
{/CLASS_ID(product, 'liquid cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘liquid cargo parameters’, 'cargo type')/
{(descriptive_representation_item.description = 'alcohol')
(descriptive_representation_item.description = 'ammonia')
(descriptive_representation_item.description = 'asphalt')
(descriptive_representation_item.description = 'aviation oil')
(descriptive_representation_item.description = 'caustic soda')
(descriptive_representation_item.description = ‘cement')
(descriptive_representation_item.description = 'chemical')
(descriptive_representation_item.description = 'crude oil')
(descriptive_representation_item.description = 'edible oil')
(descriptive_representation_item.description = ‘fuel oil')
(descriptive_representation_item.description = 'fresh water')
(descriptive_representation_item.description = 'fruit juice')
(descriptive_representation_item.description = 'hydrocloric acid')
(descriptive_representation_item.description = ‘lubricating oil')
(descriptive_representation_item.description = ‘methanol')
(descriptive_representation_item.description = 'molasses')
(descriptive_representation_item.description = 'product oil')
(descriptive_representation_item.description = 'salt water')
(descriptive_representation_item.description = 'sullage')
(descriptive_representation_item.description = ‘sludge')
(descriptive_representation_item.description = 'sulphur')
(descriptive_representation_item.description = 'user defined')
(descriptive_representation_item.description = 'vegetable oil')
(descriptive_representation_item.description = 'water ballast')
(descriptive_representation_item.description = 'wine')}

5.1.3.19.2 description

AIM element: product.description
Source: ISO 10303-41

5.1.3.19.3 flash_point

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'liquid cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('liquid cargo parameters', 'flash point',
thermodynamic_temperature_measure)/

5.1.3.19.4 pollution_code

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product
{/CLASS_ID(product, 'liquid cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘liquid cargo parameters’, 'pollution code')/
{(descriptive_representation_item.description = 'code A')
(descriptive_representation_item.description = 'code B')
(descriptive_representation_item.description = 'code C')
(descriptive_representation_item.description = 'code D')}

5.1.3.19.5 required_carriage_pressure

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'liquid cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_SPECIAL_VAL_REP_ITEM('liquid cargo parameters',
'required carriage pressure', 'pressure unit')/

5.1.3.19.6 required_carriage_temperature

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'liquid cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('liquid cargo parameters', 'required carriage temperature', thermodynamic_temperature_measure)/

5.1.3.19.7 un_type_code

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'liquid cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('liquid cargo parameters', 'un type code', numeric_measure)/

5.1.3.19.8 user_def_cargo

AIM element: descriptive_representation_item.description

Source: ISO 10303-45
Rules: 5.2.4.228
Reference path: product
{/CLASS_ID(product, 'liquid cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘liquid cargo parameters’, 'user def cargo')/

5.1.3.19.9 liquid_cargo to dangerous_goods_code (as cargo_hazard)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'liquid cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, 'dangerous goods code')/}

5.1.3.19.10 liquid_cargo to document_reference_with_address (as references)

AIM element: PATH
Reference path: product
/DOC_REF(product,’references’)/
document
{/CLASS_ID(document, ‘document reference with address’)/}

5.1.3.19.11 liquid_cargo to general_cargo_material_properties (as material_properties)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'liquid cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{(/CLASS_ID(property_definition, 'general cargo material properties')/)
(/CLASS_ID(property_definition, 'detailed cargo material properties')/)}

5.1.3.20 LIQUID_CARGO_ASSIGNMENT

AIM element: product_definition_relationship
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.158

Reference path: {[/CLASS(product_definition_relationship, 'liquid cargo assignment', 'compartment cargo assignment')/]
[/CLASS(product_definition_relationship, 'compartment cargo assignment', 'cargo assignment')/]
[/ROOT_CLASS(product_definition_relationship, 'cargo assignment')/]}

5.1.3.20.1 assignment_context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'liquid cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘liquid cargo assignment parameters’, 'assignment context')/
{(descriptive_representation_item.description = 'accomodation')
(descriptive_representation_item.description = 'cargo load')
(descriptive_representation_item.description = 'consumables')
(descriptive_representation_item.description = 'stores')
(descriptive_representation_item.description = 'unspecified')}

5.1.3.20.2 cargo_height

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'liquid cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_VAL_REP_ITEM('liquid cargo assignment parameters', 'cargo height', length_measure)/

5.1.3.20.3 cargo_identifier

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'liquid cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘liquid cargo assignment parameters’, 'cargo identifier')/

5.1.3.20.4 liquid_cargo_assignment to cargo (as cargo)

AIM element: PATH
Reference path: product_definition_relationship

{/CLASS_ID(product_definition_relationship, 'liquid cargo assignment')/}
product_definition_relationship.related_product_definition ->
product_definition
product_definition.formation ->
product_definition_formation
product_definition_formation.of_product ->
product
{(/CLASS_ID(product, ‘gaseous cargo’)/)
(/CLASS_ID(product, ‘liquid cargo’)/)}

5.1.3.20.5 liquid_cargo_assignment to compartment (as compartment)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'liquid cargo assignment')/}
product_definition_relationship.relating_product_definition ->
product_definition
{/CLASS_ID(product_definition, 'compartment')/}

5.1.3.20.6 liquid_cargo_assignment to weight_and_centre_of_gravity (as allocated_weight)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'liquid cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘allocated weight’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, ‘weight and centre of gravity’/}

5.1.3.21 PERSON_GROUP

AIM element: [product]
[group]
Source: ISO 10303-41
ISO 10303-41
Rules: 5.2.4.183, 5.2.4.166
Reference path: {[/LINK_TO_GROUP(product)/]
[/CLASS(product, ‘person group’, ‘unit cargo group’)/]
[/ROOT_CLASS(product, 'unit cargo group')/]}

5.1.3.21.1 number_of_people

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'person group')/}
product <-
product_definition_formation.of_product

product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('person group parameters', 'number of people',
numeric_measure)/

5.1.3.21.2 person_type

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product
{/CLASS_ID(product, 'person group')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘person group parameters’, 'person type')/
{(descriptive_representation_item.description = 'crew')
(descriptive_representation_item.description = 'enlisted')
(descriptive_representation_item.description = 'officers')
(descriptive_representation_item.description = 'passengers')}

5.1.3.21.3 volume

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'person group')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('person group parameters', 'volume',
volume_measure)/

5.1.3.21.4 person_group to cargo_footprint (as footprint)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'person group')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, 'cargo footprint')/}

5.1.3.21.5 person_group to weight_and_centre_of_gravity (as weight_and_centre_of_gravity)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'person group')/}
product <-
product_definition_formation.of_product

product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, ‘weight and centre of gravity‘)/}

5.1.3.22 RELATIVE_CARGO_POSITION

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, ’relative cargo position’, ‘cargo position’)/]
[/ROOT_CLASS(property_definition, ’cargo position’)/]}

5.1.3.22.1 relative_cargo_position to longitudinal_position (as longitudinal_location)

AIM element: PATH
Reference path: property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, 'longitudinal table')/}
/PROP_DEF_TO_REP/
representation.items [i] ->
representation_item =>
compound_representation_item
{representation_item.name = 'longitudinal location’}
{/CLASS_ID(compound_representation_item, ‘longitudinal position’)/}

5.1.3.22.2 relative_cargo_position to transversal_position (as transverse_location)

AIM element: PATH
Reference path: property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, 'transversal table')/}
/PROP_DEF_TO_REP/
representation.items [i] ->
representation_item =>
compound_representation_item
{representation_item.name = 'transverse location’}
{/CLASS_ID(compound_representation_item, ‘transversal position’)/}

5.1.3.22.3 relative_cargo_position to vertical_position (as vertical_location)

AIM element: PATH
Reference path: property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
property_definition_relationship.related_property_definition ->

property_definition
{/CLASS_ID(property_definition, 'vertical table')/}
/PROP_DEF_TO_REP/
representation.items [i] ->
representation_item =>
compound_representation_item
{representation_item.name = 'vertical location'}
{/CLASS_ID(compound_representation_item, 'vertical position')/}

5.1.3.23 UNIT_CARGO

AIM element: product
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.176, 5.2.4.211
Reference path: {/ROOT_CLASS(product, 'unit cargo')/}

5.1.3.23.1 cargo_type

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.228
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM('unit cargo parameters', 'cargo type')/
{(descriptive_representation_item.description = 'aircraft')
(descriptive_representation_item.description = 'boat')
(descriptive_representation_item.description = 'cable')
(descriptive_representation_item.description = 'container')
(descriptive_representation_item.description = 'drums')
(descriptive_representation_item.description = 'pallet')
(descriptive_representation_item.description = 'trailer')
(descriptive_representation_item.description = 'undefined')
(descriptive_representation_item.description = 'user defined')
(descriptive_representation_item.description = 'vehicle')}

5.1.3.23.2 description

AIM element: product.description
Source: ISO 10303-41

5.1.3.23.3 flash_point

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('unit cargo parameters', 'flash point',
thermodynamic_temperature_measure)/

5.1.3.23.4 lashing_points

AIM element: cartesian_point
Source: ISO 10303-42
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition = represented_definition
represented_definition <-
/PDR_NAME('unit cargo parameters')/ ->
representation
representation.items[i] ->
/GEO_REP_ITEM('lashing point', cartesian_point)/

5.1.3.23.5 permeability

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('unit cargo parameters', 'permeability',
ratio_measure)/

5.1.3.23.6 pollution_code

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘unit cargo parameters’, 'pollution code')/
{(descriptive_representation_item.description = 'code A')
(descriptive_representation_item.description = 'code B')
(descriptive_representation_item.description = 'code C')
(descriptive_representation_item.description = 'code D')}

5.1.3.23.7 required_carriage_temperature

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}

product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('unit cargo parameters', 'required carriage temperature', thermodynamic_temperature_measure)/

5.1.3.23.8 stack_limit

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('unit cargo parameters', 'stack limit', numeric_measure)/

5.1.3.23.9 stowage_factor

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('unit cargo parameters', 'stowage factor', volume_measure)/

5.1.3.23.10 un_type_code

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('unit cargo parameters', 'un type code', numeric_measure)/

5.1.3.23.11 user_def_cargo

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.228
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_DESC_REP_ITEM(‘unit cargo parameters’, 'user def cargo')/

5.1.3.23.12 volume

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('unit cargo parameters', 'volume',
volume_measure)/

5.1.3.23.13 unit_cargo to cargo_footprint (as footprints)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, 'cargo footprint')/}

5.1.3.23.14 unit_cargo to dangerous_goods_code (as cargo_hazard)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, 'dangerous goods code')/}

5.1.3.23.15 unit_cargo to document_reference_with_address (as references)

AIM element: PATH
Reference path: product
/DOC_REF(product,’references’)/
document
{/CLASS_ID(document, ‘document reference with address’)/}

5.1.3.23.16 unit_cargo to general_cargo_material_properties (as material_properties)

AIM element: PATH

Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{(/CLASS_ID(property_definition, 'general cargo material properties')/)
(/CLASS_ID(property_definition, 'detailed cargo material properties')/)}

5.1.3.23.17 unit_cargo to shape_representation (as bounding_space)

AIM element: PATH
Rules: 5.2.4.218
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition =>
product_definition_shape
{product_definition_shape.name = ‘bounding space’}
represented_definition = product_definition_shape
represented_definition <-
property_definition_representation.definition
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘unit cargo bounding space’)/}
property_definition_representation.used_representation ->
representation =>
shape_representation =>
non_manifold_surface_shape_representation

5.1.3.23.18 unit_cargo to shape_representation (as shape_description)

AIM element: PATH
Rules: 5.2.4.218
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition

characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition =>
product_definition_shape
{product_definition_shape.name = ‘shape description’}
represented_definition = product_definition_shape
represented_definition <-
property_definition_representation.definition
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘unit cargo shape’)/}
property_definition_representation.used_representation ->
representation =>
shape_representation =>
non_manifold_surface_shape_representation

5.1.3.23.19 unit_cargo to unit_cargo_bounding_box (as bounding_space)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition =>
product_definition_shape
{product_definition_shape.name = ‘bounding space’}
represented_definition = product_definition_shape
represented_definition <-
property_definition_representation.definition
property_definition_representation
{/CLASS_ID(property_definition_representation, 'unit cargo bounding box')/}

5.1.3.23.20 unit_cargo to weight_and_centre_of_gravity (as weight_and_centre_of_gravity)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'unit cargo')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, ‘weight and centre of gravity‘)/}

5.1.3.24 UNIT_CARGO_ASSIGNMENT

AIM element: product_definition_relationship
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.177
Reference path: {[/CLASS(product_definition_relationship, ‘unit cargo assignment', 'cargo assignment')/]
[/ROOT_CLASS(product_definition_relationship, 'cargo assignment')/]}

5.1.3.24.1 assignment_context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'unit cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘unit cargo assignment parameters’, 'assignment context')/
{(descriptive_representation_item.description = 'accomodation')
(descriptive_representation_item.description = 'cargo load')
(descriptive_representation_item.description = 'consumables')
(descriptive_representation_item.description = 'stores')
(descriptive_representation_item.description = 'unspecified')}

5.1.3.24.2 cargo_identifier

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'unit cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
/PROP_DEF_TO_DESC_REP_ITEM(‘unit cargo assignment parameters’, 'cargo identifier')/

5.1.3.24.3 unit_cargo_assignment to cargo (as cargo)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'unit cargo assignment')/}
product_definition_relationship.related_product_definition ->
product_definition
product_definition.formation ->
product_definition_formation
product_definition_formation.of_product ->
product
{(/CLASS_ID(product, ‘unit cargo’)/)
(/CLASS_ID(product, ‘unit cargo group’)/)

(/CLASS_ID(product, ‘person group’)/)}

5.1.3.24.4 unit_cargo_assignment to cargo_position (as position)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'unit cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘position’}
property_definition_relationship.related_property_definition ->
property_definition
{(/CLASS_ID(property_definition, ‘absolute cargo position’/)
(/CLASS_ID(property_definition, ‘relative cargo position’/)
(/CLASS_ID(property_definition, ‘bay position’/)}

5.1.3.24.5 unit_cargo_assignment to compartment (as assigned_to)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'unit cargo assignment')/}
product_definition_relationship.relating_product_definition ->
product_definition
{/CLASS_ID(product_definition, 'compartment')/}

5.1.3.24.6 unit_cargo_assignment to deck_zone (as assigned_to)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'unit cargo assignment')/}
product_definition_relationship.relating_product_definition ->
product_definition
{/CLASS_ID(product_definition, 'deck zone')/}

5.1.3.24.7 unit_cargo_assignment to weight_and_centre_of_gravity (as allocated_weight)

AIM element: PATH
Reference path: product_definition_relationship
{/CLASS_ID(product_definition_relationship, 'unit cargo assignment')/}
characterized_product_definition = product_definition_relationship
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘allocated weight’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, ‘weight and centre of gravity’/}

5.1.3.25 UNIT_CARGO_BOUNDING_BOX

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.178
Reference path: {/CLASS_ID(property_definition_representation, 'unit cargo bounding box')/}

5.1.3.25.1 point_max

AIM element: cartesian_point
Source: ISO 10303-42
Reference path: property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, 'unit cargo bounding box')/}
property_definition_representation.used_representation ->
representation
representation.items[i] ->
/GEO_REP_ITEM(‘point max', cartesian_point)/

5.1.3.25.2 point_min

AIM element: cartesian_point
Source: ISO 10303-42
Reference path: property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, 'unit cargo bounding box')/}
property_definition_representation.used_representation ->
representation
representation.items[i] ->
/GEO_REP_ITEM(‘point min', cartesian_point)/

5.1.3.26 UNIT_CARGO_GROUP

AIM element: [product]
[group]
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.179
Reference path: {[/LINK_TO_GROUP(product)/]
[/ROOT_CLASS(product, 'unit cargo group')/]}

5.1.3.26.1 volume

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: product
{/CLASS_ID(product, 'unit cargo group')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
/PROD_DEF_TO_VAL_REP_ITEM('unit cargo group parameters', 'volume', volume_measure)/

5.1.3.26.2 unit_cargo_group to cargo_footprint (as footprint)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'unit cargo group')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, 'cargo footprint')/}

5.1.3.26.3 unit_cargo_group to unit_cargo (as cargo)

AIM element: PATH
Reference path: group <-
/GROUPS(product, 'unit cargo group')/
{/CLASS_ID(product, 'unit cargo')/}

5.1.3.26.4 unit_cargo_group to weight_and_centre_of_gravity (as weight_and_centre_of_gravity)

AIM element: PATH
Reference path: product
{/CLASS_ID(product, 'unit cargo group')/}
product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition
characterized_product_definition = product_definition
characterized_definition = characterized_product_definition
characterized_definition <-
property_definition.definition
property_definition
{/CLASS_ID(property_definition, ‘weight and centre of gravity‘)/}

5.1.4 Coatings UoF

5.1.4.1 COATING

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.113
Reference path: {/ROOT_CLASS(property_definition, 'coating')/}

5.1.4.1.1 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.4.1.2 dry_film_thickness

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('coating parameters', 'dry film thickness', positive_length_measure)/

5.1.4.1.3 manufacturer

AIM element: applied_organization_assignment.assigned_organization
Source: ISO 10303-215
Reference path: /ORG_ASSGN(property_definition, ‘coating manufacturer’)/

5.1.4.1.4 name

AIM element: property_definition.name
Source: ISO 10303-41

5.1.4.1.5 number_of_coats

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('coating parameters', 'number of coats', numeric_measure)/

5.1.4.1.6 surface_preparation

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘coating parameters’, 'surface preparation')/
{(descriptive_representation_item.description = 'Fl')
(descriptive_representation_item.description = ' Sa1')
(descriptive_representation_item.description = 'Sa2')
(descriptive_representation_item.description = ' Sa2.5')
(descriptive_representation_item.description = 'Sa3')
(descriptive_representation_item.description = 'St2')
(descriptive_representation_item.description = 'St3')}

5.1.4.1.7 coating to coating_certification (as certification)

AIM element: PATH
Reference path: property_definition
{/CLASS_ID(property_definition, 'coating')/}
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘certification’}
property_definition_relationship.related_property_definition ->
property_definition
property_definition.definition ->
characterized_definition
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
product_definition

{/CLASS_ID(product_definition, 'coating certification')/}

5.1.4.2 COATING_CERTIFICATION

AIM element: product_definition
Source: ISO 10303-41
Reference path: {/ROOT_CLASS(product_definition, ‘coating certification’)/}

5.1.4.2.1 certifying_organization

AIM element: organization
Source: ISO 10303-41
Rules: 5.2.4.54
Reference path: /ORG_ASSGN(product_definition, ‘certifying organization‘)/

5.1.4.2.2 expiry_date

AIM element: date_and_time
Source: ISO 10303-41
Rules: 5.2.4.56
Reference path: /DAT_TIME_ASSGN(product_definition, ‘expiry date’)/

5.1.4.3 CORROSION_CONTROL_COATING

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.141, 5.2.4.197
Reference path: {[/CLASS(property_definition, 'corrosion control coating', 'coating')/]
[/ROOT_CLASS(property_definition, 'coating')/]}

5.1.4.3.1 applicability

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘corrosion control coating parameters’, 'applicability')/
{(descriptive_representation_item.description = 'B')
(descriptive_representation_item.description = ' C')
(descriptive_representation_item.description = 'RS')
(descriptive_representation_item.description = ' V')}

5.1.4.3.2 description

AIM element: /SUPERTYPE(coating)/ -- (see 5.1.4.1.1)

5.1.4.3.3 dry_film_thickness

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('corrosion control coating parameters', 'dry film thickness', length_measure)/

5.1.4.3.4 manufacturer

AIM element: applied_organization_assignment.assigned_organization
Source: ISO 10303-215

Reference path: /ORG_ASSGN(property_definition, ‘corrosion control coating manufacturer’)/

5.1.4.3.5 name

AIM element: /SUPERTYPE(coating)/ -- (see 5.1.4.1.4)

5.1.4.3.6 number_of_coats

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('corrosion control coating parameters', 'number of coats', numeric_measure)/

5.1.4.3.7 surface_preparation

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘corrosion control coating parameters’, 'surface preparation')/
{(descriptive_representation_item.description = 'Fl')
(descriptive_representation_item.description = ' Sa1')
(descriptive_representation_item.description = 'Sa2')
(descriptive_representation_item.description = ' Sa2.5')
(descriptive_representation_item.description = 'Sa3')
(descriptive_representation_item.description = 'St2')
(descriptive_representation_item.description = 'St3')}

5.1.4.3.8 type_of

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.233
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘corrosion control coating parameters’, 'type of')/
{(descriptive_representation_item.description = 'aluminium')
(descriptive_representation_item.description = 'bituminous')
(descriptive_representation_item.description = 'chlorinated_rubber')
(descriptive_representation_item.description = 'coal_tar')
(descriptive_representation_item.description = 'epoxy')
(descriptive_representation_item.description = 'glassflake')
(descriptive_representation_item.description = 'isocynate')
(descriptive_representation_item.description = 'micaceous_iron_oxide')
(descriptive_representation_item.description = 'non_oxidising')
(descriptive_representation_item.description = 'phenolic')
(descriptive_representation_item.description = 'pitch')
(descriptive_representation_item.description = 'polyester')
(descriptive_representation_item.description = 'polyurethane')
(descriptive_representation_item.description = 'tar')
(descriptive_representation_item.description = 'vinyl')
(descriptive_representation_item.description = 'water_based')
(descriptive_representation_item.description = 'zinc_rich')
(descriptive_representation_item.description = 'zinc_silicate')
(descriptive_representation_item.description = 'user_defined')}

5.1.4.3.9 user_defined_type

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘corrosion control coating parameters’, 'user defined type')/

5.1.4.3.10 corrosion_control_coating to coating_certification (as certification)

AIM element: PATH
Reference path: property_definition
{/CLASS_ID(property_definition, 'corrosion control coating')/}
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘certification’}
property_definition_relationship.related_property_definition ->
property_definition
property_definition.definition ->
characterized_definition
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
product_definition
{/CLASS_ID(product_definition, 'coating certification')/}

5.1.4.3.11 corrosion_control_coating to primer_coating (as primer)

AIM element: PATH
Reference path: property_definition
{/CLASS_ID(property_definition, 'corrosion control coating')/}
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘primer’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, 'primer coating')/}

5.1.4.4 FIRE_SAFE_COATING

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.149
Reference path: {[/CLASS(property_definition, 'fire safe coating', 'coating')/]
[/ROOT_CLASS(property_definition, 'coating')/]}

5.1.4.4.1 description

AIM element: /SUPERTYPE(coating)/ -- (see 5.1.4.1.1)

5.1.4.4.2 dry_film_thickness

AIM element: value_representation_item.value_component
Source: ISO 10303-43

Reference path: /PROP_DEF_TO_VAL_REP_ITEM('fire safe coating parameters', 'dry film thickness', length_measure)/

5.1.4.4.3 low_flame_spread

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘fire safe coating parameters’, 'low flame spread')/
{(descriptive_representation_item.description = 'TRUE')
(descriptive_representation_item.description = 'FALSE')}

5.1.4.4.4 manufacturer

AIM element: applied_organization_assignment.assigned_organization
Source: ISO 10303-215
Reference path: /ORG_ASSGN(property_definition, ‘fire safe coating manufacturer’)/

5.1.4.4.5 name

AIM element: /SUPERTYPE(coating)/ -- (see 5.1.4.1.4)

5.1.4.4.6 nitro_cellulose_based

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘fire safe coating parameters’, 'nitro cellulose based')/
{(descriptive_representation_item.description = 'TRUE')
(descriptive_representation_item.description = 'FALSE')}

5.1.4.4.7 number_of_coats

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('fire safe coating parameters', 'number of coats', numeric_measure)/

5.1.4.4.8 surface_preparation

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘fire safe coating parameters’, 'surface preparation')/
{(descriptive_representation_item.description = 'Fl')
(descriptive_representation_item.description = ' Sa1')
(descriptive_representation_item.description = 'Sa2')
(descriptive_representation_item.description = ' Sa2.5')
(descriptive_representation_item.description = 'Sa3')
(descriptive_representation_item.description = 'St2')
(descriptive_representation_item.description = 'St3')}

5.1.4.4.9 fire_safe_coating to coating_certification (as certification)

AIM element: PATH
Reference path: property_definition
{/CLASS_ID(property_definition, 'fire safe coating')/}
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘certification’}
property_definition_relationship.related_property_definition ->
property_definition
property_definition.definition ->
characterized_definition
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
product_definition
{/CLASS_ID(product_definition, 'coating certification')/}

5.1.4.4.10 fire_safe_coating to primer_coating (as primer)

AIM element: PATH
Reference path: property_definition
{/CLASS_ID(property_definition, 'fire safe coating')/}
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘primer’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, 'primer coating')/}

5.1.4.5 PRIMER_COATING

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.167
Reference path: {[/CLASS(property_definition, 'primer coating', 'coating')/]
[/ROOT_CLASS(property_definition, 'coating')/]}

5.1.4.5.1 description

AIM element: /SUPERTYPE(coating)/ -- (see 5.1.4.1.1)

5.1.4.5.2 dry_film_thickness

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('primer coating parameters', 'dry film thickness', length_measure)/

5.1.4.5.3 manufacturer

AIM element: applied_organization_assignment.assigned_organization
Source: ISO 10303-215
Reference path: /ORG_ASSGN(property_definition, ‘primer coating manufacturer’)/

5.1.4.5.4 name

AIM element: /SUPERTYPE(coating)/ -- (see 5.1.4.1.4)

5.1.4.5.5 number_of_coats

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('primer coating parameters', 'number of coats', numeric_measure)/

5.1.4.5.6 surface_preparation

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘primer coating parameters’, 'surface preparation')/
{(descriptive_representation_item.description = 'Fl')
(descriptive_representation_item.description = ' Sa1')
(descriptive_representation_item.description = 'Sa2')
(descriptive_representation_item.description = ' Sa2.5')
(descriptive_representation_item.description = 'Sa3')
(descriptive_representation_item.description = 'St2')
(descriptive_representation_item.description = 'St3')}

5.1.4.5.7 primer_coating to coating_certification (as certification)

AIM element: PATH
Reference path: property_definition
{/CLASS_ID(property_definition, 'primer coating')/}
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘certification’}
property_definition_relationship.related_property_definition ->
property_definition
property_definition.definition ->
characterized_definition
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
product_definition
{/CLASS_ID(product_definition, 'coating certification')/}

5.1.5 Compartment_design_definitions UoF

5.1.5.1 CARGO_BAY_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, 'cargo bay definition', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.5.1.1 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.5.1.2 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition_shape)/

5.1.5.1.3 cargo_bay_definition to compartment (as defined_for)

AIM element: PATH
Reference path: property_definition
{/CLASS_ID(property_definition, 'cargo bay definition')/}
property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition =
product_definition
{/CLASS_ID(product_definition, 'compartment')/}

5.1.5.1.4 cargo_bay_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.5.1.5 cargo_bay_definition to spacing_table (as cargo_positions)

AIM element: PATH
Reference path: property_definition
{/CLASS_ID(property_definition, 'cargo bay definition')/}
property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
{/CLASS_ID(product_definition, 'compartment')/}
product_definition <-
product_definition_relationship.relating_definition
product_definition_relationship
product_definition_relationship.related_definition ->
product_definition
product_definition = characterized_product_definition
characterized_product_definition = characterized_definition
characterized_definition <-
property_definition.definition
property_definition
{(/CLASS_ID(property_definition, 'longitudinal table')/)
(/CLASS_ID(property_definition, 'transversal table')/)
(/CLASS_ID(property_definition, 'vertical table')/)}

5.1.5.1.6 cargo_bay_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.5.1.7 cargo_bay_definition to named_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.5.2 COMPARTMENT_DESIGN_DEFINITION

AIM element: product_definition_shape
Source: ISO 10303-41
Reference path: {[/CLASS(product_definition_shape, 'compartment design definition',
'design definition')/]
[/CLASS(product_definition_shape, 'design definition', 'definition')/]
[/CLASS(product_definition_shape, 'definition', 'versionable object')/]
[/ROOT_CLASS(product_definition_shape, 'versionable object')/]}

5.1.5.2.1 description

AIM element: product_definition_shape.description
Source: ISO 10303-41

5.1.5.2.2 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition_shape)/

5.1.5.2.3 compartment_design_definition to compartment (as defined_for)

AIM element: PATH
Reference path: product_definition_shape <=
property_definition
property_definition.definition ->
characterized_definition =
characterized_product_definition =
product_definition
{/CLASS_ID(product_definition, 'compartment')/}

5.1.5.2.4 compartment_design_definition to compartment_property (as properties)

AIM element: PATH
Reference path: product_definition_shape <=
property_definition =

represented_definition <-
/PDR_NAME('compartment property')/

5.1.5.2.5 compartment_design_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: product_definition_shape <=
/PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.5.2.6 compartment_design_definition to external_instance_reference (as boundaries)

AIM element: PATH
Reference path: product_definition_shape <-
shape_aspect.of_shape
shape_aspect
{[shape_aspect.name = 'boundary']
[/CLASS_ID(shape_aspect, 'boundary')/]
[(/EXT_INST_REF(shape_aspect, 'ship moulded form schema', 'moulded form')/)
(/EXT_INST_REF(product_definition, 'ship structures shema', 'structural system')/)]}

5.1.5.2.7 compartment_design_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.68
Reference path: product_definition_shape
identification_item = product_definition_shape <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.5.2.8 compartment_design_definition to named_unit (as local_units)

AIM element: PATH
Reference path: product_definition_shape <=
/PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.5.2.9 compartment_design_definition to non_manifold_surface_shape (as representations)

AIM element: PATH
Rules: 5.2.4.218
Reference path: product_definition_shape <=
property_definition <-
represented_definition <-
/SDR_NAME('compartment design representation')/
property_definition_representation.used_representation ->
representation =>
shape_representation =>
non_manifold_surface_shape_representation

5.1.5.3 DECK_ZONE_DESIGN_DEFINITION

AIM element: product_definition_shape
Source: ISO 10303-41
Rules: 5.2.4.67
Reference path: {[/CLASS(product_definition_shape, 'deck zone design definition', 'design definition')/]
[/CLASS(product_definition_shape, 'design definition', 'definition')/]
[/CLASS(product_definition_shape, 'definition', 'versionable object')/]
[/ROOT_CLASS(product_definition_shape, 'versionable object')/]}

5.1.5.3.1 description

AIM element: product_definition_shape.description
Source: ISO 10303-41

5.1.5.3.2 inner_boundaries

AIM element: bounded_curve
Source: ISO 10303-42
Reference path: product_definition_shape <-
shape_aspect.of_shape
shape_aspect
{[/ROOT_CLASS(shape_aspect, 'inner boundaries')/]}
shape_aspect =
represented_definition <-
/PDR_NAME('deck zone design parameters')/ ->
representation
{representation.name = 'inner boundaries representation'}
representation.items[i] ->
representation_item
{representation_item.name = 'inner boundary'} =>
geometric_representation_item =>
bounded_curve

5.1.5.3.3 outer_boundary

AIM element: bounded_curve
Source: ISO 10303-42
Reference path: product_definition_shape <-
shape_aspect.of_shape
shape_aspect
{[/ROOT_CLASS(shape_aspect, ‘outer boundary')/]}
shape_aspect =
represented_definition <-
/PDR_NAME('deck zone design parameters')/ ->
representation
{representation.name = 'outer boundary representation'}
representation.items[i] ->
representation_item
{representation_item.name = 'outer boundary'} =>
geometric_representation_item =>
bounded_curve

5.1.5.3.4 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition_shape)/

5.1.5.3.5 deck_zone_design_definition to compartment (as constituent_compartments)

AIM element: PATH
Reference path: product_definition_shape
{/CLASS_ID(product_definition_shape, 'deck zone design definition')/}
product_definition_shape <=
property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition =
product_definition
{/CLASS_ID(product_definition, 'deck zone')/}
product_definition <-
product_definition_relationship.relating_product_definition
product_definition_relationship
product_definition_relationship.related_product_definition ->
product_definition
{/CLASS_ID(product_definition, 'compartment')/}

5.1.5.3.6 deck_zone_design_definition to compartment_property (as properties)

AIM element: PATH
Reference path: product_definition_shape <=
property_definition =
represented_definition <-
/PDR_NAME('compartment property')/

5.1.5.3.7 deck_zone_design_definition to deck_zone (as defined_for)

AIM element: PATH
Reference path: product_definition_shape <=
property_definition
property_definition.definition ->
characterized_definition =
characterized_product_definition =
product_definition
{/CLASS_ID(product_definition, 'deck zone')/}

5.1.5.3.8 deck_zone_design_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: product_definition_shape <=
/PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.5.3.9 deck_zone_design_definition to external_instance_reference (as constituent_compartments)

AIM element: PATH
Reference path: product_definition_shape <-
shape_aspect.of_shape
shape_aspect
{[shape_aspect.name = 'constituent compartment']
[/CLASS_ID(shape_aspect, 'constituent compartment')/]
[/EXT_INST_REF(product_definition, 'ship arrangement schema',
'compartment')/]}

5.1.5.3.10 deck_zone_design_definition to external_instance_reference (as deck_for_zone)

AIM element: PATH
Reference path: product_definition_shape <-
shape_aspect.of_shape
shape_aspect
{[shape_aspect.name = 'deck for zone']
[/CLASS_ID(shape_aspect, 'deck for zone')/]
([/EXT_INST_REF(shape_aspect, 'ship moulded form schema', 'moulded form')/])
([/EXT_INST_REF(product_definition, 'ship structures schema', 'structural
system')/])}

5.1.5.3.11 deck_zone_design_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.68
Reference path: product_definition_shape
identification_item = product_definition_shape <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.5.3.12 deck_zone_design_definition to named_unit (as local_units)

AIM element: PATH
Reference path: product_definition_shape <=
/PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.5.3.13 deck_zone_design_definition to non_manifold_surface_shape (as representations)

AIM element: PATH
Rules: 5.2.4.218
Reference path: product_definition_shape <=
property_definition <-
represented_definition <-
/SDR_NAME('deck zone design representation')/
property_definition_representation.used_representation ->
representation =>
shape_representation =>
non_manifold_surface_shape_representation

5.1.5.4 ZONE_DESIGN_DEFINITION

AIM element: product_definition_shape
Source: ISO 10303-41
Reference path: {[/CLASS(product_definition_shape, 'zone design definition', 'design definition')/]
[/CLASS(product_definition_shape, 'design definition', 'definition')/]
[/CLASS(product_definition_shape, 'definition', 'versionable object')/]
[/ROOT_CLASS(product_definition_shape, 'versionable object')/]}

5.1.5.4.1 description

AIM element: product_definition_shape.description
Source: ISO 10303-41

5.1.5.4.2 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition_shape)/

5.1.5.4.3 zone_design_definition to compartment (as constituent_compartments)

AIM element: PATH
Reference path: product_definition_shape
{/CLASS_ID(product_definition_shape, 'zone design definition')/}
product_definition_shape <=
property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition =
product_definition
{/CLASS_ID(product_definition, 'zone')/}
product_definition <-
product_definition_relationship.relating_product_definition
product_definition_relationship
product_definition_relationship.related_product_definition ->
product_definition
{/CLASS_ID(product_definition, 'compartment')/}:

5.1.5.4.4 zone_design_definition to compartment_property (as properties)

AIM element: PATH
Reference path: product_definition_shape <=
property_definition =
represented_definition <-
/PDR_NAME('compartment property')/

5.1.5.4.5 zone_design_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: product_definition_shape <=
/PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.5.4.6 zone_design_definition to external_instance_reference (as boundaries)

AIM element: PATH
Reference path: product_definition_shape <-
shape_aspect.of_shape
shape_aspect
{[shape_aspect.name = 'boundary']
[/CLASS_ID(shape_aspect, 'boundary')/]
[(/EXT_INST_REF(shape_aspect, 'ship moulded form schema', 'moulded form')/)
(/EXT_INST_REF(product_definition, 'ship structures shema', 'structural system')/)]}

5.1.5.4.7 zone_design_definition to external_instance_reference (as constituent_compartments)

AIM element: PATH
Reference path: product_definition_shape <-
shape_aspect.of_shape
shape_aspect
{[shape_aspect.name = 'constituent compartment']
[/CLASS_ID(shape_aspect, 'constituent compartment')/]
[/EXT_INST_REF(product_definition, 'ship arrangement schema', 'compartment')/]}

5.1.5.4.8 zone_design_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.68
Reference path: product_definition_shape
identification_item = product_definition_shape <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.5.4.9 zone_design_definition to named_unit (as local_units)

AIM element: PATH
Reference path: product_definition_shape <=
/PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.5.4.10 zone_design_definition to non_manifold_surface_shape (as representations)

AIM element: PATH
Rules: 5.2.4.218
Reference path: product_definition_shape <=
property_definition <-
represented_definition <-
/SDR_NAME('zone design representation')/
property_definition_representation.used_representation ->
representation =>
shape_representation =>
non_manifold_surface_shape_representation

5.1.5.4.11 zone_design_definition to zone (as defined_for)

AIM element: PATH
Reference path: product_definition_shape <=
property_definition
property_definition.definition ->
characterized_definition =
characterized_product_definition =
product_definition
{/CLASS_ID(product_definition, 'zone')/}

5.1.6 Compartment_properties UoF

5.1.6.1 CAPACITY_PROPERTIES

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.104, 5.2.4.186
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘capacity properties’)/}

5.1.6.1.1 capacity_context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.231
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'capacity context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'full 95 percent')
(descriptive_representation_item.description = 'full 98 percent')
(descriptive_representation_item.description = 'pressed full')
(descriptive_representation_item.description = 'slack')
(descriptive_representation_item.description = 'user defined')}

5.1.6.1.2 capacity_heel_angle

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('capacity heel angle', plane_angle_measure)/

5.1.6.1.3 capacity_level

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('capacity level', length_measure)/

5.1.6.1.4 capacity_trim_angle

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('capacity trim angle', plane_angle_measure)/

5.1.6.1.5 capacity_volume

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('capacity volume', volume_measure)/

5.1.6.1.6 user_defined_capacity_context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.231
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'user defined capacity context'}
representation_item =>
descriptive_representation_item
descriptive_representation_item.description

5.1.6.1.7 capacity_properties to to centre_location (as capacity_centre)

AIM element: PATH
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items [i] ->
representation_item
{representation_item.name = 'capacity centre'} =>
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘centre location’)/}

5.1.6.1.8 capacity_properties to centre_location (as capacity_level_origin)

AIM element: PATH
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items [i] ->
representation_item
{representation_item.name = 'capacity level origin'} =>
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘centre location’)/}

5.1.6.2 CARGO_COMPARTMENT_PROPERTY

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.105
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘cargo compartment property’)/}

5.1.6.2.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.2.2 design_stowage_density

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_SPECIAL_VAL_REP_ITEM('design stowage density', 'density unit')/

5.1.6.2.3 cargo_compartment_property to capacity_properties (as bulk_cargo_capacity)

AIM element: PATH
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation <-
representation_relationship.rep_1
representation_relationship
{representation_relationship.name = ‘bulk_cargo_capacity’}
representation_relationship
representation_relationship.rep_2 ->
representation <-
property_definition_representation.used_representation ->
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘capacity properties’)/}

5.1.6.3 COATING_LEVEL

AIM element: property_definition_representation
Source: ISO 10303-41

Rules: 5.2.4.183, 5.2.4.114
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘coating level’)/}

5.1.6.3.1 lower_extent

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('lower extent', ratio_measure)/

5.1.6.3.2 upper_extent

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('upper extent', ratio_measure)/

5.1.6.4 COMPARTMENT_ABBREVIATED_NAME

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.115
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment abbreviated name’)/}

5.1.6.4.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.4.2 name

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->

representation_item
{representation_item.name = 'name'}
representation_item =>
descriptive_representation_item
descriptive_representation_item.description

5.1.6.5 COMPARTMENT_ACCELERATION

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.116
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment acceleration’)/}

5.1.6.5.1 acceleration_g_force

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('acceleration g force', numeric_measure)/

5.1.6.5.2 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.6 COMPARTMENT_ACCESS_AUTHORIZATION

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.117, 5.2.4.189
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment access authorization’)/}

5.1.6.6.1 authorization_classification

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.234
Reference path: property_definition_representation
property_definition_representation.used_representation ->

representation
representation.items[i] ->
representation_item
{representation_item.name = 'authorization classification'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'crew only')
(descriptive_representation_item.description = 'officers only')
(descriptive_representation_item.description = 'restricted')
(descriptive_representation_item.description = 'unrestricted')
(descriptive_representation_item.description = 'user defined')}

5.1.6.6.2 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.6.3 user_defined_value

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'user defined value'}
representation_item =>
descriptive_representation_item
descriptive_representation_item.description

5.1.6.7 COMPARTMENT_AIR_CIRCULATION_RATE

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.118
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment air circulation rate’)/}

5.1.6.7.1 air_circulation_rate

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_SPECIAL_VAL_REP_ITEM('air circulation rate', 'airflow volume unit')/

5.1.6.7.2 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item => descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.8 COMPARTMENT_AREA_PROPERTY

AIM element: /SUBTYPE(compartment_vertical_longitudinal_cross_sectional_area_property)/
-- (see 5.1.6.23)
/SUBTYPE(compartment_vertical_transverse_cross_sectional_area_property)/
-- (see 5.1.6.24)
/SUBTYPE(compartment_horizontal_cross_sectional_area_property)/
-- (see 5.1.6.10)
/SUBTYPE(compartment_unstiffened_surface_area_property)/
-- (see 5.1.6.22)
/SUBTYPE(compartment_stiffened_surface_area_property)/
-- (see 5.1.6.20)

5.1.6.8.1 context

AIM element: /SUBTYPE(compartment_vertical_longitudinal_cross_sectional_area_property)/
-- (see 5.1.6.23.1)
/SUBTYPE(compartment_vertical_transverse_cross_sectional_area_property)/
-- (see 5.1.6.24.1)
/SUBTYPE(compartment_horizontal_cross_sectional_area_property)/
-- (see 5.1.6.10.1)
/SUBTYPE(compartment_unstiffened_surface_area_property)/
-- (see 5.1.6.22.1)
/SUBTYPE(compartment_stiffened_surface_area_property)/
-- (see 5.1.6.20.1)

5.1.6.9 COMPARTMENT_COATING

AIM element: property_definition_representation

Source: ISO 10303-41
Rules: 5.2.4.120
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment coating’)/}

5.1.6.9.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.9.2 compartment_coating to corrosion_protection (as corrosion_protection)

AIM element: PATH
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation <-
representation_relationship.rep_1
representation_relationship
{representation_relationship.name = ‘corrosion protection’}
representation_relationship
representation_relationship.rep_2 ->
representation <-
property_definition_representation.used_representation ->
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘corrosion protection’)/}

5.1.6.10 COMPARTMENT_HORIZONTAL_CROSS_SECTIONAL_AREA_PROPERTY

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.124
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment horizontal cross sectional area property’)/}

5.1.6.10.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->

representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.10.2 horizontal_cross_sectional_area

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('horizontal cross sectional area', area_measure)/

5.1.6.11 COMPARTMENT_ILLUMINATION

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.125
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment illumination’)/}

5.1.6.11.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.11.2 illumination_value

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('illumination value', luminous_intensity_measure)/

5.1.6.12 COMPARTMENT_INSULATION

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.126, 5.2.4.192
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment insulation’)/}

5.1.6.12.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.12.2 insulation_category

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.235
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'insulation category'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'A')
(descriptive_representation_item.description = 'B')
(descriptive_representation_item.description = 'C')
(descriptive_representation_item.description = 'D')
(descriptive_representation_item.description = 'E')
(descriptive_representation_item.description = 'F')
(descriptive_representation_item.description = 'G')
(descriptive_representation_item.description = 'H')
(descriptive_representation_item.description = 'I')
(descriptive_representation_item.description = 'J')
(descriptive_representation_item.description = 'K')
(descriptive_representation_item.description = 'L')
(descriptive_representation_item.description = 'M')
(descriptive_representation_item.description = 'N')

(descriptive_representation_item.description = 'O')
(descriptive_representation_item.description = 'P')
(descriptive_representation_item.description = 'Q')
(descriptive_representation_item.description = 'R')
(descriptive_representation_item.description = 'user defined')}

5.1.6.12.3 user_defined_value

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.235
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'user defined value'}
representation_item =>
descriptive_representation_item
descriptive_representation_item.description

5.1.6.13 COMPARTMENT_NAVAL_ADMINISTRATIVE_PROPERTY

AIM element: /SUBTYPE(compartment_nuclear_classification)/ -- (see 5.1.6.15)
/SUBTYPE(compartment_safety_class)/ -- (see 5.1.6.18)
/SUBTYPE(compartment_security_classification)/ -- (see 5.1.6.19)
/SUBTYPE(compartment_ziplist_number)/ -- (see 5.1.6.27)

5.1.6.13.1 context

AIM element: /SUBTYPE(compartment_nuclear_classification)/ -- (see 5.1.6.15.1)
/SUBTYPE(compartment_safety_class)/ -- (see 5.1.6.18.1)
/SUBTYPE(compartment_security_classification)/ -- (see 5.1.6.19.1)
/SUBTYPE(compartment_ziplist_number)/ -- (see 5.1.6.27.1)

5.1.6.14 COMPARTMENT_NOISE_CATEGORY

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.127, 5.2.4.193
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment noise category’)/}

5.1.6.14.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item

{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.14.2 noise_category

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.236
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'noise category'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'A')
(descriptive_representation_item.description = 'B')
(descriptive_representation_item.description = 'C')
(descriptive_representation_item.description = 'D')
(descriptive_representation_item.description = 'E')
(descriptive_representation_item.description = 'F')
(descriptive_representation_item.description = 'user defined')}

5.1.6.14.3 user_defined_value

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.236
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'user defined value'}
representation_item =>
descriptive_representation_item
descriptive_representation_item.description

5.1.6.15 COMPARTMENT_NUCLEAR_CLASSIFICATION

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.128
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment nuclear classification’)/}

5.1.6.15.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45

Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.15.2 nuclear_classification

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'nuclear classification'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'non nuclear')
(descriptive_representation_item.description = 'nuclear')}

5.1.6.16 COMPARTMENT_OCCUPANCY

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.129
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment occupancy’)/}

5.1.6.16.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.16.2 occupancy

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('occupancy', numeric_measure)/

5.1.6.17 COMPARTMENT_PROPERTY

AIM element: /SUBTYPE(cargo_compartment_property)/ -- (see 5.1.6.2)
/SUBTYPE(compartment_naval_administrative_property)/ -- (see 5.1.6.13)
/SUBTYPE(general_compartment_property)/ -- (see 5.1.6.29)
/SUBTYPE(tank_compartment_property)/ -- (see 5.1.6.31)

5.1.6.17.1 context

AIM element: /SUBTYPE(cargo_compartment_property)/ -- (see 5.1.6.2.1)
/SUBTYPE(compartment_naval_administrative_property)/ -- (see 5.1.6.13.1)
/SUBTYPE(general_compartment_property)/ -- (see 5.1.6.29.1)
/SUBTYPE(tank_compartment_property)/ -- (see 5.1.6.31.1)

5.1.6.18 COMPARTMENT_SAFETY_CLASS

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.130, 5.2.4.194
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment safety class’)/}

5.1.6.18.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.18.2 safety_category

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.237
Reference path: property_definition_representation

property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'safety category'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'A')
(descriptive_representation_item.description = 'B')
(descriptive_representation_item.description = 'C')
(descriptive_representation_item.description = 'user defined')}

5.1.6.18.3 user_defined_value

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.237
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'user defined value'}
representation_item =>
descriptive_representation_item
descriptive_representation_item.description

5.1.6.19 COMPARTMENT_SECURITY_CLASSIFICATION

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.131, 5.2.4.195
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment security classification’)/}

5.1.6.19.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.19.2 security_classification

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.238
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'security classification'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'classified')
(descriptive_representation_item.description = 'secret')
(descriptive_representation_item.description = 'unclassified')
(descriptive_representation_item.description = 'user defined')}

5.1.6.19.3 user_defined_value

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.238
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'user defined value'}
representation_item =>
descriptive_representation_item
descriptive_representation_item.description

5.1.6.20 COMPARTMENT_STIFFENED_SURFACE_AREA_PROPERTY

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.132
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment stiffened surface area property’)/}

5.1.6.20.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item

{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.20.2 stiffened_surface_area

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('stiffened surface area', area_measure)/

5.1.6.21 COMPARTMENT_TIGHTNESS

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.133, 5.2.4.196
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, 'compartment tightness')/}

5.1.6.21.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.21.2 required_bulkhead_tightness

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.239
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'required bulkhead tightness'}
representation_item =>
descriptive_representation_item

{(descriptive_representation_item.description = 'air tight')
(descriptive_representation_item.description = 'non tight')
(descriptive_representation_item.description = 'oil tight')
(descriptive_representation_item.description = 'water tight')
(descriptive_representation_item.description = 'weather tight')
(descriptive_representation_item.description = 'user defined')}

5.1.6.21.3 user_defined_value

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.239
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'user defined value'}
representation_item =>
descriptive_representation_item
descriptive_representation_item.description

5.1.6.22 COMPARTMENT_UNSTIFFENED_SURFACE_AREA_PROPERTY

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.134
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment unstiffened surface area property’)/}

5.1.6.22.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.22.2 unstiffened_surface_area

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->

/REP_TO_VAL_REP_ITEM('unstiffened surface area', area_measure)/

5.1.6.23 COMPARTMENT_VERTICAL_LONGITUDINAL_CROSS_SECTIONAL_-AREA_PROPERTY

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.135
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, 'compartment vertical longitudinal cross sectional area property')/}

5.1.6.23.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.23.2 vertical_longitudinal_cross_sectional_area

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('vertical longitudinal cross sectional area', area_measure)/

5.1.6.24 COMPARTMENT_VERTICAL_TRANSVERSE_CROSS_SECTIONAL_-AREA_PROPERTY

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.136
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, 'compartment vertical transverse cross sectional area property')/}

5.1.6.24.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->

representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.24.2 vertical_transverse_cross_sectional_area

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('vertical transverse cross sectional area',
area_measure)/

5.1.6.25 COMPARTMENT_VOLUME_PERMEABILITY_PROPERTY

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.137
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment volume permeability property’)/}

5.1.6.25.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.25.2 permeability

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('permeability', ratio_measure)/

5.1.6.26 COMPARTMENT_VOLUME_PROPERTY

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.138
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment volume property’)/}

5.1.6.26.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.26.2 volume

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('volume', volume_measure)/

5.1.6.26.3 compartment_volume_property to centre_location (as centre_of_volume)

AIM element: PATH
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items [i] ->
representation_item
{representation_item.name = 'centre of volume'}
representation_item =>
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘centre location’)/}

5.1.6.27 COMPARTMENT_ZIPLIST_NUMBER

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.139
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compartment ziplist number’)/}

5.1.6.27.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.27.2 department_ziplist_number

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'department ziplist number'}
representation_item =>
descriptive_representation_item
descriptive_representation_item.description

5.1.6.27.3 division_ziplist_number

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'division ziplist number'}
representation_item =>
descriptive_representation_item
descriptive_representation_item.description

5.1.6.28 CORROSION_PROTECTION

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.142
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘corrosion protection’)/}

5.1.6.28.1 cathodic_protection

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'cathodic protection'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'true')
(descriptive_representation_item.description = 'false')}

5.1.6.28.2 corrosion_protection to coating (as coating_material)

AIM element: PATH
Reference path: property_definition_representation
property_definition_representation.definition ->
represented_definition = property_definition
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘coating material’}
property_definition_relationship.related_property_definition ->
property_definition
{/CLASS_ID(property_definition, 'coating')/}

5.1.6.28.3 corrosion_protection to coating_level (as coating_height)

AIM element: PATH
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation <-
representation_relationship.rep_1
representation_relationship
{representation_relationship.name = ‘coating height’}
representation_relationship
representation_relationship.rep_2 ->
representation
representation <-
property_definition_representation.used_representation ->
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘coating level’)/}

5.1.6.29 GENERAL_COMPARTMENT_PROPERTY

AIM element: /SUBTYPE(compartment_abbreviated_name)/ -- (see 5.1.6.4)
/SUBTYPE(compartment_acceleration)/ -- (see 5.1.6.5)
/SUBTYPE(compartment_access_authorization)/ -- (see 5.1.6.6)
/SUBTYPE(compartment_insulation)/ -- (see 5.1.6.12)

/SUBTYPE(compartment_noise_category)/ -- (see 5.1.6.14)
/SUBTYPE(compartment_air_circulation_rate) -- (see /5.1.6.7)
/SUBTYPE(compartment_area_property)/ -- (see 5.1.6.8)
/SUBTYPE(compartment_coating)/ -- (see 5.1.6.9)
/SUBTYPE(compartment_illumination)/ -- (see 5.1.6.11)
/SUBTYPE(compartment_occupancy)/ -- (see 5.1.6.16)
/SUBTYPE(compartment_tightness)/ -- (see 5.1.6.21)
/SUBTYPE(compartment_volume_permeability_property)/ -- (see 5.1.6.25)
/SUBTYPE(compartment_volume_property)/ -- (see 5.1.6.26)

5.1.6.29.1 context

AIM element: /SUBTYPE(compartment_abbreviated_name)/ -- (see 5.1.6.4.1)
/SUBTYPE(compartment_acceleration)/ -- (see 5.1.6.5.2)
/SUBTYPE(compartment_access_authorization)/ -- (see 5.1.6.6.2)
/SUBTYPE(compartment_insulation)/ -- (see 5.1.6.12.1)
/SUBTYPE(compartment_noise_category)/ -- (see 5.1.6.14.1)
/SUBTYPE(compartment_air_circulation_rate)/ -- (see 5.1.6.7.2)
/SUBTYPE(compartment_area_property)/ -- (see 5.1.6.8.1)
/SUBTYPE(compartment_coating)/ -- (see 5.1.6.9.1)
/SUBTYPE(compartment_illumination)/ -- (see 5.1.6.11.1)
/SUBTYPE(compartment_occupancy)/ -- (see 5.1.6.16.1)
/SUBTYPE(compartment_tightness)/ -- (see 5.1.6.21.1)
/SUBTYPE(compartment_volume_permeability_property)/ -- (see 5.1.6.25.1)
/SUBTYPE(compartment_volume_property)/ -- (see 5.1.6.26.1)

5.1.6.30 MOMENTS_OF_INERTIA

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.164
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘moments of inertia’)/}

5.1.6.30.1 long_moment_of_inertia

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_SPECIAL_VAL_REP_ITEM('long moment of inertia', 'inertia moment unit')/

5.1.6.30.2 trans_moment_of_inertia

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_SPECIAL_VAL_REP_ITEM('trans moment of inertia', 'inertia moment unit')/

5.1.6.31 TANK_COMPARTMENT_PROPERTY

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.173
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘tank compartment property’)/}

5.1.6.31.1 context

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'context'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'calculated')
(descriptive_representation_item.description = 'estimated')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'measured')
(descriptive_representation_item.description = 'minimum')}

5.1.6.31.2 design_stowage_density

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_SPECIAL_VAL_REP_ITEM('design stowage density', 'density unit')/

5.1.6.31.3 tank_compartment_property to capacity_properties (as liquid_capacity)

AIM element: PATH
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation <-
representation_relationship.rep_1
representation_relationship
{representation_relationship.name = ‘liquid capacity’}
representation_relationship
representation_relationship.rep_2 ->
representation <-
property_definition_representation.used_representation ->
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘capacity properties’)/}

5.1.6.31.4 tank_compartment_property to moments_of_inertia (as moments_of_inertia)

AIM element: PATH
Reference path: property_definition_representation

property_definition_representation.used_representation ->
representation <-
representation_relationship.rep_1
representation_relationship
{representation_relationship.name = 'moments of inertia'}
representation_relationship
representation_relationship.rep_2 ->
representation <-
property_definition_representation.used_representation ->
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, 'moments of inertia')/}

5.1.6.31.5 tank_compartment_property to tank_geometric_parameters (as geometric_parameters)

AIM element: PATH
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation <-
representation_relationship.rep_1
representation_relationship
{representation_relationship.name = 'geometric parameters'}
representation_relationship
representation_relationship.rep_2 ->
representation <-
property_definition_representation.used_representation ->
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, 'tank geometric parameters')/}

5.1.6.31.6 tank_compartment_property to tank_piping_design_properties (as design_properties)

AIM element: PATH
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation <-
representation_relationship.rep_1
representation_relationship
{representation_relationship.name = 'design properties'}
representation_relationship
representation_relationship.rep_2 ->
representation <-
property_definition_representation.used_representation ->
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, 'tank piping design properties')/}

5.1.6.32 TANK_GEOMETRIC_PARAMETERS

AIM element: property_definition_representation
Source: ISO 10303-41

Rules: 5.2.4.183, 5.2.4.209
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘tank geometric parameters’)/}

5.1.6.32.1 breadth_wash

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('breadth wash', length_measure)/

5.1.6.32.2 length_wash

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('length wash', length_measure)/

5.1.6.33 TANK_PIPING_DESIGN_PROPERTIES

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.210
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘tank piping design properties’)/}

5.1.6.33.1 airpipe_height

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('airpipe height', length_measure)/

5.1.6.33.2 filling_height

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('filling height', length_measure)/

5.1.6.33.3 relief_valve_pressure_setting

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_SPECIAL_VAL_REP_ITEM('relief valve pressure setting', 'pressure unit')/

5.1.6.33.4 sounding_pipe_height

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('sounding pipe height', length_measure)/

5.1.7 Compartment_requirements UoF

5.1.7.1 CLASS_BULK_LOAD_REQUIREMENT_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.74, 5.2.4.108
Reference path: {[/CLASS(property_definition, 'class bulk load requirement definition',
'class_compartment_requirement_definition')/]
[/CLASS(property_definition, 'class_compartment_requirement_definition',
'design_requirement')/]
[/CLASS(property_definition, 'design_requirement', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.7.1.1 ambient_temperature

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘class bulk load requirement definition
parameters’, 'ambient temperature')/
{(descriptive_representation_item.description = 'true')
(descriptive_representation_item.description = 'false')}

5.1.7.1.2 angle_of_repose

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class bulk load requirement definition
parameters', 'angle of repose', plane_angle_measure)/

5.1.7.1.3 bulk_cargo_mass

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class bulk load requirement definition
parameters', 'bulk cargo mass', mass_measure)/

5.1.7.1.4 cargo_density

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('class bulk load requirement
definition parameters', 'cargo density', 'density unit')/

5.1.7.1.5 cargo_height

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class bulk load requirement definition parameters', 'cargo height', length_measure)/

5.1.7.1.6 coating

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘class bulk load requirement definition parameters’, 'coating')/
{(descriptive_representation_item.description = 'true')
(descriptive_representation_item.description = 'false')}

5.1.7.1.7 damage_waterline

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class bulk load requirement definition parameters', 'damage waterline', positive_length_measure)/

5.1.7.1.8 max_pressure

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('class bulk load requirement definition parameters', 'max pressure', 'pressure unit')/

5.1.7.1.9 max_temperature

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class bulk load requirement definition parameters', 'max temperature', thermodynamic_temperature_measure)/

5.1.7.1.10 min_pressure

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('class bulk load requirement definition parameters', 'min pressure', 'pressure unit')/

5.1.7.1.11 min_temperature

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class bulk load requirement definition parameters', 'min temperature', thermodynamic_temperature_measure)/

5.1.7.1.12 permeability

AIM element: value_representation_item.value_component
Source: ISO 10303-43

Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class bulk load requirement definition parameters', 'permeability', ratio_measure)/

5.1.7.1.13 top_of_hatch

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class bulk load requirement definition parameters', 'top of hatch', positive_length_measure)/

5.1.7.1.14 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.7.1.15 class_bulk_load_requirement_definition to compartment (as defined_for)

AIM element: PATH
Reference path: property_definition
property_definition.definition ->
characterized_definition =
characterized_product_definition =
product_definition
{/CLASS_ID(product_definition, 'compartment')/}

5.1.7.1.16 class_bulk_load_requirement_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.7.1.17 class_bulk_load_requirement_definition to document_reference_with_address (as specification)

AIM element: PATH
Reference path: property_definition
/DOC_REF(property_definition,'specification')/
document
{/CLASS_ID(document, 'document reference with address')/}

5.1.7.1.18 class_bulk_load_requirement_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.7.1.19 class_bulk_load_requirement_definition to named_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.7.2 CLASS_COMPARTMENT_REQUIREMENT_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.75, 5.2.4.109
Reference path: {[/CLASS(property_definition, 'class compartment requirement definition', 'design_requirement')/]
[/CLASS(property_definition, 'design_requirement', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.7.2.1 ambient_temperature

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘class compartment requirement definition parameters’, 'ambient temperature')/
{(descriptive_representation_item.description = 'true')
(descriptive_representation_item.description = 'false')}

5.1.7.2.2 cargo_density

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('class compartment requirement definition parameters', 'cargo density', 'density unit')/

5.1.7.2.3 cargo_height

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class compartment requirement definition parameters', 'cargo height', length_measure)/

5.1.7.2.4 coating

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘class compartment requirement definition parameters’, 'coating')/
{(descriptive_representation_item.description = 'true')
(descriptive_representation_item.description = 'false')}

5.1.7.2.5 damage_waterline

AIM element: value_representation_item.value_component
Source: ISO 10303-43

Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class compartment requirement definition parameters', 'damage waterline', positive_length_measure)/

5.1.7.2.6 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.7.2.7 max_pressure

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('class compartment requirement definition parameters', 'max pressure', 'pressure unit')/

5.1.7.2.8 max_temperature

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class compartment requirement definition parameters', 'max temperature', thermodynamic_temperature_measure)/

5.1.7.2.9 min_pressure

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('class compartment requirement definition parameters', 'min pressure', 'pressure unit')/

5.1.7.2.10 min_temperature

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class compartment requirement definition parameters', 'min temperature', thermodynamic_temperature_measure)/

5.1.7.2.11 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.7.2.12 class_compartment_requirement_definition to compartment (as defined_for)

AIM element: PATH
Reference path: property_definition
property_definition.definition ->
characterized_definition =
characterized_product_definition =
product_definition
{/CLASS_ID(product_definition, 'compartment')/}

5.1.7.2.13 class_compartment_requirement_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.7.2.14 class_compartment_requirement_definition to document_reference_with_address (as specification)

AIM element: PATH
Reference path: property_definition
/DOC_REF(property_definition,'specification')/
document
{/CLASS_ID(document, 'document reference with address')/}

5.1.7.2.15 class_compartment_requirement_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.7.2.16 class_compartment_requirement_definition to named_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.7.3 CLASS_DECK_LOAD_REQUIREMENT_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.76, 5.2.4.187
Reference path: {[/CLASS(property_definition, 'class deck load requirement definition', 'design_requirement')/]
[/CLASS(property_definition, 'design_requirement', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.7.3.1 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.7.3.2 grab_weight

AIM element: value_representation_item.value_component
Source: ISO 10303-43

Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('class deck load requirement definition parameters', 'grab weight', 'force unit')/

5.1.7.3.3 stowage_height

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class deck load requirement definition parameters', 'stowage height', positive_length_measure)/

5.1.7.3.4 stowage_rate

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('class deck load requirement definition parameters', 'stowage rate', 'density unit')/

5.1.7.3.5 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.7.3.6 class_deck_load_requirement_definition to deck_zone (as defined_for)

AIM element: PATH
Reference path: property_definition
property_definition.definition ->
characterized_definition =
characterized_product_definition =
product_definition
{/CLASS_ID(product_definition, 'deck zone')/}

5.1.7.3.7 class_deck_load_requirement_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.7.3.8 class_deck_load_requirement_definition to document_reference_with_address (as specification)

AIM element: PATH
Reference path: property_definition
/DOC_REF(property_definition,'specification')/
document
{/CLASS_ID(document, 'document reference with address')/}

5.1.7.3.9 class_deck_load_requirement_definition to global_id (as id)

AIM element: PATH

Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.7.3.10 class_deck_load_requirement_definition to named_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.7.3.11 class_deck_load_requirement_definition to vehicle_load_description (as vehicle_load)

AIM element: PATH
Reference path: property_definition
{/CLASS_ID(property_definition, ‘class deck load requirement definition’)/})
property_definition = represented_definition
represented_definition <-
/PDR_NAME('class deck load requirement definition parameters')/
property_definition_representation.used_representation ->
representation <-
representation_relationship.rep_1
representation_relationship
{representation_relationship.name = ‘vehicle load’}
representation_relationship
representation_relationship.rep_2 ->
representation <-
property_definition_representation.used_representation ->
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘vehicle load description’)/}

5.1.7.4 CLASS_TANK_REQUIREMENT_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.79, 5.2.4.112
Reference path: {[/CLASS(property_definition, 'class tank requirement definition',
'class_compartment_requirement_definition')/]
[/CLASS(property_definition, 'class_compartment_requirement_definition',
'design_requirement')/]
[/CLASS(property_definition, 'design_requirement', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.7.4.1 ambient_temperature

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘class tank requirement definition parameters’, 'ambient temperature')/

{(descriptive_representation_item.description = 'true')
(descriptive_representation_item.description = 'false')}

5.1.7.4.2 cargo_density

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('class tank requirement definition parameters', 'cargo density', 'density unit')/

5.1.7.4.3 cargo_height

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class tank requirement definition parameters', 'cargo height', length_measure)/

5.1.7.4.4 coating

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘class tank requirement definition parameters’, 'coating')/
{(descriptive_representation_item.description = 'true')
(descriptive_representation_item.description = 'false')}

5.1.7.4.5 damage_waterline

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class tank requirement definition parameters', 'damage waterline', positive_length_measure)/

5.1.7.4.6 free_surface_parameters

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class tank requirement definition parameters', 'free surface parameters', positive_length_measure)/

5.1.7.4.7 max_pressure

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('class tank requirement definition parameters', 'max pressure', 'pressure unit')/

5.1.7.4.8 max_temperature

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class tank requirement definition parameters', 'max temperature', thermodynamic_temperature_measure)/

5.1.7.4.9 min_pressure

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('class tank requirement definition parameters', 'min pressure', 'pressure unit')/

5.1.7.4.10 min_temperature

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class tank requirement definition parameters', 'min temperature', thermodynamic_temperature_measure)/

5.1.7.4.11 overflow_height

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class tank requirement definition parameters', 'overflow height', positive_length_measure)/

5.1.7.4.12 partial_filling

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘class tank requirement definition parameters’, 'partial filling')/
{(descriptive_representation_item.description = 'true')
(descriptive_representation_item.description = 'false')}

5.1.7.4.13 pressure_relief_setting

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_TO_SPECIAL_VAL_REP_ITEM('class tank requirement definition parameters', 'pressure relief setting', 'pressure unit')/

5.1.7.4.14 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.7.4.15 class_tank_requirement_definition to compartment (as defined_for)

AIM element: PATH
Reference path: property_definition
property_definition.definition ->
characterized_definition =
characterized_product_definition =
product_definition
{/CLASS_ID(product_definition, 'compartment')/}

5.1.7.4.16 class_tank_requirement_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.7.4.17 class_tank_requirement_definition to document_reference_with_address (as specification)

AIM element: PATH
Reference path: property_definition
/DOC_REF(property_definition,'specification')/
document
{/CLASS_ID(document, 'document reference with address')/}

5.1.7.4.18 class_tank_requirement_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.7.4.19 class_tank_requirement_definition to named_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.7.5 COMPARTMENT_DESIGN_REQUIREMENT

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.80, 5.2.4.121, 5.2.4.190
Reference path: {[/CLASS(property_definition, 'compartment design requirement', 'design requirement')/]
[/CLASS(property_definition, 'design requirement', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.7.5.1 description

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('compartment design requirement parameters', 'description')/

5.1.7.5.2 requirement_type

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.240
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘compartment design requirement parameters’, 'requirement type')/
{(descriptive_representation_item.description = 'combat system')
(descriptive_representation_item.description = 'electrical')
(descriptive_representation_item.description = 'electronic')
(descriptive_representation_item.description = 'hvac')
(descriptive_representation_item.description = 'naval architecture')
(descriptive_representation_item.description = 'outfit furnishing')
(descriptive_representation_item.description = 'painting coating')
(descriptive_representation_item.description = 'piping')
(descriptive_representation_item.description = 'structural')
(descriptive_representation_item.description = 'user defined')}

5.1.7.5.3 user_defined_value

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.240
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘compartment design requirement parameters’, 'user defined value')/

5.1.7.5.4 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.7.5.5 compartment_design_requirement to document_reference_with_address (as specification)

AIM element: PATH
Reference path: property_definition
/DOC_REF(property_definition,’specification’)/
document
{/CLASS_ID(document, ‘document reference with address’)/}

5.1.7.5.6 compartment_design_requirement to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.7.5.7 compartment_design_requirement to space (as defined_for)

AIM element: PATH
Reference path: property_definition

property_definition.definition ->
characterized_definition =
characterized_product_definition =
product_definition
{(/CLASS_ID(product_definition, 'compartment')/)
(/CLASS_ID(product_definition, 'deck zone')/)
(/CLASS_ID(product_definition, 'zone')/)}

5.1.7.6 VEHICLE_LOAD_DESCRIPTION

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.180, 5.2.4.212
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘vehicle load description’)/}

5.1.7.6.1 load_handling

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'load handling'}
representation_item =>
descriptive_representation_item
{(descriptive_representation_item.description = 'true')
(descriptive_representation_item.description = 'false')}

5.1.7.6.2 load_per_wheel

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_SPECIAL_VAL_REP_ITEM('load per wheel', 'force unit')/

5.1.7.6.3 max_tyre_pressure

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_SPECIAL_VAL_REP_ITEM('max tyre pressure', 'pressure unit')/

5.1.7.6.4 number_of_wheels

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('number of wheels', numeric_measure)/

5.1.7.6.5 print_area

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('print area', area_measure)/

5.1.7.6.6 type_of_vehicle

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'type of vehicle'}
representation_item =>
descriptive_representation_item
descriptive_representation_item.description

5.1.8 Configuration_management UoF

5.1.8.1 ALTERNATIVE_VERSION_RELATIONSHIP

AIM element: identification_assignment_relationship
Source: ISO 10303-41
Rules: 5.2.4.6, 5.2.4.5
Reference path: /ROOT_CLASS(identification_assignment_relationship, 'alternative version relationship')/

5.1.8.1.1 reason

AIM element: identification_assignment_relationship.description
Source: ISO 10303-41
Rules: 5.2.4.4

5.1.8.1.2 alternative_version_relationship to versionable_object (as alternative 1)

AIM element: PATH
Reference path: identification_assignment_relationship
identification_assignment_relationship.related_assignment ->
(/RELATE_ID_2_VO(document)/)
(/RELATE_ID_2_VO(group)/)
(/RELATE_ID_2_VO(product_definition)/)
(/RELATE_ID_2_VO(product_definition_relationship)/)
(/RELATE_ID_2_VO(product_definition_shape)/)
(/RELATE_ID_2_VO(product_related_product_category)/)
(/RELATE_ID_2_VO(property_definition)/)

5.1.8.1.3 alternative_version_relationship to versionable_object (as alternative 2)

AIM element: PATH
Reference path: identification_assignment_relationship

identification_assignment_relationship.relating_assignment ->
(/RELATE_ID_2_VO(document)/)
(/RELATE_ID_2_VO(group)/)
(/RELATE_ID_2_VO(product_definition)/)
(/RELATE_ID_2_VO(product_definition_relationship)/)
(/RELATE_ID_2_VO(product_definition_shape)/)
(/RELATE_ID_2_VO(product_related_product_category)/)
(/RELATE_ID_2_VO(property_definition)/)

5.1.8.2 APPROVAL_EVENT

AIM element: approval
Source: ISO 10303-41
Reference path: {[/CLASS(approval, 'approval event', 'event')/]
[/ROOT_CLASS(approval, 'event')/]}

5.1.8.2.1 caused_by

AIM element: person_and_organization
Source: ISO 10303-41
Rules: 5.2.4.10
Reference path: approval <-
approval_person_organization.authorized_approval
approval_person_organization
approval_person_organization.person_organization ->
person_organization_select = person_and_organization
person_and_organization

5.1.8.2.2 caused_when

AIM element: approval_date_time.date_time
Source: ISO 10303-41
Rules: 5.2.4.9
Reference path: approval <-
approval_date_time.dated_approval
approval_date_time
approval_date_time.date_time ->
{(date_time_select = date_and_time
date_and_time)
(date_time_select = calendar_date
calendar_date)}

5.1.8.2.3 description

AIM element: approval.level
Source: ISO 10303-41

5.1.8.2.4 result

AIM element: approval_status.name
Source: ISO 10303-41
Reference path: approval
approval.status ->
approval_status

approval_status.name
{(approval_status.name = 'unapproved')
(approval_status.name = 'approved')
(approval_status.name = 'rejected')}

5.1.8.2.5 user_defined_result

AIM element: approval_status.name
Source: ISO 10303-41
Reference path: approval
approval.status ->
approval_status
approval_status.name

5.1.8.2.6 approval_event to approval_history (as approval_reference)

AIM element: PATH
Rules: 5.2.4.51
Reference path: approval
group_item = approval
group_item <-
applied_group_assignment.items[i]
applied_group_assignment <=
group_assignment
/ROLE_ASSGN(group_assignment)/
{object_role.name = 'approvals'}
group_assignment.assigned_group ->
group
{/CLASS_ID(group, 'approval history')/}

5.1.8.3 APPROVAL_HISTORY

AIM element: group
Source: ISO 10303-41
Reference path: /ROOT_CLASS(group, 'approval history')/

5.1.8.4 status

AIM element: PATH
Reference path: group <-
/GROUPS(approval, 'approvals')/
approval
approval.status ->
approval_status
approval_status.name
{(approval_status.name = 'unapproved')
(approval_status.name = 'approved')
(approval_status.name = 'rejected')}

5.1.8.4.1 approval_history to approval_event (as approvals)

AIM element: PATH
Rules: 5.2.4.12, 5.2.4.227, 5.2.4.13
Reference path: group <-
/GROUPS(approval, 'approvals')/

approval

5.1.8.4.2 approval_history to definition (as subject)

AIM element: PATH
Rules: 5.2.4.11
Reference path: group <-
/GROUPS(approval, 'approvals')/
approvals <-
(/APPROVES(product_defnition, 'subject')/
product_defnition
{/CLASS_ID(product_definition, ‘definition’)/})
(/APPROVES(property_definition, 'subject')/
property_definition
{/CLASS_ID(property_definition, ‘definition’)/})
(/APPROVES(product_definition_shape, 'subject')/
product_definition_shape
{/CLASS_ID(product_definition_shape, ‘definition’)/})
(/APPROVES(product_related_product_category, 'subject')/
product_related_product_category
{/CLASS_ID(product_related_product_category, ‘definition’)/})

5.1.8.5 CHANGE

AIM element: action
Source: ISO 10303-41
Reference path: {[/CLASS(action, ’change’, ’item’)/]
[/CLASS(action, ’item’, ’definable object’)/]
[/ROOT_CLASS(action, ’definable object’)/]}

5.1.8.5.1 description

AIM element: action.description
Source: ISO 10303-41

5.1.8.5.2 name

AIM element: action.name
Source: ISO 10303-41

5.1.8.5.3 the_class

AIM element: group.name
Source: ISO 10303-41
Reference path: action
classification_item = action
classification_item <-
applied_classification_assignment.items[i]
applied_classification_assignment
applied_classification_assignment <=
classification_assignment
{classification_assignment.role ->
classification_role
classification_role.name = ‘change class‘}

classification_assignment.assigned_class ->
group
group.name
{group.name = ‘the_class’}

5.1.8.5.4 change to external_reference (as documentation)

#1: If the as documentation refers to an External_reference

AIM element: PATH
Reference path: action
action = external_identification_item
external_identification_item <-
applied_external_identification_assignment.items[i]
applied_external_identification_assignment

#2: If the as documentation refers to a Document_reference_with_address

AIM element: PATH
Reference path: action
/DOC_REF(action,’documentation’)/
document
{/CLASS_ID(document, ‘document reference with address’)/}

5.1.8.5.5 change to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.3
Reference path: action
identification_item = action <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.8.5.6 change to ship (as ship_context)

AIM element: PATH
Reference path: action <-
applied_action_assignment.assigned_action
applied_action_assignment
applied_action_assignment.items[i] ->
action_item
action_item = product
product
{/CLASS_ID(product, ‘ship’)/}

5.1.8.6 CHANGE_DEFINITION

#1: If the Change_definition is a Change_request
#2: If the Change_definition is an Change_plan
#3: If the Change_definition is a Change_realization

AIM element: #1: (versioned_action_request)
#2: (action_request_solution)
#3: (executed_action)

Source: ISO 10303-41
ISO 10303-41
ISO 10303-41

5.1.8.6.1 author

AIM element: #1: /SUBTYPE(Change_request)/) -- (see 5.1.8.10.2)
#2: /SUBTYPE(Change_plan)/ -- (see 5.1.8.8.1)
#3: /SUBTYPE(Change_realization)/ -- (see 5.1.8.9.1)

5.1.8.6.2 date_time

AIM element: #1: /SUBTYPE(Change_request)/ -- (see 5.1.8.10.3)
#2: /SUBTYPE(Change_plan)/ -- (see 5.1.8.8.2)
#3: /SUBTYPE(Change_realization)/ -- (see 5.1.8.9.2)

5.1.8.6.3 description

AIM element: #1: /SUBTYPE(Change_request)/ -- (see 5.1.8.10.4)
#2: /SUBTYPE(Change_plan)/ -- (see 5.1.8.8.3)
#3: /SUBTYPE(Change_realization)/ -- (see 5.1.8.9.3)

5.1.8.6.4 version_id

AIM element: #1: /SUBTYPE(Change_request)/ -- (see 5.1.8.10.8)
#2: /SUBTYPE(Change_plan)/ -- (see 5.1.8.8.4)
#3: /SUBTYPE(Change_realization)/ -- (see 5.1.8.9.4)

5.1.8.6.5 change_definition to change (as defined_for)

AIM element: #1: /SUBTYPE(Change_request)/ -- (see 5.1.8.10.9)
#2: /SUBTYPE(Change_plan)/ -- (see 5.1.8.8.5)
#3: /SUBTYPE(Change_realization)/ -- (see 5.1.8.9.5)

5.1.8.6.6 change_definition to global_id (as id)

AIM element: #1: /SUBTYPE(Change_request)/ -- (see 5.1.8.10.11)
#2: /SUBTYPE(Change_plan)/ -- (see 5.1.8.8.9)
#3: /SUBTYPE(Change_realization)/ -- (see 5.1.8.9.9)

5.1.8.7 CHANGE_IMPACT

AIM element: applied_action_request_assignment
Source: ISO 10303-215
Reference path: applied_action_request_assignment <=
action_request_assignment
{/ROOT_CLASS(applied_action_request_assignment, ‘change impact’)/}

5.1.8.7.1 change_impact to versionable_object_change_event (as impact)

AIM element: PATH
Rules: 5.2.4.28
Reference path: applied_action_request_assignment
applied_action_request_assignment.items [i] ->
action_request_item

action_request_item = action
{/CLASS_ID(action, ‘versionable object change event’)/}

5.1.8.8 CHANGE_PLAN

AIM element: action_request_solution
Source: ISO 10303-41
Reference path: {[/CLASS(action_request_solution, ‘change plan’, ‘change definition’)/]
[/CLASS(action_request_solution, ‘change definition’, ‘definition’)/]
[/CLASS(action_request_solution, 'definition', 'versionable object')/]
[/ROOT_CLASS(action_request_solution, 'versionable object')/]}

5.1.8.8.1 author

AIM element: applied_person_and_organization_assignment.assigned_person_and_organization
Source: ISO 10303-215
Rules: 5.2.4.14
Reference path: /PERS_ORG_ASSGN(action_request_solution, 'author')/

5.1.8.8.2 date_time

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-215
Rules: 5.2.4.33
Reference path: /DAT_TIME_ASSGN(action_request_solution, 'date time')/

5.1.8.8.3 description

AIM element: action_request_solution.description
Source: ISO 10303-41
Reference path: /DESCRIPTION_ASSGN(action_request_solution)/

5.1.8.8.4 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(action_request_solution)/

5.1.8.8.5 change_plan to change (as defined_for)

AIM element: PATH
Rules: 5.2.4.1
Reference path: action_request_solution
action_request_solution.method ->
action_method <-
action.chosen_method
action
{/CLASS_ID(action, ‘change‘)/}

5.1.8.8.6 change_plan to change_impact (as planned_impact)

AIM element: PATH
Rules: 5.2.4.29
Reference path: action_request_solution

action_request_solution.request ->
versioned_action_request <-
action_request_assignment.assigned_action_request
action_request_assignment =>
applied_action_request_assignment
{/CLASS_ID(applied_action_request_assignment, ‘change impact’)/}

5.1.8.8.7 change_plan to change_request (as chosen_solution_for)

AIM element: PATH
Reference path: action_request_solution
action_request_solution.request ->
versioned_action_request
{/CLASS_ID(versioned_action_request, ‘change request’)/}

5.1.8.8.8 change_plan to check (as checks)

AIM element: PATH
Reference path: action_request_solution
action_request_solution = action_item
action_item <-
applied_action_assignment.items[i]
applied_action_assignment <=
action_assignment
action_assignment.assigned_action ->
action
{/CLASS_ID(action, ‘check’)/}

5.1.8.8.9 change_plan to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.2
Reference path: action_request_solution
identification_item = action_request_solution <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.8.9 CHANGE_REALIZATION

AIM element: executed_action
Source: ISO 10303-41
Reference path: {[/CLASS(executed_action, ‘change realization’, ‘change definition’)/]
[/CLASS(executed_action, ‘change definition’, ‘definition’)/]
[/CLASS(executed_action, 'definition', 'versionable object')/]
[/ROOT_CLASS(executed_action, 'versionable object')/]}

5.1.8.9.1 author

AIM element: applied_person_and_organization_assignment.assigned_person_and_organization
Source: ISO 10303-215
Rules: 5.2.4.15
Reference path: /PERS_ORG_ASSGN(executed_action, 'author')/

5.1.8.9.2 date_time

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-215
Rules: 5.2.4.34
Reference path: /DAT_TIME_ASSGN(executed_action, 'date time')/

5.1.8.9.3 description

AIM element: executed_action.description
Source: ISO 10303-41

5.1.8.9.4 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(executed_action)/

5.1.8.9.5 change_realization to change (as defined_for)

AIM element: PATH
Reference path: executed_action <=
action <-
action_relationship.related_action
action_relationship
action_relationship.relating_action
action
{/CLASS_ID(action, ‘change‘)/}

5.1.8.9.6 change_realization to change_impact (as impact)

AIM element: PATH
Reference path: executed_action
action_request_item = executed_action
action_request_item <-
applied_action_request_assignment.items[i]
applied_action_request_assignment
{/CLASS_ID(applied_action_request_assignment, ‘change impact’)/}

5.1.8.9.7 change_realization to change_plan (as realization_of)

AIM element: PATH
Reference path: executed_action <=
action
action.chosen_method ->
action_method <-
action_request_solution.method
action_request_solution
{/CLASS_ID(action_request_solution, ‘change plan’)/}

5.1.8.9.8 change_realization to check (as checks)

AIM element: PATH
Reference path: executed_action

action_item = executed_action
action_item <-
applied_action_assignment.items[i]
applied_action_assignment <=
action_assignment
action_assignment.assigned_action ->
action
{/CLASS_ID(action, ‘check’)/}

5.1.8.9.9 change_realization to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.40
Reference path: executed_action
identification_item = executed_action
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.8.10 CHANGE_REQUEST

AIM element: versioned_action_request
Source: ISO 10303-41
Reference path: {[/CLASS(versioned_action_request, ‘change request’, ‘change definition’)/]
[/CLASS(versioned_action_request, ‘change definition’, ‘definition’)/]
[/CLASS(versioned_action_request, 'definition', 'versionable object')/]
[/ROOT_CLASS(versioned_action_request, 'versionable object')/]}

5.1.8.10.1 addressee

AIM element: applied_person_and_organization_assignment.assigned_person_and_organization
Source: ISO 10303-215
Reference path: /PERS_ORG_ASSGN(versioned_action_request, 'addressee')/

5.1.8.10.2 author

AIM element: applied_person_and_organization_assignment.assigned_person_and_organization
Source: ISO 10303-215
Rules: 5.2.4.16
Reference path: /PERS_ORG_ASSGN(versioned_action_request, 'author')/

5.1.8.10.3 date_time

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-215
Rules: 5.2.4.35
Reference path: /DAT_TIME_ASSGN(versioned_action_request, 'date time')/

5.1.8.10.4 description

AIM element: versioned_action_request.description
Source: ISO 10303-41

5.1.8.10.5 initiator

AIM element: applied_person_and_organization_assignment.assigned_person_and_organization
Source: ISO 10303-215
Rules: 5.2.4.47
Reference path: /PERS_ORG_ASSGN(versioned_action_request, 'initiator')/

5.1.8.10.6 problem

AIM element: versioned_action_request.purpose
Source: ISO 10303-41

5.1.8.10.7 solution_description

AIM element: action_request_solution.description
Source: ISO 10303-41
Reference path: versioned_action_request <-
action_request_solution.request
action_request_solution
/DESCRIPTION_ASSGN(action_request_solution)/

5.1.8.10.8 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(versioned_action_request)/

5.1.8.10.9 change_request to change (as defined_for)

AIM element: PATH
Reference path: versioned_action_request <-
action_request_solution.request
action_request_solution
action_request_solution.method ->
action_method <-
action.chosen_method
action
{/CLASS_ID(action, ‘change‘)/}

5.1.8.10.10 change_request to change_impact (as solution_alternatives)

AIM element: PATH
Reference path: versioned_action_request <-
action_request_assignment.assigned_action_request
action_request_assignment =>
applied_action_request_assignment
{/CLASS_ID(applied_action_request_assignment, ‘change impact’)/}

5.1.8.10.11 change_request to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.252
Reference path: versioned_action_request
identification_item = versioned_action_request

identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.8.11 CHECK

AIM element: action
Source: ISO 10303-41
Reference path: {[/CLASS(action, ‘check’, ‘event’)/]
[/ROOT_CLASS(action, ‘event’)/]}

5.1.8.11.1 caused_by

AIM element: applied_person_and_organization_assignment.assigned_person_and_organization
Source: ISO 10303-215
Rules: 5.2.4.17
Reference path: /PERS_ORG_ASSGN(action, 'caused by')/

5.1.8.11.2 caused_when

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-215
Rules: 5.2.4.22
Reference path: /DAT_TIME_ASSGN(action, 'caused when')/

5.1.8.11.3 description

AIM element: action.description
Source: ISO 10303-41

5.1.8.12 ENVISAGED_VERSION_CREATION

AIM element: action
Source: ISO 10303-41
Reference path: {[/CLASS(action, ‘envisaged version creation’,
‘versionable object change event’)/]
[/CLASS(action, ‘versionable_object change event’, ‘event’)/]
[/ROOT_CLASS(action, ‘event’)/]}

5.1.8.12.1 category

AIM element: action.name
Source: ISO 10303-41

5.1.8.12.2 caused_by

AIM element: applied_person_and_organization_assignment.assigned_person_and_organization
Source: ISO 10303-215
Rules: 5.2.4.18
Reference path: /PERS_ORG_ASSGN(action, 'caused by')/

5.1.8.12.3 caused_when

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-215

Rules: 5.2.4.23
Reference path: /DAT_TIME_ASSGN(action, 'caused when')/

5.1.8.12.4 description

AIM element: action.description
Source: ISO 10303-41
Rules: 5.2.4.39

5.1.8.12.5 envisaged_version_creation to versionable_object (as base)

AIM element: PATH
Reference path: action <-
action_assignment.assigned_action
action_assignment
/ROLE_ASSGN(action_assignment)/
{object_role.name = 'base'}
action_assignment =>
(/RELATE_ACT_2_VO(document)/)
(/RELATE_ACT_2_VO(group)/)
(/RELATE_ACT_2_VO(product_definition)/)
(/RELATE_ACT_2_VO(product_definition_relationship)/)
(/RELATE_ACT_2_VO(product_definition_shape)/)
(/RELATE_ACT_2_VO(product_related_product_category)/)
(/RELATE_ACT_2_VO(property_definition)/)

5.1.8.13 EVENT

#1: If the Event is a Versionable_object_change_event
#2: If the Event is an Approval_event
#3: If the Event is a Check

AIM element: #1: (action)
#2: (approval)
#3: (action)
Source: ISO 10303-41
ISO 10303-41
ISO 10303-41

5.1.8.13.1 caused_by

AIM element: #1: /SUBTYPE(Versionable_object_change_event)/ -- (see 5.1.8.21.1)
#2: /SUBTYPE(Approval_event)/ -- (see 5.1.8.2.1)
#3: /SUBTYPE(Check)/ -- (see 5.1.8.11.1)

5.1.8.13.2 caused_when

AIM element: #1: /SUBTYPE(Versionable_object_change_event)/ -- (see 5.1.8.21.2)
#2: /SUBTYPE(Approval_event)/ -- (see 5.1.8.2.2)
#3: /SUBTYPE(Check)/ -- (see 5.1.8.11.2)

5.1.8.13.3 description

AIM element: #1: /SUBTYPE(Versionable_object_change_event)/ -- (see 5.1.8.21.3)
#2: /SUBTYPE(Approval_event)/ -- (see 5.1.8.2.3)

#3: /SUBTYPE(Check)/ -- (see 5.1.8.11.3)

5.1.8.14 REVISION

AIM element: group
Source: ISO 10303-41
Rules: 5.2.4.50
Reference path: {[/CLASS(group, ‘revision’, ‘versionable object’)/]
[/ROOT_CLASS(group, ‘versionable object’)/]}

5.1.8.14.1 description

AIM element: group.description
Source: ISO 10303-41
Rules: 5.2.4.216

5.1.8.14.2 name

AIM element: group.name
Source: ISO 10303-41

5.1.8.14.3 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(group)/

5.1.8.14.4 revision to versionable_object (as members)

AIM element: PATH
Rules: 5.2.4.50
Reference path: group <-
(/RELATE_GROUP_2_VO(document, 'members')/)
(/RELATE_GROUP_2_VO(group, 'members')/)
(/RELATE_GROUP_2_VO(product_definition, 'members')/)
(/RELATE_GROUP_2_VO(product_definition_relationship, 'members')/)
(/RELATE_GROUP_2_VO(product_definition_shape, 'members')/)
(/RELATE_GROUP_2_VO(product_related_product_category, 'members')/)
(/RELATE_GROUP_2_VO(property_definition, 'members’)/)

5.1.8.15 REVISION_WITH_CONTEXT

AIM element: group
Source: ISO 10303-41
Reference path: {[/CLASS(group, 'revision with context', 'revision')/]
[/CLASS(group, ‘revision’, ‘versionable object’)/]
[/ROOT_CLASS(group, ‘versionable object’)/]}

5.1.8.15.1 description

AIM element: group.description
Source: ISO 10303-41
Rules: 5.2.4.216

5.1.8.15.2 name

AIM element: group.name
Source: ISO 10303-41

5.1.8.15.3 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(group)/

5.1.8.15.4 revision_with_context to definable_object (as context_of_revision)

AIM element: PATH
Rules: 5.2.4.217
Reference path: group <-
(/RELATE_GROUP_2_DO(product, 'context of revision')/)
(/RELATE_GROUP_2_DO(product_definition, 'context of revision')/)
(/RELATE_GROUP_2_DO(product_definition_relationship,
'context of revision')/)

5.1.8.15.5 revision_with_context to versionable_object (as members)

AIM element: /SUPERTYPE(revision)/ -- (see 5.1.8.14.4)

5.1.8.16 VERSION_CREATION

AIM element: action
Source: ISO 10303-41
Reference path: {[/CLASS(action, 'version creation', 'versionable object change event')/]
[/CLASS(action, 'versionable object change event', 'event')/]
[/ROOT_CLASS(action, 'event')/]}

5.1.8.16.1 caused_by

AIM element: applied_person_and_organization_assignment.assigned_person_and_organization
Source: ISO 10303-215
Rules: 5.2.4.19
Reference path: /PERS_ORG_ASSGN(action, 'caused by')/

5.1.8.16.2 caused_when

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-215
Rules: 5.2.4.24
Reference path: /DAT_TIME_ASSGN(action, 'caused when')/

5.1.8.16.3 description

AIM element: action.description
Source: ISO 10303-41
Rules: 5.2.4.241

5.1.8.16.4 version_creation to versionable_object (as base)

AIM element: PATH
Reference path: action <-
action_assignment.assigned_action
action_assignment
/ROLE_ASSGN(action_assignment)/
{object_role.name = 'base'}
action_assignment =>
(/RELATE_ACT_2_VO(document)/)
(/RELATE_ACT_2_VO(group)/)
(/RELATE_ACT_2_VO(product_definition)/)
(/RELATE_ACT_2_VO(product_definition_relationship)/)
(/RELATE_ACT_2_VO(product_definition_shape)/)
(/RELATE_ACT_2_VO(product_related_product_category)/)
(/RELATE_ACT_2_VO(property_definition)/)

5.1.8.16.5 version_creation to versionable_object (as subject)

AIM element: PATH
Reference path: action <-
action_assignment.assigned_action
action_assignment
/ROLE_ASSGN(action_assignment)/
{object_role.name = 'subject'}
action_assignment =>
(/RELATE_ACT_2_VO(document)/)
(/RELATE_ACT_2_VO(group)/)
(/RELATE_ACT_2_VO(product_definition)/)
(/RELATE_ACT_2_VO(product_definition_relationship)/)
(/RELATE_ACT_2_VO(product_definition_shape)/)
(/RELATE_ACT_2_VO(product_related_product_category)/)
(/RELATE_ACT_2_VO(property_definition)/)

5.1.8.17 VERSION_DELETION

AIM element: action
Source: ISO 10303-41
Reference path: {[/CLASS(action, 'version deletion', 'versionable_object change event')/]
[/CLASS(action, 'versionable_object change event', 'event')/]
[/ROOT_CLASS(action, 'event')/]}

5.1.8.17.1 caused_by

AIM element: applied_person_and_organization_assignment.assigned_person_and_organization
Source: ISO 10303-215
Rules: 5.2.4.20
Reference path: /PERS_ORG_ASSGN(action, 'caused by')/

5.1.8.17.2 caused_when

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-215
Rules: 5.2.4.25

Reference path: /DAT_TIME_ASSGN(action, 'caused when')/

5.1.8.17.3 description

AIM element: action.description
Source: ISO 10303-41
Rules: 5.2.4.242

5.1.8.17.4 version_deletion to versionable_object (as subject)

AIM element: PATH
Reference path: action <-
action_assignment.assigned_action
action_assignment
/ROLE_ASSGN(action_assignment)/
{object_role.name = 'subject'}
action_assignment =>
(/RELATE_ACT_2_VO(document)/)
(/RELATE_ACT_2_VO(group)/)
(/RELATE_ACT_2_VO(product_definition)/)
(/RELATE_ACT_2_VO(product_definition_relationship)/)
(/RELATE_ACT_2_VO(product_definition_shape)/)
(/RELATE_ACT_2_VO(product_related_product_category)/)
(/RELATE_ACT_2_VO(property_definition)/)

5.1.8.18 VERSION_HISTORY

AIM element: group
Source: ISO 10303-41
Rules: 5.2.4.246
Reference path: /ROOT_CLASS(group, 'version history')/

5.1.8.18.1 version_history to version_relationship (as relationships)

AIM element: PATH
Reference path: group <-
/GROUPS(identification_assignment_relationship, 'relationships')/
identification_assignment_relationship
{/CLASS_ID(identification_assignment_relationship, 'version relationship')/}

5.1.8.18.2 version_history to versionable_object (as current_version)

AIM element: PATH
Rules: 5.2.4.245, 5.2.4.243
Reference path: group <-
(/RELATE_GROUP_2_VO(document, 'current version')/)
(/RELATE_GROUP_2_VO(group, 'current version')/)
(/RELATE_GROUP_2_VO(product_definition, 'current version')/)
(/RELATE_GROUP_2_VO(product_definition_relationship, 'current version')/)
(/RELATE_GROUP_2_VO(product_definition_shape, 'current version')/)
(/RELATE_GROUP_2_VO(product_related_product_category, 'current version')/)
(/RELATE_GROUP_2_VO(property_definition, 'current version')/)

5.1.8.18.3 version_history to versionable_object (as versions)

AIM element: PATH
Rules: 5.2.4.253
Reference path: group <-
(/RELATE_GROUP_2_VO(document, 'versions')/)
(/RELATE_GROUP_2_VO(group, 'versions')/)
(/RELATE_GROUP_2_VO(product_definition, 'versions')/)
(/RELATE_GROUP_2_VO(product_definition_relationship, 'versions')/)
(/RELATE_GROUP_2_VO(product_definition_shape, 'versions')/)
(/RELATE_GROUP_2_VO(product_related_product_category, 'versions')/)
(/RELATE_GROUP_2_VO(property_definition, 'versions')/)

5.1.8.19 VERSION_MODIFICATION

AIM element: action
Source: ISO 10303-41
Reference path: {[/CLASS(action, 'version modification', 'versionable object change event')/]
[/CLASS(action, 'versionable object change event', 'event')/]
[/ROOT_CLASS(action, 'event')/]}

5.1.8.19.1 caused_by

AIM element: applied_person_and_organization_assignment.assigned_person_and_organization
Source: ISO 10303-215
Rules: 5.2.4.21
Reference path: /PERS_ORG_ASSGN(action, 'caused by')/

5.1.8.19.2 caused_when

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-215
Rules: 5.2.4.26
Reference path: /DAT_TIME_ASSGN(action, 'caused when')/

5.1.8.19.3 description

AIM element: action.description
Source: ISO 10303-41
Rules: 5.2.4.247

5.1.8.19.4 version_modification to versionable_object (as base)

AIM element: PATH
Reference path: action <-
action_assignment.assigned_action
action_assignment
/ROLE_ASSGN(action_assignment)/
{object_role.name = 'base'}
action_assignment =>
(/RELATE_ACT_2_VO(document)/)
(/RELATE_ACT_2_VO(group)/)
(/RELATE_ACT_2_VO(product_definition)/)

(/RELATE_ACT_2_VO(product_definition_relationship)/)
(/RELATE_ACT_2_VO(product_definition_shape)/)
(/RELATE_ACT_2_VO(product_related_product_category)/)
(/RELATE_ACT_2_VO(property_definition)/)

5.1.8.19.5 version_modification to versionable_object (as subject)

AIM element: PATH
Reference path: action <-
action_assignment.assigned_action
action_assignment
/ROLE_ASSGN(action_assignment)/
{object_role.name = 'subject'}
action_assignment =>
(/RELATE_ACT_2_VO(document)/)
(/RELATE_ACT_2_VO(group)/)
(/RELATE_ACT_2_VO(product_definition)/)
(/RELATE_ACT_2_VO(product_definition_relationship)/)
(/RELATE_ACT_2_VO(product_definition_shape)/)
(/RELATE_ACT_2_VO(product_related_product_category)/)
(/RELATE_ACT_2_VO(property_definition)/)

5.1.8.20 VERSION_RELATIONSHIP

AIM element: identification_assignment_relationship
Source: ISO 10303-41
Rules: 5.2.4.248, 5.2.4.250
Reference path: /ROOT_CLASS(identification_assignment_relationship, 'version relationship')/

5.1.8.20.1 reason

AIM element: identification_assignment_relationship.description
Source: ISO 10303-41
Rules: 5.2.4.249

5.1.8.20.2 version_relationship to versionable_object (as predecessor)

AIM element: PATH
Reference path: identification_assignment_relationship
identification_assignment_relationship.related_assignment ->
(/RELATE_ID_2_VO(document)/)
(/RELATE_ID_2_VO(group)/)
(/RELATE_ID_2_VO(product_definition)/)
(/RELATE_ID_2_VO(product_definition_relationship)/)
(/RELATE_ID_2_VO(product_definition_shape)/)
(/RELATE_ID_2_VO(product_related_product_category)/)
(/RELATE_ID_2_VO(property_definition)/)

5.1.8.20.3 version_relationship to versionable_object (as successor)

AIM element: PATH
Reference path: identification_assignment_relationship
identification_assignment_relationship.relating_assignment ->
(/RELATE_ID_2_VO(document)/)
(/RELATE_ID_2_VO(group)/)

(/RELATE_ID_2_VO(product_definition)/)
(/RELATE_ID_2_VO(product_definition_relationship)/)
(/RELATE_ID_2_VO(product_definition_shape)/)
(/RELATE_ID_2_VO(product_related_product_category)/)
(/RELATE_ID_2_VO(property_definition)/)

5.1.8.21 VERSIONABLE_OBJECT_CHANGE_EVENT

#1: If the Versionable_object_change_event is an Envisaged_version_creation
#2: If the Versionable_object_change_event is a Version_creation
#3: If the Versionable_object_change_event is a Version_modification
#4: If the Versionable_object_change_event is a Version_deletion

AIM element: #1: (action)
#2: (action)
#3: (action)
#4: (action)
Source: ISO 10303-41

5.1.8.21.1 caused_by

AIM element: #1: /SUBTYPE(Envisaged_version_creation)/ -- (see 5.1.8.12.2)
#2: /SUBTYPE(Version_creation)/ -- (see 5.1.8.16.1)
#3: /SUBTYPE(Version_modification)/ -- (see 5.1.8.19.1)
#4: /SUBTYPE(Version_deletion)/ -- (see 5.1.8.17.1)

5.1.8.21.2 caused_when

AIM element: #1: /SUBTYPE(Envisaged_version_creation)/ -- (see 5.1.8.12.3)
#2: /SUBTYPE(Version_creation)/ -- (see 5.1.8.16.2)
#3: /SUBTYPE(Version_modification)/ -- (see 5.1.8.19.2)
#4: /SUBTYPE(Version_deletion)/ -- (see 5.1.8.17.2)

5.1.8.21.3 description

AIM element: #1: /SUBTYPE(Envisaged_version_creation)/ -- (see 5.1.8.12.4)
#2: /SUBTYPE(Version_creation)/ -- (see 5.1.8.16.3)
#3: /SUBTYPE(Version_modification)/ -- (see 5.1.8.19.3)
#4: /SUBTYPE(Version_deletion)/ -- (see 5.1.8.17.3)

5.1.9 Damaged_stability UoF

5.1.9.1 DAMAGE_CASE

AIM element: [property_definition]
[group]
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.181, 5.2.4.213, 5.2.4.232
Reference path: {[/ROOT_CLASS(property_definition, 'damage case')/]
[/LINK_TO_GROUP(property_definition)/]}

5.1.9.1.1 damage_cause

AIM element: descriptive_representation_item.description

Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘damage case parameters’, 'damage cause')/
{(descriptive_representation_item.description = 'collision')
(descriptive_representation_item.description = 'explosion')
(descriptive_representation_item.description = 'grounding')
(descriptive_representation_item.description = 'user_defined')}

5.1.9.1.2 relative_damage_position

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘damage case parameters’, 'relative damage position')/
{(descriptive_representation_item.description = 'above waterline')
(descriptive_representation_item.description = 'below waterline')
(descriptive_representation_item.description = 'on waterline')}

5.1.9.1.3 user_defined

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘damage case parameters’, 'user defined')/

5.1.9.1.4 damage_case to compartment_design_definition (as damaged_compartments)

AIM element: PATH
Reference path: (group <-
/GROUPS(property_definition, ’damage case’)/
{/CLASS_ID(product_definition_shape, 'compartment design definition’)/}

5.1.9.1.5 damage_case to damage_position (as position_of_damage)

AIM element: PATH
Reference path: property_definition
{/CLASS_ID(property_definition, ‘damage case’)/}
property_definition = represented_definition
represented_definition <-
/PDR_NAME('damage position parameters')/

5.1.9.1.6 damage_case to loading_condition_definition (as original_loads)

AIM element: PATH
Reference path: property_definition
{/CLASS_ID(property_definition, ‘damage case’)/}
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘original loads’}
property_definition_relationship.related_definition ->
property_definition
{/CLASS_ID(property_definition, ‘loading condition definition’)/}

5.1.9.2 DAMAGE_POSITION

AIM element: property_definition_representation

Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.144
Reference path: property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘damage position parameters’)/}

5.1.9.2.1 position_accuracy

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'position accuracy'}
representation_item => descriptive_representation_item
{(descriptive_representation_item.description = 'actual')
(descriptive_representation_item.description = 'estimate')}

5.1.9.2.2 damage_position to centre_location (as centre_of_damage)

AIM element: PATH
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
representation_item
{representation_item.name = 'centre of damage'}
representation_item =>
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘centre location’)/}

5.1.9.3 DAMAGE_STABILITY_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, ‘damage stability definition’, ‘stability definition’)/]
[/CLASS(property_definition, ‘stability definition’, ‘design definition’)/]
[/CLASS(property_definition, ‘design definition’, ‘definition’)/]
[/CLASS(property_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(property_definition, ‘versionable object’)/]}

5.1.9.3.1 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.9.3.2 damage_stability_definition to damage_case (as extent_of_damage)

AIM element: PATH

Reference path: property_definition
{/CLASS_ID(property_definition, ‘damage stability definition’)/}
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘extent of damage’}
property_definition_relationship.related_definition ->
property_definition
{/CLASS_ID(property_definition, ‘damage case’)/}

5.1.9.3.3 damage_stability_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.9.3.4 damage_stability_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.9.3.5 damage_stability_definition to named_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.9.3.6 damage_stability_definition to ship (as defined_for)

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/
/PROD_DEF_PRODUCT/
{/CLASS_ID(product, ‘ship’)}

5.1.9.3.7 damage_stability_definition to stability_table (as representations)

AIM element: PATH
Rules: 5.2.4.83
Reference path: /PROP_DEF_TO_REP/
{/CLASS_ID(representation, 'stability table')/}

5.1.9.4 STABILITY_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, ‘stability definition’, ‘design definition’)/]
[/CLASS(property_definition, ‘design definition’, ‘definition’)/]

[/CLASS(property_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(property_definition, ‘versionable object’)/]}

5.1.9.4.1 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.9.4.2 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.9.4.3 stability_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.9.4.4 stability_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.9.4.5 stability_definition to named_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.9.4.6 stability_definition to ship (as defined_for)

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/
/PROD_DEF_PRODUCT/
{/CLASS_ID(product, ‘ship’)/}

5.1.9.4.7 stability_definition to stability_table (as representations)

AIM element: PATH
Rules: 5.2.4.90
Reference path: /PROP_DEF_TO_REP/
{/CLASS_ID(representation, 'stability table')/}

5.1.9.5 STABILITY_PROPERTIES_FOR_ONE_FLOATING_POSITION

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: /ROOT_CLASS(compound_representation_item, 'stability properties for one floating position')/

5.1.9.5.1 stability_properties_for_one_floating_position to centre_location (as centre_of_gravity_above_keel)

AIM element: PATH
Rules: 5.2.4.224, 5.2.4.183
Reference path: /COMPOUND('centre of gravity above keel')/
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘centre location’)/}

5.1.9.5.2 stability_properties_for_one_floating_position to floating_position (as definition_of_starting_floating_position)

AIM element: PATH
Rules: 5.2.4.224
Reference path: /COMPOUND('definition of starting floating position')/
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘floating position’)/}

5.1.9.5.3 stability_properties_for_one_floating_position to stability_property (as stability_properties_for_different_angles_of_heel)

AIM element: PATH
Rules: 5.2.4.223
Reference path: /COMPOUND('stability properties for different angles of heel')/
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘stability property’)/}

5.1.9.5.4 stability_properties_for_one_floating_position to stability_table (as related_stability_table)

AIM element: PATH
Reference path: compound_representation_item <=
representation_item <-
representation.items [i]
representation
{/CLASS_ID(representation, 'stability table')/}

5.1.9.6 STABILITY_PROPERTY

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: /ROOT_CLASS(compound_representation_item, 'stability property')/

5.1.9.6.1 angle_of_heel

AIM element: value_representation_item. value_component
Source: ISO 10303-43
Rules: 5.2.4.225, 5.2.4.183

Reference path: /COMPOUND('angle of heel')/
value_representation_item
{value_representation_item. value_component ->
measure_value = plane_angle_measure}

5.1.9.6.2 righting_arm

AIM element: value_representation_item. value_component
Source: ISO 10303-43
Rules: 5.2.4.225, 5.2.4.183
Reference path: /COMPOUND('righting arm')/
value_representation_item
{value_representation_item. value_component ->
measure_value = positive_length_measure}

5.1.9.6.3 stability_property to centre_location (as centre_of_buoyancy)

AIM element: PATH
Rules: 5.2.4.225, 5.2.4.183
Reference path: /COMPOUND('centre of buoyancy')/
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘centre location’)/}

5.1.9.7 STABILITY_TABLE

AIM element: representation
Source: ISO 10303-43
Reference path: /ROOT_CLASS(representation, 'stability table')/

5.1.9.7.1 mean_shell_thickness

AIM element: value_representation_item. value_component
Source: ISO 10303-43
Rules: 5.2.4.172, 5.2.4.183
Reference path: /REP_TO_VAL_REP_ITEM('mean shell thickness', positive_length_measure)/

5.1.9.7.2 name

AIM element: representation.name
Source: ISO 10303-43

5.1.9.7.3 stability_table to stability_properties_for_one_floating_position (as items)

AIM element: PATH
Rules: 5.2.4.171
Reference path: representation
representation.items [i] ->
representation_item =>
compound_representation_item
{/CLASS_ID(compound_representation_item, 'stability properties for one floating position')/}

5.1.10 Definitions UoF

5.1.10.1 DEFINITION

#1: if definition is a cargo_bay_definition
#2: if definition is a change_definition
#3: if definition is a design_definition
#4: if definition is a design_requirement
#5: if definition is a functional_definition
#6: if definition is a general_characteristics_definition
#7: if definition is a lightship_definition
#8: if definition is a loading_condition_definition
#9: if definition is a local_co_ordinate_system
#10: if definition is a spacing_table
#11: if definition is a tonnage_definition

AIM element: #1: /SUBTYPE(cargo_bay_definition)/ -- (see 5.1.5.1)
#2: /SUBTYPE(change_definition)/ -- (see 5.1.8.6)
#3: /SUBTYPE(design_definition)/ -- (see 5.1.10.2)
#4: /SUBTYPE(design_requirement)/ -- (see 5.1.10.3)
#5: /SUBTYPE(functional_definition)/ -- (see 5.1.10.4)
#6: /SUBTYPE(general_characteristics_definition)/ -- (see 5.1.10.5)
#7: /SUBTYPE(lightship_definition)/ -- (see 5.1.18.6)
#8: /SUBTYPE(loading_condition_definition)/ -- (see 5.1.14.3)
#9: /SUBTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4)
#10: /SUBTYPE(spacing_table)/ -- (see 5.1.15.10)
#11: /SUBTYPE(tonnage_definition)/ -- (see 5.1.21.5)

5.1.10.1.1 description

AIM element: #1: /SUBTYPE(cargo_bay_definition)/ -- (see 5.1.5.1.1)
#2: /SUBTYPE(change_definition)/ -- (see 5.1.8.6.3)
#3: /SUBTYPE(design_definition)/ -- (see 5.1.10.2.1)
#4: /SUBTYPE(design_requirement)/ -- (see 5.1.10.3.1)
#5: /SUBTYPE(functional_definition)/ -- (see 5.1.10.4.1)
#6: /SUBTYPE(general_characteristics_definition)/ -- (see 5.1.10.5.1)
#7: /SUBTYPE(lightship_definition)/ -- (see 5.1.18.6.1)
#8: /SUBTYPE(loading_condition_definition)/ -- (see 5.1.14.3.1)
#9: /SUBTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.1)
#10: /SUBTYPE(spacing_table)/ -- (see 5.1.15.10.1)
#11: /SUBTYPE(tonnage_definition)/ -- (see 5.1.21.5.1)

5.1.10.1.2 version_id

AIM element: #1: /SUBTYPE(cargo_bay_definition)/ -- (see 5.1.5.1.1)
#2: /SUBTYPE(change_definition)/ -- (see 5.1.8.6.3)
#3: /SUBTYPE(design_definition)/ -- (see 5.1.10.2.3)
#4: /SUBTYPE(design_requirement)/ -- (see 5.1.10.3.2)
#5: /SUBTYPE(functional_definition)/ -- (see 5.1.10.4.3)
#6: /SUBTYPE(general_characteristics_definition)/ -- (see 5.1.10.5.2)
#7: /SUBTYPE(lightship_definition)/ -- (see 5.1.18.6.3)
#8: /SUBTYPE(loading_condition_definition)/ -- (see 5.1.14.3.2)
#9: /SUBTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.2)
#10: /SUBTYPE(spacing_table)/ -- (see 5.1.15.10.3)

#11: /SUBTYPE(tonnage_definition)/ -- (see 5.1.21.5.3)

5.1.10.1.3 definition to definable_object (as defined_for)

AIM element:
#1: /SUBTYPE(cargo_bay_definition)/ -- (see 5.1.5.1.3)
#2: /SUBTYPE(change_definition)/ -- (see 5.1.8.6.5)
#3: /SUBTYPE(design_definition)/ -- (see 5.1.10.2.4)
#4: /SUBTYPE(design_requirement)/ -- (see 5.1.10.3.3)
#5: /SUBTYPE(functional_definition)/ -- (see 5.1.10.4.4)
#6: /SUBTYPE(general_characteristics_definition)/ -- (see 5.1.10.5.6)
#7: /SUBTYPE(lightship_definition)/ -- (see 5.1.18.6.9)
#8: /SUBTYPE(loading_condition_definition)/ -- (see 5.1.14.3.9)
#9: /SUBTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.3)
#10: /SUBTYPE(spacing_table)/ -- (see 5.1.15.10.4)
#11: /SUBTYPE(tonnage_definition)/ -- (see 5.1.21.5.12)

5.1.10.1.4 definition to derived_unit (as local_units)

AIM element:
#1: /SUBTYPE(cargo_bay_definition)/ -- (see 5.1.5.1.6)
#3: /SUBTYPE(design_definition)/ -- (see 5.1.10.2.5)
#4: /SUBTYPE(design_requirement)/ -- (see 5.1.10.3.4)
#6: /SUBTYPE(general_characteristics_definition)/ -- (see 5.1.10.5.3)
#7: /SUBTYPE(lightship_definition)/ -- (see 5.1.18.6.5)
#8: /SUBTYPE(loading_condition_definition)/ -- (see 5.1.14.3.5)
#9: /SUBTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.4)
#10: /SUBTYPE(spacing_table)/ -- (see 5.1.15.10.5)
#11: /SUBTYPE(tonnage_definition)/ -- (see 5.1.21.5.6)

5.1.10.1.5 definition to global_id (as id)

AIM element:
#1: /SUBTYPE(cargo_bay_definition)/ -- (see 5.1.5.1.4)
#2: /SUBTYPE(change_definition)/ -- (see 5.1.8.6.6)
#3: /SUBTYPE(design_definition)/ -- (see 5.1.10.2.6)
#4: /SUBTYPE(design_requirement)/ -- (see 5.1.10.3.6)
#5: /SUBTYPE(functional_definition)/ -- (see 5.1.10.4.5)
#6: /SUBTYPE(general_characteristics_definition)/ -- (see 5.1.10.5.4)
#7: /SUBTYPE(lightship_definition)/ -- (see 5.1.18.6.6)
#8: /SUBTYPE(loading_condition_definition)/ -- (see 5.1.14.3.7)
#9: /SUBTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.5)
#10: /SUBTYPE(spacing_table)/ -- (see 5.1.15.10.6)
#11: /SUBTYPE(tonnage_definition)/ -- (see 5.1.21.5.8)

5.1.10.1.6 definition to named_unit (as local_units)

AIM element:
#1: /SUBTYPE(cargo_bay_definition)/ -- (see 5.1.5.1.7)
#3: /SUBTYPE(design_definition)/ -- (see 5.1.10.2.7)
#4: /SUBTYPE(design_requirement)/ -- (see 5.1.10.3.7)
#6: /SUBTYPE(general_characteristics_definition)/ -- (see 5.1.10.5.5)
#7: /SUBTYPE(lightship_definition)/ -- (see 5.1.18.6.8)
#8: /SUBTYPE(loading_condition_definition)/ -- (see 5.1.14.3.8)
#9: /SUBTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.9)
#10: /SUBTYPE(spacing_table)/ -- (see 5.1.15.10.7)
#11: /SUBTYPE(tonnage_definition)/ -- (see 5.1.21.5.10)

5.1.10.2 DESIGN_DEFINITION

#1: if design_definition is a compartment_design_definition
#2: if design_definition is a deck_zone_design_definition
#3: if design_definition is a stability_definition
#4: if design_definition is a zone_design_definition

AIM element: #1: /SUBTYPE(compartment_design_definition)/ -- (see 5.1.5.2)
#2: /SUBTYPE(deck_zone_design_definition)/ -- (see 5.1.5.3)
#3: /SUBTYPE(stability_definition)/ -- (see 5.1.9.4)
#4: /SUBTYPE(zone_design_definition)/ -- (see 5.1.5.4)

5.1.10.2.1 description

AIM element: #1: /SUBTYPE(compartment_design_definition)/ -- (see 5.1.5.2.1)
#2: /SUBTYPE(deck_zone_design_definition)/ -- (see 5.1.5.3.1)
#3: /SUBTYPE(stability_definition)/ -- (see 5.1.9.4.1)
#4: /SUBTYPE(zone_design_definition)/ -- (see 5.1.5.4.1)

5.1.10.2.2 representations

AIM element: #1: /SUBTYPE(compartment_design_definition)/ -- (see 5.1.5.2.9)
#2: /SUBTYPE(deck_zone_design_definition)/ -- (see 5.1.5.3.13)
#3: /SUBTYPE(stability_definition)/ -- (see 5.1.9.4.7)
#4: /SUBTYPE(zone_design_definition)/ -- (see 5.1.5.4.10)

5.1.10.2.3 version_id

AIM element: #1: /SUBTYPE(compartment_design_definition)/ -- (see 5.1.5.2.2)
#2: /SUBTYPE(deck_zone_design_definition)/ -- (see 5.1.5.3.4)
#3: /SUBTYPE(stability_definition)/ -- (see 5.1.9.4.2)
#4: /SUBTYPE(zone_design_definition)/ -- (see 5.1.5.4.2)

5.1.10.2.4 design_definition to definable_object (as defined_for)

AIM element: #1: /SUBTYPE(compartment_design_definition)/ -- (see 5.1.5.2.3)
#2: /SUBTYPE(deck_zone_design_definition)/ -- (see 5.1.5.3.7)
#3: /SUBTYPE(stability_definition)/ -- (see 5.1.9.4.6)
#4: /SUBTYPE(zone_design_definition)/ -- (see 5.1.5.4.11)

5.1.10.2.5 design_definition to derived_unit (as local_units)

AIM element: #1: /SUBTYPE(compartment_design_definition)/ -- (see 5.1.5.2.5)
#2: /SUBTYPE(deck_zone_design_definition)/ -- (see 5.1.5.3.8)
#3: /SUBTYPE(stability_definition)/ -- (see 5.1.9.4.3)
#4: /SUBTYPE(zone_design_definition)/ -- (see 5.1.5.4.5)

5.1.10.2.6 design_definition to global_id (as id)

AIM element: #1: /SUBTYPE(compartment_design_definition)/ -- (see 5.1.5.2.7)
#2: /SUBTYPE(deck_zone_design_definition)/ -- (see 5.1.5.3.11)
#3: /SUBTYPE(stability_definition)/ -- (see 5.1.9.4.4)
#4: /SUBTYPE(zone_design_definition)/ -- (see 5.1.5.4.8)

5.1.10.2.7 design_definition to named_unit (as local_units)

AIM element: #1: /SUBTYPE(compartment_design_definition)/ -- (see 5.1.5.2.8)
#2: /SUBTYPE(deck_zone_design_definition)/ -- (see 5.1.5.3.12)
#3: /SUBTYPE(stability_definition)/ -- (see 5.1.9.4.5)
#4: /SUBTYPE(zone_design_definition)/ -- (see 5.1.5.4.9)

5.1.10.3 DESIGN_REQUIREMENT

#1: if design_requirement is a compartment_design_requirement
#2: if design_requirement is a class_compartment_requirement_definition
#3: if design_requirement is a class_deck_load_requirement_definition

AIM element: #1: /SUBTYPE(compartment_design_requirement)/ -- (see 5.1.7.5)
#2: /SUBTYPE(class_compartment_requirement_definition)/ -- (see 5.1.7.2)
#3: /SUBTYPE(class_deck_load_requirement_definition)/ -- (see 5.1.7.3)

5.1.10.3.1 description

AIM element: #1: /SUBTYPE(compartment_design_requirement)/ -- (see 5.1.7.5.1)
#2: /SUBTYPE(class_compartment_requirement_definition)/ -- (see 5.1.7.2.6)
#3: /SUBTYPE(class_deck_load_requirement_definition)/ -- (see 5.1.7.3.1)

5.1.10.3.2 version_id

AIM element: #1: /SUBTYPE(compartment_design_requirement)/ -- (see 5.1.7.5.4)
#2: /SUBTYPE(class_compartment_requirement_definition)/ -- (see 5.1.7.2.11)
#3: /SUBTYPE(class_deck_load_requirement_definition)/ -- (see 5.1.7.3.5)

5.1.10.3.3 design_requirement to definable_object (as defined_for)

AIM element: #1: /SUBTYPE(compartment_design_requirement)/ -- (see 5.1.7.5.7)
#2: /SUBTYPE(class_compartment_requirement_definition)/ -- (see 5.1.7.2.12)
#3: /SUBTYPE(class_deck_load_requirement_definition)/ -- (see 5.1.7.3.6)

5.1.10.3.4 design_requirement to derived_unit (as local_units)

AIM element: #2: /SUBTYPE(class_compartment_requirement_definition)/ -- (see 5.1.7.2.13)
#3: /SUBTYPE(class_deck_load_requirement_definition)/ -- (see 5.1.7.3.7)

5.1.10.3.5 design_requirement to document_reference (as specification)

AIM element: #1: /SUBTYPE(compartment_design_requirement)/ -- (see 5.1.7.5.5)
#2: /SUBTYPE(class_compartment_requirement_definition)/ -- (see 5.1.7.2.14)
#3: /SUBTYPE(class_deck_load_requirement_definition)/ -- (see 5.1.7.3.8)

5.1.10.3.6 design_requirement to global_id (as id)

AIM element: #1: /SUBTYPE(compartment_design_requirement)/ -- (see 5.1.7.5.6)
#2: /SUBTYPE(class_compartment_requirement_definition)/ -- (see 5.1.7.2.15)
#3: /SUBTYPE(class_deck_load_requirement_definition)/ -- (see 5.1.7.3.9)

5.1.10.3.7 design_requirement to named_unit (as local_units)

AIM element: #2: /SUBTYPE(class_compartment_requirement_definition)/ -- (see 5.1.7.2.16)

#3: /SUBTYPE(class_deck_load_requirement_definition)/ -- (see 5.1.7.3.10)

5.1.10.4 FUNCTIONAL_DEFINITION

#1: if functional_definition is a compartment_functional_definition
#2: if functional_definition is a deck_zone_functional_definition
#3: if functional_definition is a shiptype
#4: if functional_definition is a zone_functional_definition

AIM element: #1: /SUBTYPE(compartment_functional_definition)/ -- (see 5.1.20.2)
#2: /SUBTYPE(deck_zone_functional_definition)/ -- (see 5.1.20.4)
#3: /SUBTYPE(shiptype)/ -- (see 5.1.18.15)
#4: /SUBTYPE(zone_functional_definition)/ -- (see 5.1.20.7)

5.1.10.4.1 description

AIM element: #1: /SUBTYPE(compartment_functional_definition)/ -- (see 5.1.20.2.1)
#2: /SUBTYPE(deck_zone_functional_definition)/ -- (see 5.1.20.4.1)
#3: /SUBTYPE(shiptype)/ -- (see 5.1.18.15.1)
#4: /SUBTYPE(zone_functional_definition)/ -- (see 5.1.20.7.1)

5.1.10.4.2 user_def_function

AIM element: #1: /SUBTYPE(compartment_functional_definition)/ -- (see 5.1.20.2.3)
#2: /SUBTYPE(deck_zone_functional_definition)/ -- (see 5.1.20.4.3)
#3: /SUBTYPE(shiptype)/ -- (see 5.1.18.15.2)
#4: /SUBTYPE(zone_functional_definition)/ -- (see 5.1.20.7.3)

5.1.10.4.3 version_id

AIM element: #1: /SUBTYPE(compartment_functional_definition)/ -- (see 5.1.20.2.4)
#2: /SUBTYPE(deck_zone_functional_definition)/ -- (see 5.1.20.4.4)
#3: /SUBTYPE(shiptype)/ -- (see 5.1.18.15.3)
#4: /SUBTYPE(zone_functional_definition)/ -- (see 5.1.20.7.4)

5.1.10.4.4 functional_definition to definable_object (as defined_for)

AIM element: #1: /SUBTYPE(compartment_functional_definition)/ -- (see 5.1.20.2.5)
#2: /SUBTYPE(deck_zone_functional_definition)/ -- (see 5.1.20.4.5)
#3: /SUBTYPE(shiptype)/ -- (see 5.1.18.15.5)
#4: /SUBTYPE(zone_functional_definition)/ -- (see 5.1.20.7.5)

5.1.10.4.5 functional_definition to global_id (as id)

AIM element: #1: /SUBTYPE(compartment_functional_definition)/ -- (see 5.1.20.2.6)
#2: /SUBTYPE(deck_zone_functional_definition)/ -- (see 5.1.20.4.6)
#3: /SUBTYPE(shiptype)/ -- (see 5.1.18.15.4)
#4: /SUBTYPE(zone_functional_definition)/ -- (see 5.1.20.7.6)

5.1.10.5 GENERAL_CHARACTERISTICS_DEFINITION

#1: if general_characteristics_definition is a class_and_statutory_designation
#2: if general_characteristics_definition is a class_parameters
#3: if general_characteristics_definition is a freeboard_characteristics
#4: if general_characteristics_definition is a global_axis_placement
#5: if general_characteristics_definition is an owner_designation

#6: if general_characteristics_definition is a principal_characteristics
#7: if general_characteristics_definition is a ship_designation
#8: if general_characteristics_definition is a shipyard_designation

AIM element: #1: /SUBTYPE(class_and_statutory_designation)/ -- (see 5.1.18.2)
#2: /SUBTYPE(class_parameters)/ -- (see 5.1.18.4)
#3: /SUBTYPE(freeboard_characteristics)/ -- (see 5.1.18.5)
#4: /SUBTYPE(global_axis_placement)/ -- (see 5.1.15.3)
#5: /SUBTYPE(owner_designation)/ -- (see 5.1.18.10)
#6: /SUBTYPE(principal_characteristics)/ -- (see 5.1.18.11)
#7: /SUBTYPE(ship_designation)/ -- (see 5.1.18.14)
#8: /SUBTYPE(shipyard_designation)/ -- (see 5.1.18.16)

5.1.10.5.1 description

AIM element: #1: /SUBTYPE(class_and_statutory_designation)/ -- (see 5.1.18.2.2)
#2: /SUBTYPE(class_parameters)/ -- (see 5.1.18.4.2)
#3: /SUBTYPE(freeboard_characteristics)/ -- (see 5.1.18.5.3)
#4: /SUBTYPE(global_axis_placement)/ -- (see 5.1.15.3.2)
#5: /SUBTYPE(owner_designation)/ -- (see 5.1.18.10.1)
#6: /SUBTYPE(principal_characteristics)/ -- (see 5.1.18.11.2)
#7: /SUBTYPE(ship_designation)/ -- (see 5.1.18.14.2)
#8: /SUBTYPE(shipyard_designation)/ -- (see 5.1.18.16.1)

5.1.10.5.2 version_id

AIM element: #1: /SUBTYPE(class_and_statutory_designation)/ -- (see 5.1.18.2.3)
#2: /SUBTYPE(class_parameters)/ -- (see 5.1.18.4.8)
#3: /SUBTYPE(freeboard_characteristics)/ -- (see 5.1.18.5.6)
#4: /SUBTYPE(global_axis_placement)/ -- (see 5.1.15.3.4)
#5: /SUBTYPE(owner_designation)/ -- (see 5.1.18.10.6)
#6: /SUBTYPE(principal_characteristics)/ -- (see 5.1.18.11.12)
#7: /SUBTYPE(ship_designation)/ -- (see 5.1.18.14.7)
#8: /SUBTYPE(shipyard_designation)/ -- (see 5.1.18.16.6)

5.1.10.5.3 general_characteristics_definition to derived_unit (as local_units)

AIM element: #2: /SUBTYPE(class_parameters)/ -- (see 5.1.18.4.9)
#3: /SUBTYPE(freeboard_characteristics)/ -- (see 5.1.18.5.7)
#4: /SUBTYPE(global_axis_placement)/ -- (see 5.1.15.3.5)
#6: /SUBTYPE(principal_characteristics)/ -- (see 5.1.18.11.13)

5.1.10.5.4 general_characteristics_definition to global_id (as id)

AIM element: #1: /SUBTYPE(class_and_statutory_designation)/ -- (see 5.1.18.2.5)
#2: /SUBTYPE(class_parameters)/ -- (see 5.1.18.4.10)
#3: /SUBTYPE(freeboard_characteristics)/ -- (see 5.1.18.5.8)
#4: /SUBTYPE(global_axis_placement)/ -- (see 5.1.15.3.6)
#5: /SUBTYPE(owner_designation)/ -- (see 5.1.18.10.7)
#6: /SUBTYPE(principal_characteristics)/ -- (see 5.1.18.11.14)
#7: /SUBTYPE(ship_designation)/ -- (see 5.1.18.14.8)
#8: /SUBTYPE(shipyard_designation)/ -- (see 5.1.18.16.7)

5.1.10.5.5 general_characteristics_definition to named_unit (as local_units)

AIM element: #2: /SUBTYPE(class_parameters)/ -- (see 5.1.18.4.11)
#3: /SUBTYPE(freeboard_characteristics)/ -- (see 5.1.18.5.10)
#4: /SUBTYPE(global_axis_placement)/ -- (see 5.1.15.3.7)
#6: /SUBTYPE(principal_characteristics)/ -- (see 5.1.18.11.15)

5.1.10.5.6 general_characteristics_definition to ship (as defined_for)

AIM element: #1: /SUBTYPE(class_and_statutory_designation)/ -- (see 5.1.18.2.7)
#2: /SUBTYPE(class_parameters)/ -- (see 5.1.18.4.12)
#3: /SUBTYPE(freeboard_characteristics)/ -- (see 5.1.18.5.11)
#4: /SUBTYPE(global_axis_placement)/ -- (see 5.1.15.3.8)
#5: /SUBTYPE(owner_designation)/ -- (see 5.1.18.10.8)
#6: /SUBTYPE(principal_characteristics)/ -- (see 5.1.18.11.16)
#7: /SUBTYPE(ship_designation)/ -- (see 5.1.18.14.9)
#8: /SUBTYPE(shipyard_designation)/ -- (see 5.1.18.16.8)

5.1.11 External_references UoF

5.1.11.1 DOCUMENT

AIM element: document
Source: ISO 10303-41
Rules: 5.2.4.37
Reference path: {[/CLASS(document, ‘document’, ‘versionable object’)/]
[/ROOT_CLASS(document, ‘versionable object’)/]}

5.1.11.1.1 author

#1: if author is a person
#2: if author is an organization
#3: if author is a person_and_organization

AIM element: #1: applied_person_assignment.assigned_person
#2: applied_organization_assignment.assigned_organization
#3:
applied_person_and_organization_assignment.assigned_person_and_organization
Source: #1: ISO 10303-215
#2: ISO 10303-215
#3: ISO 10303-215
Reference path: #1: /PERS_ASSGN(document, ‘author’)/
#2: /ORG_ASSGN(document, ‘author’)/
#3: /PERS_ORG_ASSGN(document, ‘author’)/

5.1.11.1.2 description

AIM element: document.description
Source: ISO 10303-41

5.1.11.1.3 source_type

AIM element: document_representation_type.name
Source: ISO 10303-41
Rules: 5.2.4.36

Reference path: document <-
document_representation_type.represented_document
document_representation_type
document_representation_type.name

5.1.11.1.4 title

AIM element: document.name
Source: ISO 10303-41

5.1.11.1.5 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(document)/

5.1.11.2 DOCUMENT_PORTION

AIM element: document_usage_constraint
Source: ISO 10303-41

5.1.11.2.1 element_type

AIM element: document_usage_constraint.subject_element
Source: ISO 10303-41

5.1.11.2.2 element_value

AIM element: document_usage_constraint.subject_element_value
Source: ISO 10303-41

5.1.11.2.3 document_portion to document (as source)

AIM element: PATH
Reference path: document_usage_constraint
document_usage_constraint.source ->
document

5.1.11.3 DOCUMENT_REFERENCE

AIM element: document
Source: ISO 10303-41
Reference path: {/ROOT_CLASS(document, ‘document reference’)/}

5.1.11.3.1 document_reference to document (as assigned_document)

AIM element: PATH
Reference path: document

5.1.11.3.2 document_reference to document_portion (as assigned_document)

AIM element: PATH
Reference path: document <-
document_usage_constraint.source

document_usage_constraint

5.1.11.4 DOCUMENT_REFERENCE_WITH_ADDRESS

AIM element: document
Source: ISO 10303-41
Rules: 5.2.4.38
Reference path: {[/CLASS(document,‘document reference with address’, ‘document reference’)/]
[/ROOT_CLASS(document, ‘document reference’)/]
[/CLASS(document, ‘document reference with address’, ‘external reference’)/]
[/ROOT_CLASS(document, ‘external reference’)/]}

5.1.11.4.1 description

AIM element: identification_role.description
Source: ISO 10303-41
Reference path: document
external_identification_item = document
external_identification_item <-
applied_external_identification_assignment.items[i]
applied_external_identification_assignment <=
external_identification_assignment <=
identification_assignment
identification_assignment.role ->
identificaton_role
{identification_role.name = 'external reference'}
identification_role.description

5.1.11.4.2 document_reference_with_address to document (as assigned_document)

AIM element: PATH
Reference path: document

5.1.11.4.3 document_reference_with_address to document_portion (as assigned_document)

AIM element: PATH
Reference path: document <-
document_usage_constraint.source
document_usage_constraint

5.1.11.4.4 document_reference_with_address to external_storage (as location)

AIM element: PATH
Reference path: document
external_identification_item = document
external_identification_item <-
applied_external_identification_assignment.items[i]
applied_external_identification_assignment <=
external_identification_assignment
[external_identification_assignment.source ->
external_source
{/CLASS_ID(external_source,'external storage')/}]
[external_identification_assignment <=
identification_assignment
identification_assignment.role ->

identificaton_role
{identification_role.name = 'external reference'}]

5.1.11.4.5 document_reference_with_address to universal_resource_locator (as location)

AIM element: PATH
Reference path: document
external_identification_item = document
external_identification_item <-
applied_external_identification_assignment.items[i]
applied_external_identification_assignment <=
external_identification_assignment
[external_identification_assignment.source ->
external_source
{/CLASS_ID(external_source,'universal resource locator')/}]
[external_identification_assignment <=
identification_assignment
identification_assignment.role ->
identificaton_role
{identification_role.name = 'external reference'}]

5.1.11.5 EXTERNAL_INSTANCE_REFERENCE

AIM element: applied_external_identification_assignment
Source: ISO 10303-215
Reference path: applied_external_identification_assignment <=
external_identification_assignment <=
{/ID_ROLE('external instance reference')/}

5.1.11.5.1 entity_type

AIM element: external_source.source_id
Source: ISO 10303-41
Rules: 5.2.4.49
Reference path: applied_external_identification_assignment <=
external_identification_assignment
external_identification_assignment.source ->
external_source
{/DESCRIPTION_ASSGN(external_source)/
description_attribute.attribute_value = ‘schema name’}
external_source_relationship.relating_source <-
external_source_relationship
{external_source_relationship.name = ‘composition’}
external_source_relationship.related_source ->
external_source
{/DESCRIPTION_ASSGN(external_source)/
description_attribute.attribute_value = ‘entity type’ }
external_source.source_id ->
source_item
source_item = identifier

5.1.11.5.2 schema_name

AIM element: external_source.source_id

Source: ISO 10303-41
Reference path: applied_external_identification_assignment <=
external_identification_assignment
external_identification_assignment.source ->
external_source
{/DESCRIPTION_ASSGN(external_source)/
description_attribute.attribute_value = ‘schema name’}
external_source.source_id ->
source_item
source_item = identifier

5.1.11.5.3 external_instance_reference to global_id (as target_guid)

AIM element: PATH
Reference path: applied_external_identification_assignment <=
external_identification_assignment <=
identification_assignment
identification_assignment.assigned_id

5.1.11.6 EXTERNAL_REFERENCE

AIM element: applied_external_identification_assignment
Source: ISO 10303-215
Reference path: applied_external_identification_assignment <=
external_identification_assignment <=
identification_assignment
identification_assignment.role ->
identificaton_role
{identification_role.name = 'external reference'}

5.1.11.6.1 description

AIM element: identification_role.description
Source: ISO 10303-41
Rules: 5.2.4.46
Reference path: applied_external_identification_assignment <=
external_identification_assignment <=
identification_assignment
identification_assignment.role ->
identificaton_role
identification_role.description

5.1.11.6.2 external_reference to external_storage (as location)

AIM element: PATH
Reference path: applied_external_identification_assignment <=
external_identification_assignment
external_identification_assignment.source ->
external_source
{/CLASS_ID(external_source, 'external storage')/}

5.1.11.6.3 external_reference to universal_resource_locator (as location)

AIM element: PATH
Reference path: applied_external_identification_assignment <=

external_identification_assignment
external_identification_assignment.source ->
external_source
{/CLASS_ID(external_source, 'universal resource locator')/}

5.1.11.7 EXTERNAL_STORAGE

AIM element: external_source
Source: ISO 10303-41
Reference path: /ROOT_CLASS(external_source, 'external storage')/

5.1.11.7.1 location

AIM element: external_source.source_id
Source: ISO 10303-41
Reference path: external_source
external_source.source_id ->
source_item
source_item = identifier

5.1.11.8 UNIVERSAL_RESOURCE_LOCATOR

AIM element: external_source
Source: ISO 10303-41
Reference path: {/ROOT_CLASS(external_source, 'universal resource locator')/}

5.1.11.8.1 location

AIM element: external_source.source_id
Source: ISO 10303-41
Reference path: external_source
external_source.source_id ->
source_item
source_item = identifier}

5.1.12 Hull_class_applicability UoF

5.1.12.1 HULL_APPLICABILITY

AIM element: serial_numbered_effectivity
Source: ISO 10303-41

5.1.12.1.1 end_hull

AIM element: serial_numbered_effectivity.effectivity_end_id
Source: ISO 10303-41

5.1.12.1.2 start_hull

AIM element: serial_numbered_effectivity.effectivity_start_id
Source: ISO 10303-41

5.1.12.1.3 hull_applicability to definition (as definitions_for_hulls)

AIM element: PATH
Reference path: serial_numbered_effectivity <=
effectivity
{effectivity.id = 'hull applicability'}
effectivity <-
effectivity_assignment.assigned_effectivity
effectivity_assignment
/ROLE_ASSGN(effectivity_assignment)/
{object_role.name = 'definitions for hulls'}
effectivity_assignment =>
applied_effectivity_assignment
applied_effectivity_assignment.items[i] ->
effectivity_item
(effectivity_item = product_definition
{CLASS_ID(product_definition, 'definition')})
(effectivity_item = property_definition
{CLASS_ID(property_definition, 'definition')})
(effectivity_item = product_definition_shape
{CLASS_ID(product_definition_shape, 'definition')})
(effectivity_item = product_related_product_category
{CLASS_ID(product_related_product_category, 'definition')})

5.1.12.1.4 hull_applicability to item (as items_for_hulls)

AIM element: PATH
Reference path: serial_numbered_effectivity <=
effectivity
{effectivity.id = 'hull applicability'}
effectivity <-
effectivity_assignment.assigned_effectivity
effectivity_assignment
/ROLE_ASSGN(effectivity_assignment)/
{object_role.name = 'items for hulls'}
effectivity_assignment =>
applied_effectivity_assignment
applied_effectivity_assignment.items[i] ->
effectivity_item
(effectivity_item = product_definition
{CLASS_ID(product_definition, 'item')})

5.1.13 Items UoF

5.1.13.1 DEFINABLE_OBJECT

#1: if definable_object is an item
#2: if definable_object is an item_relationship
#3: if definable_object is an item_structure

AIM element: #1: /SUBTYPE(item)/ -- (see 5.1.13.3)
#2: /SUBTYPE(item_relationship)/ -- (see 5.1.13.4)
#3: /SUBTYPE(item_structure)/ -- (see 5.1.13.5)

5.1.13.1.1 definable_object to global_id (as id)

AIM element: #1: /SUBTYPE(item)/ -- (see 5.1.13.3.4)
#2: /SUBTYPE(item_relationship)/ -- (see 5.1.13.4.5)
#3: /SUBTYPE(item_structure)/ -- (see 5.1.13.5.4)

5.1.13.2 GLOBAL_ID

AIM element: applied_identification_assignment
Source: ISO 10303-215
Reference path: applied_identification_assignment <=
identification_assignment
{identification_assignment.role ->
identification_role
identification_role.name = 'globally unambiguous identifier'}

5.1.13.2.1 id

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Reference path: applied_identification_assignment <=
identification_assignment
identification_assignment.assigned_id

5.1.13.3 ITEM

#1: if item is a change
#2: if item is a ship
#3: if item is a space
#4: if item is a space_product_structure

AIM element: #1: /SUBTYPE(change)/ -- (see 5.1.8.5)
#2: /SUBTYPE(ship)/ -- (see 5.1.13.6)
#3: /SUBTYPE(space)/ -- (see 5.1.20.5)
#4: /SUBTYPE(space_product_structure)/ -- (see 5.1.16.1)

5.1.13.3.1 description

AIM element: #1: /SUBTYPE(change)/ -- (see 5.1.8.5.1)
#2: /SUBTYPE(ship)/ -- (see 5.1.13.6.1)
#3: /SUBTYPE(space)/ -- (see 5.1.20.5.1)
#4: /SUBTYPE(space_product_structure)/ -- (see 5.1.16.1.1)

5.1.13.3.2 name

AIM element: #1: /SUBTYPE(change)/ -- (see 5.1.8.5.2)
#2: /SUBTYPE(ship)/ -- (see 5.1.13.6.2)
#3: /SUBTYPE(space)/ -- (see 5.1.20.5.2)
#4: /SUBTYPE(space_product_structure)/ -- (see 5.1.16.1.2)

5.1.13.3.3 item to external_reference (as documentation)

AIM element: #1: /SUBTYPE(change)/ -- (see 5.1.8.5.4)
#2: /SUBTYPE(ship)/ -- (see 5.1.13.7.2)

#3: /SUBTYPE(space)/ -- (see 5.1.20.5.3)
#4: /SUBTYPE(space_product_structure)/ -- (see 5.1.16.1.5)

5.1.13.3.4 item to global_id (as id)

AIM element: #1: /SUBTYPE(change)/ -- (see 5.1.8.5.5)
#2: /SUBTYPE(ship)/ -- (see 5.1.13.7.3)
#3: /SUBTYPE(space)/ -- (see 5.1.20.5.4)
#4: /SUBTYPE(space_product_structure)/ -- (see 5.1.16.1.7)

5.1.13.3.5 item to ship (as ship_context)

AIM element: #1: /SUBTYPE(change)/ -- (see 5.1.8.5.6)
#3: /SUBTYPE(space)/ -- (see 5.1.20.5.5)
#4: /SUBTYPE(space_product_structure)/ -- (see 5.1.16.1.9)

5.1.13.4 ITEM_RELATIONSHIP

#1: if item_relationship is a space_arrangement_relationship

AIM element: #1: /SUBTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3)

5.1.13.4.1 description

AIM element: #1: /SUBTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.1)

5.1.13.4.2 version_id

AIM element: #1: /SUBTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.2)

5.1.13.4.3 item_relationship to external_instance_reference (as external_item_1)

AIM element: #1: /SUBTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.3)

5.1.13.4.4 item_relationship to external_instance_reference (as external_item_2)

AIM element: #1: /SUBTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.4)

5.1.13.4.5 item_relationship to global_id (as id)

AIM element: #1: /SUBTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.5)

5.1.13.4.6 item_relationship to item (as item_1)

AIM element: #1: /SUBTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.6)

5.1.13.4.7 item_relationship to item (as item_2)

AIM element: #1: /SUBTYPE(space_arrangement_relationship)/ -- (see 5.1.2.3.7)

5.1.13.5 ITEM_STRUCTURE

#1: if item_structure is a space_product_structure

AIM element: #1: /SUBTYPE(space_product_structure)/ -- (see 5.1.16.1)

5.1.13.5.1 description

AIM element: #1: /SUBTYPE(space_product_structure)/ -- (see 5.1.16.1.1)

5.1.13.5.2 version_id

AIM element: #1: /SUBTYPE(space_product_structure)/ -- (see 5.1.16.1.4)

5.1.13.5.3 item_structure to external_instance_reference (as external_items)

AIM element: #1: /SUBTYPE(space_product_structure)/ -- (see 5.1.16.1.6)

5.1.13.5.4 item_structure to global_id (as id)

AIM element: #1: /SUBTYPE(space_product_structure)/ -- (see 5.1.16.1.7)

5.1.13.5.5 item_structure to item (as items)

AIM element: #1: /SUBTYPE(space_product_structure)/ -- (see 5.1.16.1.8)

5.1.13.6 SHIP

AIM element: product
Source: ISO 10303-41
Reference path: {[/CLASS(product, 'ship', 'item')/]
[/CLASS(product, 'item', 'definable object')/]
[/ROOT_CLASS(product, 'definable object')/]}

5.1.13.6.1 description

AIM element: product.description
Source: ISO 10303-41

5.1.13.6.2 name

AIM element: product.name
Source: ISO 10303-41

5.1.13.7 single_hull_or_class

AIM element: product_context
Source: ISO 10303-41
Reference path: product
product.frame_of_reference ->
product_context
{(product_context.discipline_type = 'design for single hull')
(product_context.discipline_type = 'design for multiple hulls')}

5.1.13.7.1 ship to derived_unit (as units)

AIM element: PATH
Reference path: product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition.formation

/PROD_DEF_TO_UNITS('global units')/
unit
unit = derived_unit
derived_unit

5.1.13.7.2 ship to external_reference (as documentation)

#1: If as documentation refers to an External_reference

AIM element: PATH
Reference path: product
product = external_identification_item
external_identification_item <-
applied_external_identification_assignment.items[i]
applied_external_identification_assignment

#2: If as documentation refers to a Document_reference_with_address

AIM element: PATH
Reference path: product
/DOC_REF(product,'documentation')/
document
{/CLASS_ID(document, 'document reference with address')/}

5.1.13.7.3 ship to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.73
Reference path: product
identification_item = product <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.13.8 ship to item (as ship_items)

#1: if item is a space or space_product_structure
#2: if item is a change

AIM element: #1: PATH
#2: PATH

Reference path: #1: product <-
product_definition_formation.of_product
product_definition_formation <-
product_definition.formation
product_definition

#2: product
action_item = product
action_item <-
applied_action_assignment.items[i]
applied_action_assignment
applied_action_assignment.assigned_action ->
action

5.1.13.8.1 ship to named_unit (as units)

AIM element: PATH
Reference path: product <-
product_definition_formation.of_product
product_definition_formation<-
product_definition.formation
/PROD_DEF_TO_UNITS('global units')/
unit
unit = named_unit
named_unit

5.1.13.9 VERSIONABLE_OBJECT

#1: if versionable_object is a definition
#2: if versionable_object is a document
#3: if versionable_object is an item_relationship
#4: if versionable_object is an item_structure
#5: if versionable_object is a revision

AIM element: #1: /SUBTYPE(definition)/ -- (see 5.1.10.1)
#2: /SUBTYPE(document)/ -- (see 5.1.11.1)
#3: /SUBTYPE(item_relationship)/ -- (see 5.1.13.4)
#4: /SUBTYPE(item_structure)/ -- (see 5.1.13.5)
#5: /SUBTYPE(revision)/ -- (see 5.1.8.14)

5.1.13.9.1 description

AIM element: #1: /SUBTYPE(definition)/ -- (see 5.1.10.1.1)
#2: /SUBTYPE(document)/ -- (see 5.1.11.1.2)
#3: /SUBTYPE(item_relationship)/ -- (see 5.1.13.4.1)
#4: /SUBTYPE(item_structure)/ -- (see 5.1.13.5.1)
#5: /SUBTYPE(revision)/ -- (see 5.1.8.14.1)

5.1.13.9.2 version_id

AIM element: #1: /SUBTYPE(definition)/ -- (see 5.1.10.1.2)
#2: /SUBTYPE(document)/ -- (see 5.1.11.1.5)
#3: /SUBTYPE(item_relationship)/ -- (see 5.1.13.4.2)
#4: /SUBTYPE(item_structure)/ -- (see 5.1.13.5.2)
#5: /SUBTYPE(revision)/ -- (see 5.1.8.14.3)

5.1.14 Loading_conditions UoF

5.1.14.1 DEADWEIGHT

AIM element: property_definition_representation
Source: ISO 10303-41
Reference path: property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation,
'deadweight')/}

5.1.14.1.1 deadweight_value

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('deadweight value', mass_measure)/

5.1.14.1.2 deadweight to cargo_assignment (as deadweight_items)

AIM element: PATH
Reference path: property_definition_representation
property_definition_representation.definition ->
represented_definition = property_definition
property_definition
{property_definition.name = ‘deadweight items’}
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition_relationship
product_definition_relationship
{(/CLASS_ID(product_definition_relationship‘bulk cargo assignment’)/)
(/CLASS_ID(product_definition_relationship‘deck cargo assignment’)/)
(/CLASS_ID(product_definition_relationship‘gaseous cargo assignment’)/)
(/CLASS_ID(product_definition_relationship‘liquid cargo assignment’)/)
(/CLASS_ID(product_definition_relationship‘unit cargo assignment’)/)}

5.1.14.2 FLOATING_POSITION

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: /ROOT_CLASS(compound_representation_item, 'floating position')/

5.1.14.2.1 angle_of_heel

AIM element: value_representation_item. value_component
Source: ISO 10303-41
Rules: 5.2.4.183,5.2.4.42
Reference path: /COMPOUND('angle of heel')/
value_representation_item
{value_representation_item. value_component ->
measure_value = plane_angle_measure}

5.1.14.2.2 angle_of_trim

AIM element: value_representation_item. value_component
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.42
Reference path: /COMPOUND('angle of trim')/
value_representation_item
{value_representation_item. value_component ->
measure_value = plane_angle_measure}

5.1.14.2.3 breadth_of_waterline

AIM element: value_representation_item. value_component

Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.42
Reference path: /COMPOUND('breadth of waterline')/
value_representation_item
{value_representation_item. value_component ->
measure_value = positive_length_measure}

5.1.14.2.4 draught_at_amidships

AIM element: value_representation_item. value_component
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.42
Reference path: /COMPOUND('draught at amidships')/
value_representation_item
{value_representation_item. value_component ->
measure_value = positive_length_measure}

5.1.14.2.5 length_of_waterline

AIM element: value_representation_item. value_component
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.42
Reference path: /COMPOUND('length of waterline')/
value_representation_item
{value_representation_item. value_component ->
measure_value = positive_length_measure}

5.1.14.2.6 moulded_form_displacement

AIM element: value_representation_item. value_component
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.42
Reference path: /COMPOUND('moulded form displacement')/
value_representation_item
{value_representation_item. value_component ->
measure_value = volume_measure}

5.1.14.3 LOADING_CONDITION_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, ‘loading condition definition’, ‘definition’)/]
[/CLASS(property_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(property_definition, ‘versionable object’)/]}

5.1.14.3.1 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.14.3.2 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215

Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.14.3.3 loading_condition_definition to cargo_assignment (as cargo_loads)

AIM element: PATH
Reference path: property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition_relationship
product_definition_relationship
{[product_definition_relationship.name = 'cargo_loads']
[(/CLASS_ID(product_definition_relationship,'bulk cargo assignment')/)
(/CLASS_ID(product_definition_relationship, 'deck cargo assignment')/)
(/CLASS_ID(product_definition_relationship 'gaseous cargo assignment')/)
(/CLASS_ID(product_definition_relationship, 'liquid cargo assignment')/)
(/CLASS_ID(product_definition_relationship, 'unit cargo assignment')/)]}

5.1.14.3.4 loading_condition_definition to deadweight (as deadweight)

AIM element: PATH
Reference path: property_definition
represented_definition = property_definition
represented_definition <-
property_definition_representation.definition
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation,
'deadweight')/}

5.1.14.3.5 loading_condition_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.14.3.6 loading_condition_definition to floating_position (as floating_position)

AIM element: PATH
Reference path: property_definition
represented_definition = property_definition
represented_definition <-
property_definition_representation.definition
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, 'floating
position parameters')/}
property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
compound_representation_item
{/CLASS_ID(compound_representation_item, 'floating position')/}

5.1.14.3.7 loading_condition_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.14.3.8 loading_condition_definition to named_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.14.3.9 loading_condition_definition to ship (as defined_for)

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/
/PROD_DEF_PRODUCT/
{/CLASS_ID(product, ‘ship’)/}

5.1.14.4 LOADING_CONDITION_DESIGN_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.159
Reference path: {[/CLASS(property_definition, ‘loading condition design definition’, ‘loading condition definition’)/]
[/CLASS(property_definition, ‘loading condition definition’, ‘definition’)/]
[/CLASS(property_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(property_definition, ‘versionable object’)/]}

5.1.14.4.1 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.14.4.2 type_of

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('loading condition design definition parameters', 'type of')/
{(descriptive_representation_item.description = 'actual')
(descriptive_representation_item.description = 'expected')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'minimum')
(descriptive_representation_item.description = 'other')}

5.1.14.4.3 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.14.4.4 loading_condition_design_definition to cargo_assignment (as cargo_loads)

AIM element: PATH
Reference path: property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition_relationship
product_definition_relationship
{[product_definition_relationship.name = 'cargo_loads']
{[(/CLASS_ID(product_definition_relationship,'bulk cargo assignment')/)
(/CLASS_ID(product_definition_relationship, 'deck cargo assignment')/)
(/CLASS_ID(product_definition_relationship 'gaseous cargo assignment')/)
(/CLASS_ID(product_definition_relationship, 'liquid cargo assignment')/)
(/CLASS_ID(product_definition_relationship, 'unit cargo assignment')/)]}

5.1.14.4.5 loading_condition_design_definition to deadweight (as deadweight)

AIM element: PATH
Reference path: property_definition
represented_definition = property_definition
represented_definition <-
property_definition_representation.definition
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation,
'deadweight')/}

5.1.14.4.6 loading_condition_design_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.14.4.7 loading_condition_design_definition to floating_position (as floating_position)

AIM element: PATH
Reference path: property_definition
represented_definition = property_definition
represented_definition <-
property_definition_representation.definition
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, 'floating
position parameters')/}
property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->

compound_representation_item
{/CLASS_ID(compound_representation_item, ‘floating position’)/}

5.1.14.4.8 loading_condition_design_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.14.4.9 loading_condition_design_definition to named_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.14.4.10 loading_condition_design_definition to ship (as defined_for)

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/
/PROD_DEF_PRODUCT/
{/CLASS_ID(product, ‘ship’)/}

5.1.14.5 LOADING_CONDITION_OPERATING_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.160, 5.2.4.205
Reference path: {[/CLASS(property_definition, ‘loading condition operating definition’, ‘loading condition definition’)/]
[/CLASS(property_definition, ‘loading condition definition’, ‘definition’)/]
[/CLASS(property_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(property_definition, ‘versionable object’)/]}

5.1.14.5.1 date_of_loading

AIM element: date_and_time
Source: ISO 10303-41
Rules: 5.2.4.84
Reference path: /DAT_TIME_ASSGN(property_definition, ‘date of loading’)/

5.1.14.5.2 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.14.5.3 place_of_loading

AIM element: descriptive_representation_item.description

Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('loading condition operating definition parameters', 'place of loading')/

5.1.14.5.4 type_of

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('loading condition operating definition parameters', 'type of')/
{(descriptive_representation_item.description = 'actual')
(descriptive_representation_item.description = 'expected')
(descriptive_representation_item.description = 'maximum')
(descriptive_representation_item.description = 'minimum')
(descriptive_representation_item.description = 'other')}

5.1.14.5.5 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.14.5.6 loading_condition_operating_definition to cargo_assignment (as cargo_loads)

AIM element: PATH
Reference path: property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition_relationship
product_definition_relationship
{[product_definition_relationship.name = ‘cargo_loads’]
{[(/CLASS_ID(product_definition_relationship,‘bulk cargo assignment’)/)
(/CLASS_ID(product_definition_relationship, ‘deck cargo assignment’)/)
(/CLASS_ID(product_definition_relationship ‘gaseous cargo assignment’)/)
(/CLASS_ID(product_definition_relationship, ‘liquid cargo assignment’)/)
(/CLASS_ID(product_definition_relationship, ‘unit cargo assignment’)/)]}

5.1.14.5.7 loading_condition_operating_definition to deadweight (as deadweight)

AIM element: PATH
Reference path: property_definition
represented_definition = property_definition
represented_definition <-
property_definition_representation.definition
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘deadweight’)/}

5.1.14.5.8 loading_condition_operating_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit

derived_unit

5.1.14.5.9 loading_condition_operating_definition to floating_position (as floating_position)

AIM element: PATH
Reference path: property_definition
represented_definition = property_definition
represented_definition <-
property_definition_representation.definition
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘floating position parameters’)/}
property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘floating position’)/}

5.1.14.5.10 loading_condition_operating_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.14.5.11 loading_condition_operating_definition to named_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.14.5.12 loading_condition_operating_definition to ship (as defined_for)

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/
/PROD_DEF_PRODUCT/
{/CLASS_ID(product, ‘ship’)/}

5.1.15 Location_concepts UoF

5.1.15.1 BUTTOCK_TABLE

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, ‘buttock table’, ‘transversal table’)/]
[/CLASS(property_definition, ‘transversal table’, ‘spacing table’)/]
[/CLASS(property_definition, ‘spacing table’, ‘definition’)/]
[/CLASS(property_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(property_definition, ‘versionable object’)/]}

5.1.15.1.1 description

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.1)

5.1.15.1.2 name

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.2)

5.1.15.1.3 version_id

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.3)

5.1.15.1.4 buttock_table to definable_object (as defined_for)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.4)

5.1.15.1.5 buttock_table to derived_unit (as local_units)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.5)

5.1.15.1.6 buttock_table to global_id (as id)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.6)

5.1.15.1.7 buttock_table to named_unit (as local_units)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.7)

5.1.15.1.8 buttock_table to transversal_position (as spacing_table_representations)

AIM element: /SUPERTYPE(transversal_table)/ -- (see 5.1.15.13.8)

5.1.15.2 FRAME_TABLE

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, ‘frame table’, ‘longitudinal table’)/]
[/CLASS(property_definition, ‘longitudinal table’, ‘spacing table’)/]
[/CLASS(property_definition, ‘spacing table’, ‘definition’)/]
[/CLASS(property_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(property_definition, ‘versionable object’)/]}

5.1.15.2.1 description

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.1)

5.1.15.2.2 name

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.2)

5.1.15.2.3 version_id

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.3)

5.1.15.2.4 frame_table to definable_object (as defined_for)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.4)

5.1.15.2.5 frame_table to derived_unit (as local_units)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.5)

5.1.15.2.6 frame_table to global_id (as id)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.6)

5.1.15.2.7 frame_table to longitudinal_position (as spacing_table_representations)

AIM element: /SUPERTYPE(longitudinal_table)/ -- (see 5.1.15.7.7)

5.1.15.2.8 frame_table to named_unit (as local_units)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.7)

5.1.15.3 GLOBAL_AXIS_PLACEMENT

AIM element: [product_definition]
[axis2_placement_3d]
Source: ISO 10303-41
ISO 10303-42
Rules: 5.2.4.44, 5.2.4.153, 5.2.4.183
Reference path: {[/CLASS(product_definition, ‘global axis placement’, ‘general characteristics definition’)/]
[/CLASS(product_definition, ‘general characteristics definition’, ‘definition’)/]
[/CLASS(product_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(product_definition, ‘versionable object’)/]}
/PROD_DEF_TO_REP(‘global axis placement’)/
representation
{{representation.name = ‘global axis representation‘}
representation.context_of_items ->
representation_context =>
{representation_context.context_type = ‘global coordinate space’}
geometric_representation_context
{geometric_representation_context.coordinate_space_dimension = 3}}
/REP_ITEM(’global axes and origin’)/
geometric_representation_item =>
placement =>
axis2_placement_3d

5.1.15.3.1 after_perpendicular_offset

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.153, 5.2.4.183
Reference path: /PROD_DEF_TO_REP(‘global axis placement’)/
representation
{{representation.name = ‘global axis representation‘}
representation.context_of_items ->
representation_context =>
{representation_context.context_type = ‘global coordinate space’}
geometric_representation_context
{geometric_representation_context.coordinate_space_dimension = 3}}

/REP_TO_VAL_REP_ITEM(‘after perpendicular offset‘, length_measure)/

5.1.15.3.2 description

AIM element: product_definition.description
Source: ISO 10303-41

5.1.15.3.3 orientation

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.153, 5.2.4.183
Reference path: /PROD_DEF_TO_REP(‘global axis placement’)/
representation
{{representation.name = ‘global axis representation‘}
representation.context_of_items ->
representation_context =>
{representation_context.context_type = ‘global coordinate space’}}
/REP_ITEM(‘orientation’)/
descriptive_representation_item
descriptive_representation_item.description
{(descriptive_representation_item.description = ‘forward poining‘)
(descriptive_representation_item.description = ‘aft pointing‘)}

5.1.15.3.4 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition)/

5.1.15.3.5 global_axis_placement to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS(‘local units’)/
unit
unit = derived_unit
derived_unit

5.1.15.3.6 global_axis_placement to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.15.3.7 global_axis_placement to named_unit (as local_units)

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS(‘local units’)/
unit
unit = named_unit

named_unit

5.1.15.3.8 global_axis_placement to ship (as defined_for)

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product, 'ship')/}

5.1.15.4 LOCAL_CO_ORDINATE_SYSTEM

AIM element: [property_definition]
[axis2_placement_3d]
Source: ISO 10303-41
ISO 10303-42
Rules: 5.2.4.87, 5.2.4.162, 5.2.4.183
Reference path: {[/CLASS(property_definition, 'local co ordinate system', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}
/PROP_DEF_REP_HELP('local coordinate system')/
representation
{{representation.name = 'local axis representation'}
representation.context_of_items ->
representation_context =>
{representation_context.context_type = 'local coordinate space'}
geometric_representation_context
{geometric_representation_context.coordinate_space_dimension = 3}}
/REP_ITEM('local axes and origin')/
geometric_representation_item =>
placement =>
axis2_placement_3d

5.1.15.4.1 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.15.4.2 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.15.4.3 local_co_ordinate_system to definable_object (as defined_for)

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/

5.1.15.4.4 local_co_ordinate_system to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit

derived_unit

5.1.15.4.5 local_co_ordinate_system to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.15.4.6 local_co_ordinate_system to global_axis_placement (as parent)

AIM element: PATH
Rules: 5.2.4.184
Reference path: axis2_placement_3d <=
placement <=
geometric_representation_item <=
representation_item <-
representation_map.mapping_origin
representation_map
{representation_map.mapped_representation ->
representation
representation.name = 'local axis representation'}
representation_map <-
mapped_item.mapping_source
mapped_item
{mapped_item.name = 'local coordinate system position in global coordinate system'}
mapped_item.mapping_target ->
representation_item =>
geometric_representation_item =>
placement =>
axis2_placement_3d

5.1.15.4.7 local_co_ordinate_system to local_co_ordinate_system (as parent)

AIM element: PATH
Rules: 5.2.4.184
Reference path: axis2_placement_3d <=
placement <=
geometric_representation_item <=
representation_item <-
representation_map.mapping_origin
representation_map
{representation_map.mapped_representation ->
representation
representation.name = 'local axis representation'}
representation_map <-
mapped_item.mapping_source
mapped_item
{mapped_item.name = 'local coordinate system position in parent local coordinate system'}

mapped_item.mapping_target ->
representation_item =>
geometric_representation_item =>
placement =>
axis2_placement_3d

5.1.15.4.8 local_co_ordinate_system to local_co_ordinate_system_with_position_reference (as parent)

AIM element: PATH
Rules: 5.2.4.184
Reference path: axis2_placement_3d <=
placement <=
geometric_representation_item <=
representation_item <-
representation_map.mapping_origin
representation_map
{representation_map.mapped_representation ->
representation
representation.name = 'local axis representation'}
representation_map <-
mapped_item.mapping_source
mapped_item
{mapped_item.name = 'local coordinate system position in parent local coordinate system with position reference'}
mapped_item.mapping_target ->
representation_item =>
geometric_representation_item =>
placement =>
axis2_placement_3d

5.1.15.4.9 local_co_ordinate_system to named_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.15.5 LOCAL_CO_ORDINATE_SYSTEM_WITH_POSITION_REFERENCE

AIM element: [property_definition]
[axis2_placement_3d]
Source: ISO 10303-41
ISO 10303-42
Rules: 5.2.4.88, 5.2.4.214, 5.2.4.183
Reference path: {[/CLASS(property_definition, 'local co ordinate system with position reference', 'local co ordinate system')/]
[/CLASS(property_definition, 'local co ordinate system', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}
/PROP_DEF_REP_HELP('local coordinate system with position reference')/
representation

{{representation.name = ‘local axis with position reference representation‘}
representation.context_of_items ->
representation_context =>
{representation_context.context_type = ‘local coordinate space’}
geometric_representation_context
{geometric_representation_context.coordinate_space_dimension = 3}}
/REP_ITEM(’local axes and origin’)/
geometric_representation_item =>
placement =>
axis2_placement_3d

5.1.15.5.1 description

AIM element: /SUPERTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.1)

5.1.15.5.2 longitudinal_ref

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_REP_HELP(‘local coordinate system with position reference’)/
representation
{representation.name = ‘local axis with position reference representation‘}
representation.items [i] ->
representation_item
representation_item =>
{representation_item.name = ‘longitudinal ref’}
value_representation_item
{value_representation_item.value_component ->
measure_value = length_measure}

5.1.15.5.3 transversal_ref

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_REP_HELP(‘local coordinate system with position reference’)/
representation
{representation.name = ‘local axis with position reference representation‘}
representation.items [i] ->
representation_item
representation_item =>
{representation_item.name = ‘transversal ref’}
value_representation_item
{value_representation_item.value_component ->
measure_value = length_measure}

5.1.15.5.4 version_id

AIM element: /SUPERTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.2)

5.1.15.5.5 vertical_ref

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROP_DEF_REP_HELP(‘local coordinate system with position reference’)/
representation

{representation.name = ‘local axis with position reference representation‘}
representation.items [i] ->
representation_item
representation_item =>
{representation_item.name = ‘vertical ref’}
value_representation_item
{value_representation_item.value_component ->
measure_value = length_measure}

5.1.15.5.6 local_co_ordinate_system_with_position_reference to definable_object (as defined_for)

AIM element: /SUPERTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.3)

5.1.15.5.7 local_co_ordinate_system_with_position_reference to derived_unit (as local_units)

AIM element: /SUPERTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.4)

5.1.15.5.8 local_co_ordinate_system_with_position_reference to global_id (as id)

AIM element: /SUPERTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.5)

5.1.15.5.9 local_co_ordinate_system_with_position_reference to global_axis_placement (as parent)

AIM element: /SUPERTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.6)

5.1.15.5.10 local_co_ordinate_system to local_co_ordinate_system (as parent)

AIM element: /SUPERTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.7)

5.1.15.5.11 local_co_ordinate_system to local_co_ordinate_system_with_position_reference (as parent)

AIM element: /SUPERTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.8)

5.1.15.5.12 local_co_ordinate_system_with_position_reference to named_unit (as local_units)

AIM element: /SUPERTYPE(local_co_ordinate_system)/ -- (see 5.1.15.4.9)

5.1.15.5.13 local_co_ordinate_system_with_position_reference to longitudinal_position (as longitudinal_ref)

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: /PROP_DEF_REP_HELP(‘local coordinate system with position reference’)/
representation
{representation.name = ‘local axis with position reference representation‘}
representation.items [i] ->
representation_item
(representation_item =>
compound_representation_item
{/CLASS_HELP(compound_representation_item)/
(group.name = ‘longitudinal position’)

(group.name = ‘spacing position with offset’)
class}}

5.1.15.5.14 local_co_ordinate_system_with_position_reference to transversal_position (as transversal_ref)

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: /PROP_DEF_REP_HELP(‘local coordinate system with position reference’)/
representation
{representation.name = ‘local axis with position reference representation‘}
representation.items [i] ->
representation_item
representation_item =>
compound_representation_item
{/CLASS_HELP(compound_representation_item)/
(group.name = ‘transversal position’)
(group.name = ‘spacing position with offset’)
class}}

5.1.15.5.15 local_co_ordinate_system_with_position_reference to vertical_position (as vertical_ref)

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: /PROP_DEF_REP_HELP(‘local coordinate system with position reference’)/
representation
{representation.name = ‘local axis with position reference representation‘}
representation.items [i] ->
representation_item
representation_item =>
compound_representation_item
{/CLASS_HELP(compound_representation_item)/
(group.name = ‘vertical position’)
(group.name = ‘spacing position with offset’)
class}}

5.1.15.6 LONGITUDINAL_POSITION

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: {[/CLASS(compound_representation_item, ‘longitudinal position’, ‘spacing position’)/]
[/ROOT_CLASS(compound_representation_item, ‘spacing position’)/]}
{compound_representation_item <-
representation.items[i]
representation
representation.context_of_items
representation_context =>
{representation_context.context_type = ‘global coordinate space’}
geometric_representation_context}

5.1.15.6.1 name

AIM element: /SUPERTYPE(spacing_position)/ -- (see 5.1.15.8.1)

5.1.15.6.2 position

AIM element: /SUPERTYPE(spacing_position)/ -- (see 5.1.15.8.2)

5.1.15.6.3 position_number

AIM element: /SUPERTYPE(spacing_position)/ -- (see 5.1.15.8.3)

5.1.15.7 LONGITUDINAL_TABLE

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, 'longitudinal table', 'spacing table')/]
[/CLASS(property_definition, 'spacing table', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.15.7.1 description

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.1)

5.1.15.7.2 name

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.2)

5.1.15.7.3 version_id

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.3)

5.1.15.7.4 longitudinal_table to definable_object (as defined_for)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.4)

5.1.15.7.5 longitudinal_table to derived_unit (as local_units)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.5)

5.1.15.7.6 longitudinal_table to global_id (as id)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.6)

5.1.15.7.7 longitudinal_table to longitudinal_position (as spacing_table_representations)

AIM element: PATH
Reference path: /PROP_DEF_TO_REP/
representation.items [i] ->
representation_item =>
compound_representation_item
{/CLASS_ID(compound_representation_item, 'longitudinal position')/}

5.1.15.7.8 longitudinal_table to named_unit (as local_units)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.7)

5.1.15.8 SPACING_POSITION

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: {/ROOT_CLASS(compound_representation_item, ‘spacing position’)/}
{compound_representation_item <-
representation.items[i]
representation
representation.context_of_items
representation_context =>
{representation_context.context_type = ‘global coordinate space’}
geometric_representation_context}

5.1.15.8.1 name

AIM element: compound_representation_item.name
Source: ISO 10303-43

5.1.15.8.2 position

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.220, 5.2.4.183
Reference path: /COMPOUND(‘position‘)/
value_representation_item
{value_representation_item.value_component ->
measure_value = length_measure}

5.1.15.8.3 position_number

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.220, 5.2.4.183
Reference path: /COMPOUND(‘position number‘)/
value_representation_item
{value_representation_item.value_component ->
measure_value = count_measure}

5.1.15.9 SPACING_POSITION_WITH_OFFSET

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: {[/CLASS(compound_representation_item, ‘spacing position with offset’, ‘spacing position’)/]
[/ROOT_CLASS(compound_representation_item, ‘spacing position’)/]}
{compound_representation_item <-
representation.items[i]
representation
representation.context_of_items
representation_context =>
{representation_context.context_type = ‘global coordinate space’}
geometric_representation_context}

5.1.15.9.1 name

AIM element: /SUPERTYPE(spacing_position)/ -- (see 5.1.15.8.1)

5.1.15.9.2 offset

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.222, 5.2.4.183
Reference path: /COMPOUND('offset')/
value_representation_item
{value_representation_item.value_component ->
measure_value = length_measure}

5.1.15.9.3 position

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /COMPOUND('position')/
value_representation_item
{value_representation_item.value_component ->
measure_value = length_measure}

5.1.15.9.4 position_number

AIM element: /SUPERTYPE(spacing_position)/ -- (see 5.1.15.8.3)

5.1.15.9.5 spacing_position_with_offset to spacing_position (as relating_spacing_position)

AIM element: PATH
Rules: 5.2.4.221
Reference path: /COMPOUND('relating spacing position')/
{/CLASS_HELP(compound_representation_item)/
(group.name = 'longitudinal position')
(group.name = 'transversal position')
(group.name = 'vertical position')
class}

5.1.15.10 SPACING_TABLE

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, 'spacing table', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.15.10.1 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.15.10.2 name

AIM element: property_definition.name
Source: ISO 10303-41

5.1.15.10.3 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.15.10.4 spacing_table to definable_object (as defined_for)

AIM element: PATH
Reference path: property_definition
property_definition.definition ->
characterized_definition
characterized_definition = characterized_product_definition
(characterized_product_definition = product_definition
product_definition
{/CLASS_ID(product_definition, 'compartment')})
(characterized_product_definition = product_definition
product_definition
product_definition.formation ->
product_definition_formation
product_definition_formation.of_product ->
product
{/CLASS_ID(product, 'ship')})

5.1.15.10.5 spacing_table to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.15.10.6 spacing_table to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.15.10.7 spacing_table to named_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.15.10.8 spacing_table to spacing_position (as spacing_table_representations)

AIM element: PATH
Reference path: /PROP_DEF_TO_REP/

representation.items [i] ->
representation_item =>
compound_representation_item
{/CLASS_HELP(compound_representation_item)/
(group.name = ‘longitudinal position’)
(group.name = ‘transversal position’)
(group.name = ‘vertical position’)
(group.name = ‘spacing position with offset’)
class}

5.1.15.11 STATION_TABLE

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, ‘station table’, ‘longitudinal table’)/]
[/CLASS(property_definition, ‘longitudinal table’, ‘spacing table’)/]
[/CLASS(property_definition, ‘spacing table’, ‘definition’)/]
[/CLASS(property_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(property_definition, ‘versionable object’)/]}

5.1.15.11.1 description

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.1)

5.1.15.11.2 name

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.2)

5.1.15.11.3 version_id

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.3)

5.1.15.11.4 station_table to definable_object (as defined_for)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.4)

5.1.15.11.5 station_table to derived_unit (as local_units)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.5)

5.1.15.11.6 station_table to global_id (as id)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.6)

5.1.15.11.7 station_table to longitudinal_position (as spacing_table_representations)

AIM element: /SUPERTYPE(longitudinal_table)/ -- (see 5.1.15.7.7)

5.1.15.11.8 station_table to named_unit (as local_units)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.7)

5.1.15.12 TRANSVERSAL_POSITION

AIM element: compound_representation_item

Source: ISO 10303-43
Reference path: {[/CLASS(compound_representation_item, ‘transversal position’, ‘spacing position’)/]
[/ROOT_CLASS(compound_representation_item, ‘spacing position’)/]}
{compound_representation_item <-
representation.items[i]
representation
representation.context_of_items
representation_context =>
{representation_context.context_type = ‘global coordinate space’}
geometric_representation_context}

5.1.15.12.1 name

AIM element: /SUPERTYPE(spacing_position)/ -- (see 5.1.15.8.1)

5.1.15.12.2 position

AIM element: /SUPERTYPE(spacing_position)/ -- (see 5.1.15.8.2)

5.1.15.12.3 position_number

AIM element: /SUPERTYPE(spacing_position)/ -- (see 5.1.15.8.3)

5.1.15.13 TRANSVERSAL_TABLE

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, ‘transversal table’, ‘spacing table’)/]
[/CLASS(property_definition, ‘spacing table’, ‘definition’)/]
[/CLASS(property_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(property_definition, ‘versionable object’)/]}

5.1.15.13.1 description

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.1)

5.1.15.13.2 name

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.2)

5.1.15.13.3 version_id

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.3)

5.1.15.13.4 transversal_table to definable_object (as defined_for)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.4)

5.1.15.13.5 transversal_table to derived_unit (as local_units)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.5)

5.1.15.13.6 transversal_table to global_id (as id)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.6)

5.1.15.13.7 transversal_table to named_unit (as local_units)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.7)

5.1.15.13.8 transversal_table to transversal_position (as spacing_table_representations)

AIM element: PATH
Reference path: /PROP_DEF_TO_REP/
representation.items [i] ->
representation_item =>
compound_representation_item
{/CLASS_ID(compound_representation_item, 'transversal position')/}

5.1.15.14 VERTICAL_POSITION

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: {[/CLASS(compound_representation_item, 'vertical position', 'spacing position')/]
[/ROOT_CLASS(compound_representation_item, 'spacing position')/]}
{compound_representation_item <-
representation.items[i]
representation
representation.context_of_items
representation_context =>
{representation_context.context_type = 'global coordinate space'}
geometric_representation_context}

5.1.15.14.1 name

AIM element: /SUPERTYPE(spacing_position)/ -- (see 5.1.15.8.1)

5.1.15.14.2 position

AIM element: /SUPERTYPE(spacing_position)/ -- (see 5.1.15.8.2)

5.1.15.14.3 position_number

AIM element: /SUPERTYPE(spacing_position)/ -- (see 5.1.15.8.3)

5.1.15.15 VERTICAL_TABLE

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, 'vertical table', 'spacing table')/]
[/CLASS(property_definition, 'spacing table', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.15.15.1 description

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.1)

5.1.15.15.2 name

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.2)

5.1.15.15.3 version_id

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.3)

5.1.15.15.4 vertical_table to definable_object (as defined_for)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.4)

5.1.15.15.5 vertical_table to derived_unit (as local_units)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.5)

5.1.15.15.6 vertical_table to global_id (as id)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.6)

5.1.15.15.7 vertical_table to named_unit (as local_units)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.7)

5.1.15.15.8 vertical_table to vertical_position (as spacing_table_representations)

AIM element: PATH
Reference path: /PROP_DEF_TO_REP/
representation.items [i] ->
representation_item =>
compound_representation_item
{/CLASS_ID(compound_representation_item, 'vertical position')/}

5.1.15.16 WATERLINE_TABLE

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, 'waterline table', 'vertical table')/]
[/CLASS(property_definition, 'vertical table', 'spacing table')/]
[/CLASS(property_definition, 'spacing table', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.15.16.1 description

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.1)

5.1.15.16.2 name

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.2)

5.1.15.16.3 version_id

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.3)

5.1.15.16.4 waterline_table to definable_object (as defined_for)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.4)

5.1.15.16.5 waterline_table to derived_unit (as local_units)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.5)

5.1.15.16.6 waterline_table to global_id (as id)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.6)

5.1.15.16.7 waterline_table to named_unit (as local_units)

AIM element: /SUPERTYPE(spacing_table)/ -- (see 5.1.15.10.7)

5.1.15.16.8 waterline_table to vertical_position (as spacing_table_representations)

AIM element: /SUPERTYPE(vertical_table)/ -- (see 5.1.15.15.8)

5.1.16 Product_structures UoF

5.1.16.1 SPACE_PRODUCT_STRUCTURE

AIM element: [product_definition]
[group]
Source: ISO 10303-41
Reference path: {[/CLASS(product_definition, 'space product structure', 'item')/]
[/CLASS(product_definition, 'item', 'definable object')/]
[/ROOT_CLASS(product_definition, 'definable object')/]
[/CLASS(product_definition, 'space product structure', 'item structure')/]
[/CLASS(product_definition, 'item structure', 'definable object')/]
[/CLASS(product_definition, 'item structure', 'versionable object')/]
[/ROOT_CLASS(product_definition, 'versionable object')/]
[product_definition
/LINK_TO_GROUP(product_definition)/]}

5.1.16.1.1 description

AIM element: product_definition.description
Source: ISO 10303-41

5.1.16.1.2 name

AIM element: product_definition.name
Source: ISO 10303-41
Reference path: /NAME_ASSGN(product_definition)/

5.1.16.1.3 product_structure_type

AIM element: group.name
Source: ISO 10303-41
Reference path: product_definition = classification_item
classification_item <-
applied_classification_assignment.items[i]
applied_classification_assignment
applied_classification_assignment <=
classification_assignment
{classification_assignment.role ->

classification_role
classification_role.name = ‘product structure type‘}
classification_assignment.assigned_class ->
group
group.name
{(group.name = ‘compartments in arrangement’)
(group.name = ‘items in compartment’)}

5.1.16.1.4 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition)/

5.1.16.1.5 space_product_structure to external_reference (as documentation)

#1: If as documentation refers to an External_reference

AIM element: PATH
Reference path: product_definition
product_definition = external_identification_item
external_identification_item <-
applied_external_identification_assignment.items[i]
applied_external_identification_assignment

#2: If as documentation refers to a Document_reference_with_address

AIM element: PATH
Reference path: product_definition
/DOC_REF(product_definition,’documentation’)/
document
{/CLASS_ID(document, ‘document reference with address’)/}

5.1.16.1.6 space_product_structure to external_instance_reference (as external_items)

AIM element: PATH
Reference path: group <-
/GROUPS(product_definition, ’item structure’)/
{([/CLASS_ID(product_definition, 'structural part')/]
[/EXT_INST_REF(product_definition, 'ship structures schema', 'structural part')/])
([/CLASS_ID(product_definition, 'plant item')/]
[/EXT_INST_REF(product_definition, 'plant spatial configuration', 'plant item')/])
([/CLASS_ID(product_definition, 'compartment')/]
[/EXT_INST_REF(product_definition, 'ship arrangement schema',
'compartment')/])}

5.1.16.1.7 space_product_structure to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition
identification_item <-
applied_identification_assignment.items[i]

applied_identification_assignment

5.1.16.1.8 space_product_structure to item (as items)

AIM element: PATH
Reference path: group <-
/GROUPS(product_definition, ’item structure’)/
{/CLASS_ID(product_definition, 'compartment’)/}

5.1.16.1.9 space_product_structure to ship (as ship_context)

AIM element: PATH
Reference path: product_definition
product_definition.formation ->
product_definition_formation
{product_definition_formation.id = 'ship arrangement'}
product_definition_formation.of_product ->
product
{/CLASS_ID(product, ’ship’)/}

5.1.16.1.10 space_product_structure to space (as contained_in)

AIM element: PATH
Reference path: product_definition
{/CLASS_ID(product_definition, 'space product structure')/}
product_definition <-
product_definition_relationship.relating_definition
product_definition_relationship
product_definition_relationship.related_definition ->
product_definition
({/CLASS_ID(product_definition, 'compartment')/})
({/CLASS_ID(product_definition, 'deck zone')/})

5.1.16.2 SPACE_PRODUCT_STRUCTURE_REVISION

AIM element: group
Source: ISO 10303-41
Reference path: {[CLASS(group, 'space product structure revision', 'revision with context')]
[CLASS(group, 'revision with context', 'revision')]
[CLASS(group, ‘revision’, ‘versionable object’)]
[ROOT_CLASS(group, ‘versionable object’)]}

5.1.16.2.1 description

AIM element: group.description
Source: ISO 10303-41
Rules: 5.2.4.216

5.1.16.2.2 name

AIM element: group.name
Source: ISO 10303-41

5.1.16.2.3 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(group)/

5.1.16.2.4 space_product_structure_revision to space_product_structure (as context_of_revision)

AIM element: PATH
Rules: 5.2.4.217
Reference path: group <-
/(RELATE_GROUP_2_DO(product_definition, ‘space product structure’)/

5.1.16.2.5 space_product_structure_revision to design_definition (as members)

AIM element: PATH
Rules: 5.2.4.50
Reference path: group <-
(/RELATE_GROUP_2_VO(product_definition_shape,
'compartment design definition’)/)
(/RELATE_GROUP_2_VO(applied_external_identification_assignment,
'external instance reference')/)

5.1.17 Shapes UoF

5.1.17.1 NON_MANIFOLD_SURFACE_SHAPE

AIM element: non_manifold_surface_shape_representation
Source: ISO 10303-508
Rules: 5.2.4.182, 5.2.4.218

5.1.18 Ship_general_characteristics UoF

5.1.18.1 CARRIER

AIM element: product_related_product_category
Source: ISO 10303-41
Reference path: {[/PROD_CAT_NAME(‘carrier‘)/]
[/CLASS(product_related_product_category,’carrier’, ’shiptype’)/]
[/CLASS(product_related_product_category, ‘shiptype’, ‘functional definition’)/]
[/CLASS(product_related_product_category, ‘functional definition’, ‘definition’)/]
[/CLASS(product_related_product_category, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(product_related_product_category, ‘versionable object’)/]}

5.1.18.1.1 description

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.1)

5.1.18.1.2 has_type

AIM element: product_category.name
Source: ISO 10303-41
Reference path: /PROD_CAT_NAME(‘carrier‘)/

product_category <-
product_category_relationship.category
product_category_relationship
{product_category_relationship.name = ‘carrier types‘}
product_category_relationship.sub_category ->
product_category
{(product_category.name = ‘Container carrier‘)
(product_category.name = ‘Bulk carrier’)
(product_category.name = ‘Ore carrier’)
(product_category.name = ‘Oil tanker’)
(product_category.name = ‘Roro vessel’)
(product_category.name = ‘Ferry’)
(product_category.name = ‘Car ferry’)
(product_category.name = ‘Cruise liner’)
(product_category.name = ‘Passenger vessel’)
(product_category.name = ‘Cargo ship carrying passengers’)
(product_category.name = ‘Product tanker’)
(product_category.name = ‘Gas carrier’)
(product_category.name = ‘Liquefied gas tanker’)
(product_category.name = ‘LNG carrier’)
(product_category.name = ‘LPG carrier’)
(product_category.name = ‘Chemical tanker’)
(product_category.name = ‘Chemical tanker type 1’)
(product_category.name = ‘Tanker for refrigerated fruit juice’)
(product_category.name = ‘General cargo carrier’)
(product_category.name = ‘Dry cargo vessel’)
(product_category.name = ‘Refrigerated cargo carrying ship’)
(product_category.name = ‘High speed craft passenger’)
(product_category.name = ‘High speed craft cargo’)
(product_category.name = ‘Hydrofoil’)
(product_category.name = ‘Car carrier’)
(product_category.name = ‘Barge’)
(product_category.name = ‘Barge for deck loading’)
(product_category.name = ‘Barge for oil’)
(product_category.name = ‘Barge for liquefied gas’)
(product_category.name = ‘Barge pontoon’)
(product_category.name = ‘user defined’)}

5.1.18.1.3 user_def_function

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.2)

5.1.18.1.4 version_id

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.3)

5.1.18.1.5 carrier to global_id (as id)

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.4)

5.1.18.1.6 carrier to ship (as defined_for)

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.5)

5.1.18.2 CLASS_AND_STATUTORY_DESIGNATION

AIM element: product_definition
Source: ISO 10303-41
Rules: 5.2.4.30
Reference path: {[/CLASS(product_definition, ‘class and statutory designation’, ‘general characteristics definition’)/]
[/CLASS(product_definition, ‘general characteristics definition’, ‘definition’)/]
[/CLASS(product_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(product_definition, ‘versionable object’)/]}

5.1.18.2.1 class_number

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.107
Reference path: /PROD_DEF_TO_DESC_REP_ITEM(‘class and statutory designation’, ‘class number‘)/

5.1.18.2.2 description

AIM element: product_definition.description
Source: ISO 10303-41

5.1.18.2.3 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition)/

5.1.18.2.4 class_and_statutory_designation to class_notation (as the_class)

AIM element: PATH
Rules: 5.2.4.55
Reference path: /PROD_DEF_PROP_DEF/
{/CLASS_ID(property_definition, ‘class notation’)/}

5.1.18.2.5 class_and_statutory_designation to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.18.2.6 class_and_statutory_designation to regulation (as the_statutory)

AIM element: PATH
Rules: 5.2.4.63
Reference path: /PROD_DEF_PROP_DEF/
{/CLASS_ID(property_definition, ‘regulation’)/}

5.1.18.2.7 class_and_statutory_designation to ship (as defined_for)

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product, 'ship')}

5.1.18.3 CLASS_NOTATION

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.77
Reference path: /ROOT_CLASS(property_definition, 'class notation')/

5.1.18.3.1 approval_required_for_heavy_cargo

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.188
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('class notation', 'approval required for heavy cargo')/
{(descriptive_representation_item.description = 'HC')
(descriptive_representation_item.description = 'HC_E')
(descriptive_representation_item.description = 'HC_EA')}

5.1.18.3.2 approval_required_for_oil_cargo

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.110
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('class notation', 'approval required for oil cargo')/
{(descriptive_representation_item.description = 'TRUE')
(descriptive_representation_item.description = 'FALSE')}

5.1.18.3.3 approval_required_loading_unloading_aground

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.110
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('class notation', 'approval required for loading unloading aground')/
{(descriptive_representation_item.description = 'TRUE')
(descriptive_representation_item.description = 'FALSE')}

5.1.18.3.4 approval_required_loading_unloading_grabs

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.110
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('class notation', 'approval required for unloading grabs')/
{(descriptive_representation_item.description = 'TRUE')
(descriptive_representation_item.description = 'FALSE')}

5.1.18.3.5 class_notations_hull

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.31
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('class notation', 'class notations hull')/

5.1.18.3.6 class_notations_machinery

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.31
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('class notation', 'class notations machinery')/

5.1.18.3.7 class_society

AIM element: organization_assignment.assigned_organization
Source: ISO 10303-41
Rules: 5.2.4.78
Reference path: /ORG_ASSGN(property_definition, 'class society')/

5.1.18.3.8 ice_class_notation

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.188
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('class notation', 'ice class notation')/

5.1.18.3.9 service_area

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.110
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('class notation', 'service area')/

5.1.18.3.10 service_factor

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.188
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('class notation', 'service factor', count_measure)/

5.1.18.4 CLASS_PARAMETERS

AIM element: product_definition
Source: ISO 10303-41
Rules: 5.2.4.32
Reference path: {[/CLASS(product_definition, 'class parameters', 'general characteristics definition')/]
[/CLASS(product_definition, 'general characteristics definition', 'definition')/]
[/CLASS(product_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(product_definition, 'versionable object')/]}

5.1.18.4.1 block_coefficient_class

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.111
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘class parameters’, ‘block coefficient class‘, ratio_measure)/

5.1.18.4.2 description

AIM element: product_definition.description
Source: ISO 10303-41

5.1.18.4.3 design_speed_ahead

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.111
Reference path: /PROD_DEF_TO_SPECIAL_VAL_REP_ITEM(‘class parameters’, ‘design speed ahead‘, 'speed unit')/

5.1.18.4.4 design_speed_astern

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.111
Reference path: /PROD_DEF_TO_SPECIAL_VAL_REP_ITEM(‘class parameters’, ‘design speed astern‘, 'speed unit')/

5.1.18.4.5 length_class

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.111
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘class parameters’, ‘length class‘, positive_length_measure)/

5.1.18.4.6 length_solas

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.111
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘class parameters’, ‘length solas‘, positive_length_measure)/

5.1.18.4.7 scantlings_draught

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.111
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘class parameters’, ‘scantlings draught‘, positive_length_measure)/

5.1.18.4.8 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition)/

5.1.18.4.9 class_parameters to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.18.4.10 class_parameters to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.18.4.11 class_parameters to named_unit (as local_units)

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.18.4.12 class_parameters to ship (as defined_for)

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product, 'ship')/}

5.1.18.5 FREEBOARD_CHARACTERISTICS

AIM element: product_definition
Source: ISO 10303-41
Rules: 5.2.4.43
Reference path: {[/CLASS(product_definition, 'freeboard characteristics', 'general characteristics definition')/]
[/CLASS(product_definition, 'general characteristics definition', 'definition')/]
[/CLASS(product_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(product_definition, 'versionable object')/]}

5.1.18.5.1 assigned_code

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.150

Reference path: /PROD_DEF_TO_DESC_REP_ITEM(‘freeboard characteristics’, ‘assigned code‘)/

5.1.18.5.2 date_freeboard_assigned

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-215
Rules: 5.2.4.69
Reference path: /DAT_TIME_ASSGN(product_definition, ‘date freeboard assigned’)/

5.1.18.5.3 description

AIM element: product_definition.description
Source: ISO 10303-41

5.1.18.5.4 freeboard

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.150
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘freeboard characteristics’, ‘freeboard‘, positive_length_measure)/

5.1.18.5.5 freeboard_assigned_by

AIM element: organization_assignment.assigned_organization
Source: ISO 10303-41
Rules: 5.2.4.70
Reference path: /ORG_ASSGN(product_definition, ‘freeboard assigned by‘)/

5.1.18.5.6 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition)/

5.1.18.5.7 freeboard_characteristics to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS(‘local units’)/
unit
unit = derived_unit
derived_unit

5.1.18.5.8 freeboard_characteristics to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.18.5.9 freeboard_characteristics to loadline (as applicable_loadline)

AIM element: PATH
Rules: 5.2.4.58
Reference path: /PROD_DEF_PROP_DEF/
{/CLASS_ID(property_definition, ‘loadline’)/}

5.1.18.5.10 freeboard_characteristics to named_unit (as local_units)

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS(‘local units’)/
unit
unit = named_unit
named_unit

5.1.18.5.11 freeboard_characteristics to ship (as defined_for)

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product, ‘ship’)/}

5.1.18.6 LIGHTSHIP_DEFINITION

AIM element: product_definition
Source: ISO 10303-41
Rules: 5.2.4.48
Reference path: {[/CLASS(product_definition, ‘lightship definition’, ‘definition’)/]
[/CLASS(product_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(product_definition, ‘versionable object’)/]}

5.1.18.6.1 description

AIM element: product_definition.description
Source: ISO 10303-41

5.1.18.6.2 lightship_weight

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.155
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘lightship definition’, ‘lightship weight‘, mass_measure)/

5.1.18.6.3 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition)/

5.1.18.6.4 lightship_definition to centre_location (as lightship_centre_of_gravity)

AIM element: PATH
Rules: 5.2.4.183, 5.2.4.155
Reference path: /PROD_DEF_TO_REP(‘lightship definition parameters’)/

/REP_ITEM(‘lightship centre of gravity‘)/
compound_representation_item
{/CLASS_ID(compound_representation_item, 'centre location')}

5.1.18.6.5 lightship_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS(‘local units’)/
unit
unit = derived_unit
derived_unit

5.1.18.6.6 lightship_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.18.6.7 lightship_definition to lightship_weight_item (as lightship_items)

AIM element: PATH
Reference path: /PROD_DEF_PROP_DEF/
{/CLASS_ID(property_definition, ‘lightship_weight_item’)/}

5.1.18.6.8 lightship_definition to named_unit (as local_units)

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS(‘local units’)/
unit
unit = named_unit
named_unit

5.1.18.6.9 lightship_definition to ship (as defined_for)

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product, ‘ship’)/}

5.1.18.7 LIGHTSHIP_WEIGHT_ITEM

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.98
Reference path: {/CLASS_ID(property_definition, ‘lightship weight item’)/}

5.1.18.7.1 aft_weight_extent

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.156

Reference path: /PROP_DEF_TO_VAL_REP_ITEM(‘lightship weight item’, ‘aft weight extent‘, length_measure)/

5.1.18.7.2 fwd_weight_extent

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.156
Reference path: /PROP_DEF_TO_VAL_REP_ITEM(‘lightship weight item’, ‘fwd weight extent‘, length_measure)/

5.1.18.7.3 lightship_item_description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.18.7.4 mass

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.215
Reference path: /PROP_DEF_TO_VAL_REP_ITEM(‘weight and centre of gravity’, ‘mass‘, mass_measure)/

5.1.18.7.5 lightship_weight_item to centre_location (as centre_of_gravity)

AIM element: PATH
Rules: 5.2.4.183, 5.2.4.215
Reference path: /PROP_DEF_TO_REP/
/REP_ITEM(‘centre of gravity‘)/
compound_representation_item
{/CLASS_ID(compound_representation_item, 'centre location')}

5.1.18.7.6 lightship_weight_item to moment_3d (as moment)

AIM element: PATH
Reference path: /PROP_DEF_TO_REP/
{representation
representation.name= ‘moment 3d‘}

5.1.18.8 LOADLINE

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.93
Reference path: {/CLASS_ID(property_definition, ‘loadline’)/}

5.1.18.8.1 load_line_block_coefficient

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.161
Reference path: /PROP_DEF_TO_VAL_REP_ITEM(‘loadline’, ‘load line block coefficient‘, ratio_measure)/

5.1.18.8.2 load_line_depth

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.161
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('loadline', 'load line depth', positive_length_measure)/

5.1.18.8.3 load_line_displacement

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.161
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('loadline', 'load line displacement', volume_measure)/

5.1.18.8.4 load_line_length

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.161
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('loadline', 'load line length', positive_length_measure)/

5.1.18.8.5 load_line_regulation

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.161
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('loadline', 'load line regulation')/
{(descriptive_representation_item.description = 'ILLC_1930')
(descriptive_representation_item.description = 'ILLC_1966')
(descriptive_representation_item.description = 'other')}

5.1.18.9 NAVY_SHIP

AIM element: product_related_product_category
Source: ISO 10303-41
Reference path: {[/PROD_CAT_NAME('navy ship')/]
[/CLASS(product_related_product_category,'navy ship','shiptype')/]
[/CLASS(product_related_product_category, 'shiptype', 'functional definition')/]
[/CLASS(product_related_product_category, 'functional definition', 'definition')/]
[/CLASS(product_related_product_category, 'definition', 'versionable object')/]
[/ROOT_CLASS(product_related_product_category, 'versionable object')/]}

5.1.18.9.1 description

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.1)

5.1.18.9.2 has_type

AIM element: product_category.name
Source: ISO 10303-41
Reference path: /PROD_CAT_NAME('navy ship')/

product_category <-
product_category_relationship.category
product_category_relationship
{product_category_relationship.name = 'navy ship types‘}
product_category_relationship.category ->
product_category
product_category_relationship
product_category_relationship.sub_category ->
product_category
{(product_category.name = ‘Aircraft carrier’)
(product_category.name = ‘Corvette’)
(product_category.name = ‘Cruiser’)
(product_category.name = ‘Destroyer’)
(product_category.name = ‘Fleet auxiliary vessel’)
(product_category.name = ‘Frigate’)
(product_category.name = ‘Mine warfare ship’)
(product_category.name = ‘Patrol force vessel’)
(product_category.name = ‘Service craft’)
(product_category.name = ‘Submarine’)
(product_category.name = ‘Auxiliary oiler’)
(product_category.name = ‘Landing platform dock’)
(product_category.name = ‘Landing platform helicopter’)
(product_category.name = ‘user defined’)}

5.1.18.9.3 user_def_function

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.2)

5.1.18.9.4 version_id

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.3)

5.1.18.9.5 navy_ship to global_id (as id)

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.4)

5.1.18.9.6 navy_ship to ship (as defined_for)

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.5)

5.1.18.10 OWNER_DESIGNATION

AIM element: product_definition
Source: ISO 10303-41
Reference path: {[/CLASS(product_definition, ‘owner designation’, ‘general characteristics definition’)/]
[/CLASS(product_definition, ‘general characteristics definition’, ‘definition’)/]
[/CLASS(product_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(product_definition, ‘versionable object’)/]}

5.1.18.10.1 description

AIM element: product_definition.description
Source: ISO 10303-41

5.1.18.10.2 managing_company

AIM element: organization_assignment.assigned_organization
Source: ISO 10303-41
Rules: 5.2.4.59
Reference path: /ORG_ASSGN(product_definition, ‘managing company‘)/

5.1.18.10.3 ordering_company

AIM element: organization_assignment.assigned_organization
Source: ISO 10303-41
Rules: 5.2.4.60
Reference path: /ORG_ASSGN(product_definition, ‘ordering company‘)/

5.1.18.10.4 owner_approval

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.206
Reference path: /PROD_DEF_TO_DESC_REP_ITEM('owner designation', 'owner approval')/

5.1.18.10.5 owning_company

AIM element: organization_assignment.assigned_organization
Source: ISO 10303-41
Rules: 5.2.4.61
Reference path: /ORG_ASSGN(product_definition, ‘owning company‘)/

5.1.18.10.6 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition)/

5.1.18.10.7 owner_designation to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.18.10.8 owner_designation to ship (as defined_for)

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product, ‘ship’)/}

5.1.18.11 PRINCIPAL_CHARACTERISTICS

AIM element: product_definition
Source: ISO 10303-41

Rules: 5.2.4.52
Reference path: {[/CLASS(product_definition, ‘principal characteristics’, ‘general characteristics definition’)/]
[/CLASS(product_definition, ‘general characteristics definition’, ‘definition’)/]
[/CLASS(product_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(product_definition, ‘versionable object’)/]}

5.1.18.11.1 block_coefficient

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.207
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘principal characteristics’, ‘block coefficient‘, ratio_measure)/

5.1.18.11.2 description

AIM element: product_definition.description
Source: ISO 10303-41

5.1.18.11.3 design_deadweight

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.168
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘principal characteristics’, ‘design deadweight‘, mass_measure)/

5.1.18.11.4 design_draught

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.168
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘principal characteristics’, ‘design draught‘, positive_length_measure)/

5.1.18.11.5 length_between_perpendiculars

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.168
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘principal characteristics’, ‘length between perpendiculars’, positive_length_measure)/

5.1.18.11.6 max_draught_at_ap

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.168
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘principal characteristics’, ‘max draught at ap’, positive_length_measure)/

5.1.18.11.7 max_draught_at_fp

AIM element: value_representation_item.value_component

Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.168
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘principal characteristics’, ‘max draught at fp’, positive_length_measure)/

5.1.18.11.8 min_draught_at_ap

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.168
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘principal characteristics’, ‘min draught at ap’, positive_length_measure)/

5.1.18.11.9 min_draught_at_fp

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.168
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘principal characteristics’, ‘min draught at fp’, positive_length_measure)/

5.1.18.11.10 moulded_breadth

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.168
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘principal characteristics’, ‘moulded breadth‘, positive_length_measure)/

5.1.18.11.11 moulded_depth

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.168
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘principal characteristics’, ‘moulded depth‘, positive_length_measure)/

5.1.18.11.12 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition)/

5.1.18.11.13 principal_characteristics to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS(‘local units’)/
unit
unit = derived_unit
derived_unit

5.1.18.11.14 principal_characteristics to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.18.11.15 principal_characteristics to named_unit (as local_units)

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.18.11.16 principal_characteristics to ship (as defined_for)

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product, 'ship')/}

5.1.18.12 REGULATION

AIM element: property_definition
Source: ISO 10303-41
Reference path: {/ROOT_CLASS(property_definition, 'regulation')/}

5.1.18.12.1 regulation to external_reference (as international_regulations)

#1: If as international_regulations is an External_reference

AIM element: PATH
Reference path: property_definition
external_identification_item = property_definition
external_identification_item <-
applied_external_identification_assignment.items[i]
applied_external_identification_assignment <=
external_identification_assignment <=
identification_assignment
identification_assignment.role ->
identificaton_role
{[identification_role.name = 'external reference']
[identification_role.description = 'international regulations']}

#2: If as international_regulations is a Document_reference_with_address

AIM element: PATH
Reference path: property_definition
/DOC_REF(property_definition,'international regulations')/
document
{/CLASS_ID(document, 'document reference with address')/}

5.1.18.12.2 regulation to external_reference (as national_regulations)

#1: If as national_regulations is an External_reference

AIM element: PATH
Reference path: property_definition
external_identification_item = property_definition
external_identification_item <-
applied_external_identification_assignment.items[i]
applied_external_identification_assignment <=
external_identification_assignment <=
identification_assignment
identification_assignment.role ->
identificaton_role
{[identification_role.name = 'external reference']
[identification_role.description = 'national regulations']}

#2: If as national_regulations is a Document_reference_with_address

AIM element: PATH
Reference path: property_definition
/DOC_REF(property_definition,'national regulations')/
document
{/CLASS_ID(document, 'document reference with address')/}

5.1.18.12.3 regulation to external_reference (as standards)

#1: If as standards is an External_reference

AIM element: PATH
Reference path: property_definition
external_identification_item = property_definition
external_identification_item <-
applied_external_identification_assignment.items[i]
applied_external_identification_assignment <=
external_identification_assignment <=
identification_assignment
identification_assignment.role ->
identificaton_role
{[identification_role.name = 'external reference']
[identification_role.description = 'standards']}

#2: If as standards is a Document_reference_with_address

AIM element: PATH
Reference path: property_definition
/DOC_REF(property_definition,'standards')/
document
{/CLASS_ID(document, 'document reference with address')/}

5.1.18.13 RESEARCH_SHIP

AIM element: product_related_product_category

Source: ISO 10303-41
Reference path: {[/PROD_CAT_NAME(‘research ship‘)/]
[/CLASS(product_related_product_category, ‘research ship’, ‘shiptype’)/]
[/CLASS(product_related_product_category, ‘shiptype’, ‘functional definition’)/]
[/CLASS(product_related_product_category, ‘functional definition’, ‘definition’)/]
[/CLASS(product_related_product_category, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(product_related_product_category, ‘versionable object’)/]}

5.1.18.13.1 description

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.1)

5.1.18.13.2 has_type

AIM element: product_category.name
Source: ISO 10303-41
Reference path: /PROD_CAT_NAME(‘research ship‘)/
product_category <-
product_category_relationship.category
product_category_relationship
{product_category_relationship.name = ‘research ship types‘}
product_category_relationship.sub_category ->
product_category
{product_category.name = ‘user defined’}

5.1.18.13.3 user_def_function

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.2)

5.1.18.13.4 version_id

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.3)

5.1.18.13.5 research_ship to global_id (as id)

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.4)

5.1.18.13.6 research_ship to ship (as defined_for)

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.5)

5.1.18.14 SHIP_DESIGNATION

AIM element: product_definition
Source: ISO 10303-41
Reference path: {[/CLASS(product_definition, ‘ship designation’, ‘general characteristics definition’)/]
[/CLASS(product_definition, ‘general characteristics definition’, ‘definition’)/]
[/CLASS(product_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(product_definition, ‘versionable object’)/]}

5.1.18.14.1 call_sign

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.53

Reference path: /HAS_ID_1_ROLE(product_definition, ‘call sign’)/

5.1.18.14.2 description

AIM element: product_definition.description
Source: ISO 10303-41

5.1.18.14.3 flag_state

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.57
Reference path: /HAS_ID_1_ROLE(product_definition, ‘flag state’)/

5.1.18.14.4 port_of_registration

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.62
Reference path: /HAS_ID_1_ROLE(product_definition, ‘port of registration’)/

5.1.18.14.5 ship_identification

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.219
Reference path: /HAS_ID_2_ROLES(product_definition, ‘IMO number’, ‘pennant hull number’)/

5.1.18.14.6 ship_name

AIM element: product_definition.name
Source: ISO 10303-41
Reference path: /NAME_ASSGN(product_definition)/

5.1.18.14.7 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition)/

5.1.18.14.8 ship_designation to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.18.14.9 ship_designation to ship (as defined_for)

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/

{/CLASS_ID(product, ‘ship’)/}

5.1.18.15 SHIPTYPE

#1: Shiptype is a carrier
#2: Shiptype is a navy_ship
#3: Shiptype is a research_ship
#4: Shiptype is a working_ship

AIM element: #1: /SUBTYPE(carrier)/ -- (see 5.1.18.1)
#2: /SUBTYPE(navy_ship)/ -- (see 5.1.18.9)
#3: /SUBTYPE(research_ship)/ -- (see 5.1.18.13)
#4: /SUBTYPE(working_ship)/ -- (see 5.1.18.17)

5.1.18.15.1 description

AIM element: product_category.description
Source: ISO 10303-41
Reference path: product_related_product_category <=
product_category
product_category.description

5.1.18.15.2 user_def_function

AIM element: PATH
Source: ISO 10303-41
Reference path: product_related_product_category <=
product_category
{product_category.name = ‘user defined’}
product_category<-
product_category_relationship.category
product_category_relationship
product_category_relationship.sub_category ->
product_category
product_category.name

5.1.18.15.3 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_related_product_category)/

5.1.18.15.4 shiptype to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.72
Reference path: product_related_product_category
identification_item = product_related_product_category
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.18.15.5 shiptype to ship (as defined_for)

AIM element: PATH
Reference path: product_related_product_category
product_related_product_category.products[i] ->
product
{/CLASS_ID(product, ‘ship’)/}

5.1.18.16 SHIPYARD_DESIGNATION

AIM element: product_definition
Source: ISO 10303-41
Reference path: {[/CLASS(product_definition, ‘shipyard designation’, ‘general characteristics definition’)/]
[/CLASS(product_definition, ‘general characteristics definition’, ‘definition’)/]
[/CLASS(product_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(product_definition, ‘versionable object’)/]}

5.1.18.16.1 description

AIM element: product_definition.description
Source: ISO 10303-41

5.1.18.16.2 role

AIM element: organization_assignment.role
Source: ISO 10303-41
Reference path: /ORG_ASSGN_PART(product_definition)/
{organization_assignment.role ->
organization_role
{(organization_role.name = ‘prime design’)
(organization_role.name = ‘prime build’)
(organization_role.name = ‘prime repair’)
(organization_role.name = ‘prime’)
(organization_role.name = ‘subcontractor’)}

5.1.18.16.3 shipyard

AIM element: organization_assignment.assigned_organization
Source: ISO 10303-41
Rules: 5.2.4.64
Reference path: /ORG_ASSGN(product_definition, ‘shipyard‘)/

5.1.18.16.4 shipyard_new_building_id

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Reference path: /HAS_ID_1_ROLE(product_definition, ‘shipyard new building id’)/

5.1.18.16.5 shipyard_project_name

AIM element: organizational_project.name
Source: ISO 10303-41
Reference path: /ORG_ASSGN(product_definition, ‘shipyard‘)/ <-

organizational_project.responsible_organizations[i]
organizational_project
{organizational_project.description = ‘shipyard project name‘}
organizational_project
organizational_project.name

5.1.18.16.6 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(product_definition)/

5.1.18.16.7 shipyard_designation to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.18.16.8 shipyard_designation to ship (as defined_for)

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product, ‘ship’)/}

5.1.18.17 WORKING_SHIP

AIM element: product_related_product_category
Source: ISO 10303-41
Reference path: {[/PROD_CAT_NAME(‘working ship‘)/]
[/CLASS(product_related_product_category, ‘working ship’, ‘shiptype’)/]
[/CLASS(product_related_product_category, ‘shiptype’, ‘functional definition’)/]
[/CLASS(product_related_product_category, ‘functional definition’, ‘definition’)/]
[/CLASS(product_related_product_category, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(product_related_product_category, ‘versionable object’)/]}

5.1.18.17.1 description

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.1)

5.1.18.17.2 has_type

AIM element: product_category.name
Source: ISO 10303-41
Reference path: /PROD_CAT_NAME(‘working ship‘)/
product_category <-
product_category_relationship.category
product_category_relationship
{product_category_relationship.name = ‘working ship types‘}
product_category_relationship.sub_category ->
product_category

{(product_category.name = ‘Tug’)
(product_category.name = ‘Sealer’)
(product_category.name = ‘Fire fighter’)
(product_category.name = ‘Drilling vessel’)
(product_category.name = ‘Pipe laying vessel’)
(product_category.name = ‘Crane vessel’)
(product_category.name = ‘Dredger’)
(product_category.name = ‘Supply vessel’)
(product_category.name = ‘Ice breaker’)
(product_category.name = ‘Fishing vessel’)
(product_category.name = ‘Floating dock’)
(product_category.name = ‘Pilot boat’)
(product_category.name = ‘Floating hotel’)
(product_category.name = ‘Well stimulation vessel’)
(product_category.name = ‘Pusher’)
(product_category.name = ‘Stern trawler’)
(product_category.name = ‘Reefer’)
(product_category.name = ‘Offshore supply vessel’)
(product_category.name = ‘Oil production vessel’)
(product_category.name = ‘Oil storage vessel’)
(product_category.name = ‘Oil production and storage vessel’)
(product_category.name = ‘Shuttle tanker’)
(product_category.name = ‘FPSO’)
(product_category.name = ‘FPGO’)
(product_category.name = ‘user defined’)}

5.1.18.17.3 user_def_function

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.2)

5.1.18.17.4 version_id

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.3)

5.1.18.17.5 working_ship to global_id (as id)

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.4)

5.1.18.17.6 working_ship to ship (as defined_for)

AIM element: /SUPERTYPE(Shiptype)/ -- (see 5.1.18.15.5)

5.1.19 Ship_measures UoF

5.1.19.1 CENTRE_LOCATION

AIM element: compound_representation_item
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.27
Reference path: /ROOT_CLASS(compound_representation_item, ‘centre location’)/

5.1.19.1.1 longitudinal_location

AIM element: value_representation_item.value_component

Source: ISO 10303-43
Reference path: compound_representation_item
compound_representation_item.item_element ->
compound_item_definition = list_representation_item
list_representation_item [i] ->
representation_item =>
{representation_item.name = 'longitudinal location'}
value_representation_item
{value_representation_item.value_component ->
measure_value = length_measure}

5.1.19.1.2 transversal_location

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: compound_representation_item
compound_representation_item.item_element ->
compound_item_definition = list_representation_item
list_representation_item [i] ->
representation_item =>
{representation_item.name = 'transversal location'}
value_representation_item
{value_representation_item.value_component ->
measure_value = length_measure}

5.1.19.1.3 vertical_location

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: compound_representation_item
compound_representation_item.item_element ->
compound_item_definition = list_representation_item
list_representation_item [i] ->
representation_item =>
{representation_item.name = 'vertical location'}
value_representation_item
{value_representation_item.value_component ->
measure_value = length_measure}

5.1.19.2 DERIVED_UNIT

NOTE See L.1.1 for additional discussion on the use of measures and units in this part of ISO 10303.

AIM element: derived_unit
Source: ISO 10303-41
Reference path: (derived_unit
/NAME_ASSGN_WITH_VAL(derived_unit, ‘airflow volume unit’)/
derived_unit.elements[i] ->
[derived_unit_element
{[derived_unit_element.unit ->
([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]
[derived_unit_element.exponent = 3]}]

[derived_unit_element
{[derived_unit_element.unit ->
([time_unit][si_unit])
([time_unit][conversion_based_unit])
([time_unit][context_dependent_unit])]
[derived_unit_element.exponent = -1]}])

(derived_unit
/NAME_ASSGN_WITH_VAL(derived_unit, 'area unit')/
derived_unit.elements[i] ->
derived_unit_element
derived_unit_element.unit ->
[([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]
[derived_unit_element.exponent = 2]

(derived_unit
/NAME_ASSGN_WITH_VAL(derived_unit, 'volume unit')/
derived_unit.elements[i] ->
derived_unit_element
derived_unit_element.unit ->
[([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]
[derived_unit_element.exponent = 3]

(derived_unit
/NAME_ASSGN_WITH_VAL(derived_unit, 'density unit')/
derived_unit.elements[i] ->
[derived_unit_element
{[derived_unit_element.unit ->
([mass_unit][si_unit])
([mass_unit][conversion_based_unit])
([mass_unit][context_dependent_unit])]
[derived_unit_element.exponent = 1]}]
[derived_unit_element
{[derived_unit_element.unit ->
([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]
[derived_unit_element.exponent = -3]}])

(derived_unit
/NAME_ASSGN_WITH_VAL(derived_unit, 'coefficient of thermal expansion unit')/
derived_unit.elements[i] ->
[derived_unit_element
{[derived_unit_element.unit ->
([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]


```
[derived_unit_element.exponent = 1]}]
[derived_unit_element
{[derived_unit_element.unit ->
([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]
[derived_unit_element.exponent = -1]}]
[derived_unit_element
{[derived_unit_element.unit ->
([thermodynamic_temperature_unit][si_unit])
([thermodynamic_temperature_unit][conversion_based_unit])
([thermodynamic_temperature_unit][context_dependent_unit])]
[derived_unit_element.exponent = -1]}])

(derived_unit
/NAME_ASSGN_WITH_VAL(derived_unit, 'coefficient of vicosity unit')/
derived_unit.elements[i] ->
[derived_unit_element
{[derived_unit_element.unit ->
([mass_unit][si_unit])
([mass_unit][conversion_based_unit])
([mass_unit][context_dependent_unit])]
[derived_unit_element.exponent = 1]}]
[derived_unit_element
{[derived_unit_element.unit ->
([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]
[derived_unit_element.exponent = -1]}]
[derived_unit_element
{[derived_unit_element.unit ->
([time_unit][si_unit])
([time_unit][conversion_based_unit])
([time_unit][context_dependent_unit])]
[derived_unit_element.exponent = -1]}])

(derived_unit
/NAME_ASSGN_WITH_VAL(derived_unit, 'specific heat capacity unit')/
derived_unit.elements[i] ->
[derived_unit_element
{[derived_unit_element.unit ->
([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]
[derived_unit_element.exponent = 2]}]
[derived_unit_element
{[derived_unit_element.unit ->
([time_unit][si_unit])
([time_unit][conversion_based_unit])
([time_unit][context_dependent_unit])]
[derived_unit_element.exponent = -2]}]
[derived_unit_element
{[derived_unit_element.unit ->
```

```
([thermodynamic_temperature_unit][si_unit])
([thermodynamic_temperature_unit][conversion_based_unit])
([thermodynamic_temperature_unit][context_dependent_unit])]
[derived_unit_element.exponent = -1]}])

(derived_unit
/NAME_ASSGN_WITH_VAL(derived_unit, 'thermal conductivity unit')/
derived_unit.elements[i] ->
[derived_unit_element
{[derived_unit_element.unit ->
([mass_unit][si_unit])
([mass_unit][conversion_based_unit])
([mass_unit][context_dependent_unit])]
[derived_unit_element.exponent = 1]}]
[derived_unit_element
{[derived_unit_element.unit ->
([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]
[derived_unit_element.exponent = 1]}]
[derived_unit_element
{[derived_unit_element.unit ->
([time_unit][si_unit])
([time_unit][conversion_based_unit])
([time_unit][context_dependent_unit])]
[derived_unit_element.exponent = -3]}]
[derived_unit_element
{[derived_unit_element.unit ->
([thermodynamic_temperature_unit][si_unit])
([thermodynamic_temperature_unit][conversion_based_unit])
([thermodynamic_temperature_unit][context_dependent_unit])]
[derived_unit_element.exponent = -1]}])

(derived_unit
/NAME_ASSGN_WITH_VAL(derived_unit, 'pressure unit')/
derived_unit.elements[i] ->
[derived_unit_element
{[derived_unit_element.unit ->
([mass_unit][si_unit])
([mass_unit][conversion_based_unit])
([mass_unit][context_dependent_unit])]
[derived_unit_element.exponent = 1]}]
[derived_unit_element
{[derived_unit_element.unit ->
([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]
[derived_unit_element.exponent = -1]}]
[derived_unit_element
{[derived_unit_element.unit ->
([time_unit][si_unit])
([time_unit][conversion_based_unit])
```

```
([time_unit][context_dependent_unit])]
[derived_unit_element.exponent = -2]}])

(derived_unit
/NAME_ASSGN_WITH_VAL(derived_unit, 'force unit')/
derived_unit.elements[i] ->
[derived_unit_element
{[derived_unit_element.unit ->
([mass_unit][si_unit])
([mass_unit][conversion_based_unit])
([mass_unit][context_dependent_unit])]
[derived_unit_element.exponent = 1]}]
[derived_unit_element
{[derived_unit_element.unit ->
([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]
[derived_unit_element.exponent = 1]}]
[derived_unit_element
{[derived_unit_element.unit ->
([time_unit][si_unit])
([time_unit][conversion_based_unit])
([time_unit][context_dependent_unit])]
[derived_unit_element.exponent = -2]}])

(derived_unit
/NAME_ASSGN_WITH_VAL(derived_unit, 'inertia moment unit')/
derived_unit.elements[i] ->
derived_unit_element
{[derived_unit_element.unit ->
([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]
[derived_unit_element.exponent = 4]})

(derived_unit
/NAME_ASSGN_WITH_VAL(derived_unit, 'moment unit')/
derived_unit.elements[i] ->
[derived_unit_element
{[derived_unit_element.unit ->
([mass_unit][si_unit])
([mass_unit][conversion_based_unit])
([mass_unit][context_dependent_unit])]
[derived_unit_element.exponent = 1]}]
[derived_unit_element
{[derived_unit_element.unit ->
([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]
[derived_unit_element.exponent = 2]}]
[derived_unit_element
{[derived_unit_element.unit ->
([time_unit][si_unit])
```

([time_unit][conversion_based_unit])
([time_unit][context_dependent_unit])]
[derived_unit_element.exponent = -2]}])

(derived_unit
/NAME_ASSGN_WITH_VAL(derived_unit, 'speed unit')/
derived_unit.elements[i] ->
[derived_unit_element
{[derived_unit_element.unit ->
([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]
[derived_unit_element.exponent = 1]}]
[derived_unit_element
{[derived_unit_element.unit ->
([time_unit][si_unit])
([time_unit][conversion_based_unit])
([time_unit][context_dependent_unit])]
[derived_unit_element.exponent = -1]}])

5.1.19.3 NAMED_UNIT

NOTE See L.1.1 for additional discussion on the use of measures and units in this part of ISO 10303.

AIM element: named_unit
Source: ISO 10303-41
Reference path: named_unit =>
[(si_unit)
(conversion_based_unit)
(context_dependent_unit)]
[(length_unit)
(luminous_intensity_unit)
(mass_unit)
(plane_angle_unit)
(ratio_unit)
(thermodynamic_temperature_unit)]

5.1.19.4 PRECISION

AIM element: uncertainty_measure_with_unit
Source: ISO 10303-43

5.1.19.4.1 minimum_point_spacing

AIM element: measure_with_unit.value_component
Source: ISO 10303-43
Reference path: uncertainty_measure_with_unit
{uncertainty_measure_with_unit.name = 'distance_accuracy_value'}
uncertainty_measure_with_unit <=
measure_with_unit
{[measure_with_unit.unit_component ->
unit
unit = named_unit

named_unit =>
([length_unit][si_unit])
([length_unit][conversion_based_unit])
([length_unit][context_dependent_unit])]
[measure_with_unit.value_component ->
measure_value = positive_length_measure]}

5.1.20 Spaces UoF

5.1.20.1 COMPARTMENT

AIM element: product_definition
Source: ISO 10303-41
Reference path: {[/CLASS(product_definition, 'compartment', 'space')/]
[/CLASS(product_definition, 'space', 'item')/]
[/CLASS(product_definition, 'item', 'definable object')/]
[/ROOT_CLASS(product_definition, 'definable object')/]}

5.1.20.1.1 description

AIM element: product_definition.description
Source: ISO 10303-41

5.1.20.1.2 name

AIM element: product_definition.name
Source: ISO 10303-41
Reference path: /NAME_ASSGN(product_definition)/

5.1.20.1.3 compartment to external_reference (as documentation)

#1: If as documentation refers to an External_reference

AIM element: PATH
Reference path: product_definition
product_definition = external_identification_item
external_identification_item <-
applied_external_identification_assignment.items[i]
applied_external_identification_assignment

#2: If as documentation refers to a Document_reference_with_address

AIM element: PATH
Reference path: product_definition
/DOC_REF(product_definition,’documentation’)/
document
{/CLASS_ID(document, ‘document reference with address’)/}

5.1.20.1.4 compartment to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition
identification_item <-

applied_identification_assignment.items[i]
applied_identification_assignment

5.1.20.1.5 compartment to ship (as ship_context)

AIM element: PATH
Reference path: product_definition
product_definition.formation ->
product_definition_formation
{product_definition_formation.id = 'compartmentation'}
product_definition_formation.of_product ->
product
{/CLASS_ID(product, 'ship')/}

5.1.20.2 COMPARTMENT_FUNCTIONAL_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.81
Reference path: {[/CLASS(property_definition, 'compartment functional definition',
'functional definition')/]
[/CLASS(property_definition, 'functional definition', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.20.2.1 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.20.2.2 used_for

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.122, 5.2.4.230
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘compartment function parameters’, ‘used for’)/
{(descriptive_representation_item.description = 'access trunk')
(descriptive_representation_item.description = 'aft peak tank')
(descriptive_representation_item.description = 'auxiliary engine room')
(descriptive_representation_item.description = 'ballast tank')
(descriptive_representation_item.description = 'battery room')
(descriptive_representation_item.description = 'berthing compartment')
(descriptive_representation_item.description = 'boiler room')
(descriptive_representation_item.description = 'bottom wing tank')
(descriptive_representation_item.description = 'bow thruster room')
(descriptive_representation_item.description = 'cabin')
(descriptive_representation_item.description = 'cargo compartment')
(descriptive_representation_item.description = 'centre tank')
(descriptive_representation_item.description = 'chainlocker')
(descriptive_representation_item.description = 'cofferdam')
(descriptive_representation_item.description = 'compressor room')
(descriptive_representation_item.description = 'control')

(descriptive_representation_item.description = 'crossover tank')
(descriptive_representation_item.description = 'deck tank')
(descriptive_representation_item.description = 'deep tank')
(descriptive_representation_item.description = 'diving well')
(descriptive_representation_item.description = 'double bottom and side tank')
(descriptive_representation_item.description = 'double bottom tank')
(descriptive_representation_item.description = 'double side tank')
(descriptive_representation_item.description = 'drill well')
(descriptive_representation_item.description = 'duct keel')
(descriptive_representation_item.description = 'electric motor room')
(descriptive_representation_item.description = 'emergency fire pump room')
(descriptive_representation_item.description = 'equipment room')
(descriptive_representation_item.description = 'escape trunk')
(descriptive_representation_item.description = 'fore peak tank')
(descriptive_representation_item.description = 'forecastle')
(descriptive_representation_item.description = 'habitable compartment')
(descriptive_representation_item.description = 'heeling tank')
(descriptive_representation_item.description = 'hopper tank')
(descriptive_representation_item.description = 'insulated tank')
(descriptive_representation_item.description = 'lounge')
(descriptive_representation_item.description = 'machinery compartment')
(descriptive_representation_item.description = 'main engine room')
(descriptive_representation_item.description = 'medical')
(descriptive_representation_item.description = 'passageway')
(descriptive_representation_item.description = 'poop')
(descriptive_representation_item.description = 'pump room')
(descriptive_representation_item.description = 'rudder trunk')
(descriptive_representation_item.description = 'separator room')
(descriptive_representation_item.description = 'settling tank')
(descriptive_representation_item.description = 'shaft tunnel')
(descriptive_representation_item.description = 'side tank')
(descriptive_representation_item.description = 'side wing tank')
(descriptive_representation_item.description = 'stabiliser room')
(descriptive_representation_item.description = 'stability tank')
(descriptive_representation_item.description = 'steering gear room')
(descriptive_representation_item.description = 'stern tank')
(descriptive_representation_item.description = 'stool tank')
(descriptive_representation_item.description = 'tank')
(descriptive_representation_item.description = 'thruster room')
(descriptive_representation_item.description = 'top wing tank')
(descriptive_representation_item.description = 'trimming tank')
(descriptive_representation_item.description = 'trunk')
(descriptive_representation_item.description = 'void')
(descriptive_representation_item.description = 'waterjet room')
(descriptive_representation_item.description = 'wheelhouse')
(descriptive_representation_item.description = 'wing tank')
(descriptive_representation_item.description = 'user defined')}

5.1.20.2.3 user_def_function

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.191

Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘compartment function parameters’, ‘user def function’)/

5.1.20.2.4 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.20.2.5 compartment_functional_definition to compartment (as defined_for)

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/

5.1.20.2.6 compartment_functional_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.20.3 DECK_ZONE

AIM element: product_definition
Source: ISO 10303-41
Reference path: {[/CLASS(product_definition, 'deck zone', 'space')/]
[/CLASS(product_definition, 'space', 'item')/]
[/CLASS(product_definition, 'item', 'definable object')/]
[/ROOT_CLASS(product_definition, 'definable object')/]}

5.1.20.3.1 description

AIM element: product_definition.description
Source: ISO 10303-41

5.1.20.3.2 name

AIM element: product_definition.name
Source: ISO 10303-41
Reference path: /NAME_ASSGN(product_definition)/

5.1.20.3.3 deck_zone to external_reference (as documentation)

#1: If as documentation refers to an External_reference

AIM element: PATH
Reference path: product_definition
product_definition = external_identification_item
external_identification_item <-

applied_external_identification_assignment.items[i]
applied_external_identification_assignment

#2: If as documentation refers to a Document_reference_with_address

AIM element: PATH
Reference path: product_definition
/DOC_REF(product_definition,'documentation')/
document
{/CLASS_ID(document, 'document reference with address')/}

5.1.20.3.4 deck_zone to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.20.3.5 deck_zone to ship (as ship_context)

AIM element: PATH
Reference path: product_definition
product_definition.formation ->
product_definition_formation
{product_definition_formation.id = 'deck zones'}
product_definition_formation.of_product ->
product
{/CLASS_ID(product, 'ship')/}

5.1.20.4 DECK_ZONE_FUNCTIONAL_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.85
Reference path: {[/CLASS(property_definition, 'deck zone functional definition',
'functional definition')/]
[/CLASS(property_definition, 'functional definition', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.20.4.1 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.20.4.2 used_for

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.147, 5.2.4.230
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('deck zone function parameters', 'used for')/

{(descriptive_representation_item.description = 'boat deck zone')
(descriptive_representation_item.description = 'bridge deck zone')
(descriptive_representation_item.description = 'car deck zone')
(descriptive_representation_item.description = 'deck compartment arrangement')
(descriptive_representation_item.description = 'deck loading zone')
(descriptive_representation_item.description = 'main deck zone')
(descriptive_representation_item.description = 'sun deck zone')
(descriptive_representation_item.description = 'topside zone')
(descriptive_representation_item.description = 'weather deck zone')
(descriptive_representation_item.description = 'user defined')}

5.1.20.4.3 user_def_function

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.199
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('deck zone function parameters', 'user def function')/

5.1.20.4.4 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.20.4.5 deck_zone_functional_definition to deck_zone (as defined_for)

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/

5.1.20.4.6 deck_zone_functional_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.20.5 SPACE

#1: space is a compartment
#2: space is a deck_zone
#3: space is a zone

AIM element: #1: /SUBTYPE(compartment)/ -- (see 5.1.20.1)
#2: /SUBTYPE(deck_zone)/ -- (see 5.1.20.3)
#3: /SUBTYPE(zone)/ -- (see 5.1.20.6)

5.1.20.5.1 description

AIM element: #1: /SUBTYPE(compartment)/ -- (see 5.1.20.1.1)

#2: /SUBTYPE(deck_zone)/ -- (see 5.1.20.3.1)
#3: /SUBTYPE(zone)/ -- (see 5.1.20.6.1)

5.1.20.5.2 name

AIM element: #1: /SUBTYPE(compartment)/ -- (see 5.1.20.1.2)
#2: /SUBTYPE(deck_zone)/ -- (see 5.1.20.3.2)
#3: /SUBTYPE(zone)/ -- (see 5.1.20.6.2)

5.1.20.5.3 space to external_reference (as documentation)

AIM element: #1: /SUBTYPE(compartment)/ -- (see 5.1.20.1.3)
#2: /SUBTYPE(deck_zone)/ -- (see 5.1.20.3.3)
#3: /SUBTYPE(zone)/ -- (see 5.1.20.6.3)

5.1.20.5.4 space to global_id (as id)

AIM element: #1: /SUBTYPE(compartment)/ -- (see 5.1.20.1.4)
#2: /SUBTYPE(deck_zone)/ -- (see 5.1.20.3.4)
#3: /SUBTYPE(zone)/ -- (see 5.1.20.6.4)

5.1.20.5.5 space to ship (as ship_context)

AIM element: #1: /SUBTYPE(compartment)/ -- (see 5.1.20.1.5)
#2: /SUBTYPE(deck_zone)/ -- (see 5.1.20.3.5)
#3: /SUBTYPE(zone)/ -- (see 5.1.20.6.5)

5.1.20.6 ZONE

AIM element: product_definition
Source: ISO 10303-41
Reference path: {[/CLASS(product_definition, 'zone', 'space')/]
[/CLASS(product_definition, 'space', 'item')/]
[/CLASS(product_definition, 'item', 'definable object')/]
[/ROOT_CLASS(product_definition, 'definable object')/]}

5.1.20.6.1 description

AIM element: product_definition.description
Source: ISO 10303-41

5.1.20.6.2 name

AIM element: product_definition.name
Source: ISO 10303-41
Reference path: /NAME_ASSGN(product_definition)/

5.1.20.6.3 zone to external_reference (as documentation)

#1: If as documentation refers to an External_reference

AIM element: PATH
Reference path: product_definition
product_definition = external_identification_item
external_identification_item <-
applied_external_identification_assignment.items[i]

applied_external_identification_assignment

#2: If as documentation refers to a Document_reference_with_address

AIM element: PATH
Reference path: product_definition
/DOC_REF(product_definition,'documentation')/
document
{/CLASS_ID(document, 'document reference with address')/}

5.1.20.6.4 zone to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.20.6.5 zone to ship (as ship_context)

AIM element: PATH
Reference path: product_definition
product_definition.formation ->
product_definition_formation
{product_definition_formation.id = 'zones'}
product_definition_formation.of_product ->
product
{/CLASS_ID(product, 'ship')/}

5.1.20.7 ZONE_FUNCTIONAL_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.92
Reference path: {[/CLASS(property_definition, 'zone functional definition', 'functional definition')/]
[/CLASS(property_definition, 'functional definition', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.20.7.1 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.20.7.2 used_for

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.181, 5.2.4.230
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('zone function parameters', 'used for')/
{(descriptive_representation_item.description = 'arrangement zone')
(descriptive_representation_item.description = 'bottom external zone')

(descriptive_representation_item.description = 'bow external zone')
(descriptive_representation_item.description = 'bridging structure zone')
(descriptive_representation_item.description = 'damage control zone')
(descriptive_representation_item.description = 'deckhouse zone')
(descriptive_representation_item.description = 'design zone')
(descriptive_representation_item.description = 'double bottom zone')
(descriptive_representation_item.description = 'external zone')
(descriptive_representation_item.description = 'fire zone')
(descriptive_representation_item.description = 'hatch zone')
(descriptive_representation_item.description = 'pressure zone')
(descriptive_representation_item.description = 'ship side external zone')
(descriptive_representation_item.description = 'stern external zone')
(descriptive_representation_item.description = 'subsafe zone')
(descriptive_representation_item.description = 'superstructure zone')
(descriptive_representation_item.description = 'tank top zone')
(descriptive_representation_item.description = 'tween deck zone')
(descriptive_representation_item.description = 'user defined')}

5.1.20.7.3 user_def_function

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.213
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘zone function parameters’, ‘user def function’)

5.1.20.7.4 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.20.7.5 zone_functional_definition to deck_zone (as defined_for)

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/

5.1.20.7.6 zone_functional_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.21 Tonnage UoF

5.1.21.1 COMPARTMENT_GROUP

AIM element: [product_definition]
[group]
Source: ISO 10303-41

Rules: 5.2.4.183, 5.2.4.123
Reference path: {[/LINK_TO_GROUP(product_definition)/]
[/ROOT_CLASS(product_definition, 'compartment group')/]}

5.1.21.1.1 tonnage_volume

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘compartment group parameters’, ‘tonnage volume‘, volume_measure)/

5.1.21.1.2 compartment_group to compartment (as compartment)

AIM element: PATH
Reference path: group <-
/GROUPS(product_definition, ’compartment group’)/
{/CLASS_ID(product_definition, 'compartment’)/)}

5.1.21.2 COMPENSATED_GROSS_TONNAGE

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.140
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘compensated gross tonnage’)/}

5.1.21.2.1 compensation_factor

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_VAL_REP_ITEM('compensation factor', numeric_measure)/

5.1.21.2.2 tonnage_value

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation
representation.items[i] ->
/REP_TO_SPECIAL_VAL_REP_ITEM('tonnage value', 'force unit')/

5.1.21.2.3 compensated_gross_tonnage to gross_tonnage (as gross_tonnage_measurement)

AIM element: PATH
Reference path: property_definition_representation
property_definition_representation.used_representation ->
representation <-
representation_relationship.rep_1
representation_relationship
{representation_relationship.name = ‘gross tonnage measurement’}
representation_relationship

representation_relationship.rep_2 ->
representation <-
property_definition_representation.used_representation ->
property_definition_representation
{/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘gross tonnage’)/}

5.1.21.3 GROSS_TONNAGE

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.154
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘gross tonnage’)/}

5.1.21.3.1 date_of_measurement

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-215
Rules: 5.2.4.95
Reference path: /DAT_TIME_ASSGN(property_definition_representation, ‘date of measurement’)/

5.1.21.3.2 overdeck_tonnage

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_SPECIAL_VAL_REP_ITEM('overdeck tonnage', 'force unit')/

5.1.21.3.3 tonnage_value

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_SPECIAL_VAL_REP_ITEM('tonnage value', 'force unit')/

5.1.21.3.4 underdeck_tonnage

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_SPECIAL_VAL_REP_ITEM('underdeck tonnage', 'force unit')/

5.1.21.3.5 gross_tonnage to compartment_group (as spaces_included)

AIM element: PATH
Reference path: property_definition_representation ->
represented_definition = property_definition
property_definition
{/CLASS_ID(property_definition, ‘tonnage definition’)/}
property_definition <-
property_definition_relationship.relating_property_definition

property_definition_relationship
{property_definition_relationship.name = ‘spaces included’}
property_definition_relationship.related_definition ->
property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
product_definition
{/CLASS_ID(product_definition, ‘compartment group’)/}

5.1.21.4 NET_TONNAGE

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.165
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, ‘net tonnage’)/}

5.1.21.4.1 date_of_measurement

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-215
Rules: 5.2.4.96
Reference path: /DAT_TIME_ASSGN(property_definition_representation, ‘date of measurement’)/

5.1.21.4.2 tonnage_value

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_SPECIAL_VAL_REP_ITEM('tonnage value', 'force unit')/

5.1.21.4.3 net_tonnage to compartment_group (as spaces_included)

AIM element: PATH
Reference path: property_definition_representation ->
represented_definition = property_definition
property_definition
{/CLASS_ID(property_definition, ‘tonnage definition’)/}
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = ‘spaces included’}
property_definition_relationship.related_definition ->
property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
product_definition
{/CLASS_ID(product_definition, ‘compartment group’)/}

5.1.21.5 TONNAGE_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.91, 5.2.4.174, 5.2.4.226, 5.2.4.82
Reference path: {[/CLASS(property_definition, ‘tonnage definition’, ‘definition’)/]
[/CLASS(property_definition, ‘definition’, ‘versionable object’)/]
[/ROOT_CLASS(property_definition, ‘versionable object’)/]}

5.1.21.5.1 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.21.5.2 tonnage_regulation

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘tonnage definition parameters’, ‘tonnage regulation‘)/
{(descriptive_representation_item.description = 'convention1969')
(descriptive_representation_item.description = 'panama')
(descriptive_representation_item.description = 'suez')}

5.1.21.5.3 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-215
Rules: 5.2.4.251
Reference path: /VERSION_ID(property_definition)/

5.1.21.5.4 tonnage_definition to compartment (as spaces_excluded)

AIM element: PATH
Reference path: property_definition
{/CLASS_ID(property_definition, ‘tonnage definition’)/}
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
product_definition <-
product_definition_relationship.relating_product_definition
product_definition_relationship
{product_definition_relationship.name = ‘space excluded’}
product_definition_relationship.related_product_definition ->
product_definition
{{/CLASS_ID(product_definition, ‘compartment’)/}

5.1.21.5.5 tonnage_definition to compensated_gross_tonnage (as compensated_gross_tonnage)

AIM element: PATH
Reference path: property_definition
represented_definition = property_definition
represented_definition <-
/PDR_NAME('compensated gross tonnage')/

5.1.21.5.6 tonnage_definition to derived_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.21.5.7 tonnage_definition to document_reference_with_address (as certificate)

AIM element: PATH
Reference path: property_definition
/DOC_REF(property_definition,'certificate')/
document
{/CLASS_ID(document, 'document reference with address')/}

5.1.21.5.8 tonnage_definition to global_id (as id)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.97
Reference path: property_definition
identification_item = property_definition
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.21.5.9 tonnage_definition to gross_tonnage (as gross_tonnage)

AIM element: PATH
Rules: 5.2.4.86
Reference path: property_definition
represented_definition = property_definition
represented_definition <-
/PDR_NAME('gross tonnage')/

5.1.21.5.10 tonnage_definition to named_unit (as local_units)

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.21.5.11 tonnage_definition to net_tonnage (as net_tonnage)

AIM element: PATH
Rules: 5.2.4.89
Reference path: property_definition
represented_definition = property_definition
represented_definition <-
/PDR_NAME('net tonnage')/

5.1.21.5.12 tonnage_definition to ship (as defined_for)

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/
/PROD_DEF_PRODUCT/
{/CLASS_ID(product, 'ship')}

5.1.21.6 TONNAGE_MEASUREMENT

AIM element: property_definition_representation
Source: ISO 10303-41
Rules: 5.2.4.183, 5.2.4.175
Reference path: {/NAME_ASSGN_WITH_VAL(property_definition_representation, 'tonnage measurement')/}

5.1.21.6.1 date_of_measurement

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-215
Rules: 5.2.4.94
Reference path: /DAT_TIME_ASSGN(property_definition_representation, 'date of measurement')/

5.1.21.6.2 tonnage_value

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: property_definition_representation
property_definition_representation.used_representation ->
/REP_TO_SPECIAL_VAL_REP_ITEM('tonnage value', 'force unit')/

5.1.21.6.3 tonnage_measurement to compartment_group (as spaces_included)

AIM element: PATH
Reference path: property_definition_representation
property_definition_representation ->
represented_definition = property_definition
property_definition
{/CLASS_ID(property_definition, 'tonnage definition')/}
property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = 'spaces included'}
property_definition_relationship.related_definition ->
property_definition
property_definition.definition ->
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
product_definition
{/CLASS_ID(product_definition, 'compartment group')/}

5.1.22 Weights UoF

5.1.22.1 MOMENT_3D

AIM element: representation

Source: ISO 10303-43
Reference path: {representation
representation.name = ‘moment 3d‘}

5.1.22.1.1 longitudinal_moment

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.163
Reference path: /REP_TO_SPECIAL_VAL_REP_ITEM(‘longitudinal moment‘, 'moment unit')/

5.1.22.1.2 transverse_moment

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.163
Reference path: /REP_TO_SPECIAL_VAL_REP_ITEM(‘transverse moment‘, 'moment unit')/

5.1.22.1.3 vertical_moment

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.163
Reference path: /REP_TO_SPECIAL_VAL_REP_ITEM(‘vertical moment‘, 'moment unit')/

5.1.22.1.4 moment_3d to centre_location (as origin)

AIM element: PATH
Rules: 5.2.4.183, 5.2.4.163
Reference path: /REP_ITEM(‘origin‘)/
compound_representation_item
{/CLASS_ID(compound_representation_item, 'centre location')/

5.1.22.2 WEIGHT_AND_CENTRE_OF_GRAVITY

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.99
Reference path: {/ROOT_CLASS(property_definition, ‘weight and centre of gravity‘)/}

5.1.22.2.1 mass

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Rules: 5.2.4.183, 5.2.4.215
Reference path: /PROP_DEF_TO_VAL_REP_ITEM(‘weight and centre of gravity’, ‘mass‘,
mass_measure)/

5.1.22.2.2 weight_and_centre_of_gravity to centre_location (as centre_of_gravity)

AIM element: PATH
Rules: 5.2.4.183, 5.2.4.215
Reference path: /PROP_DEF_TO_REP/
/REP_ITEM(‘centre of gravity‘)/

compound_representation_item
{/CLASS_ID(compound_representation_item, 'centre location')/

5.1.22.2.3 weight_and_centre_of_gravity to moment_3d (as moment)

AIM element: PATH
Reference path: /PROP_DEF_TO_REP/
{representation
representation.name = ‘moment 3d‘}

5.2 AIM EXPRESS short listing

This clause specifies the EXPRESS schema that uses elements from the integrated resources and the AIC's and contains the types, entity specializations, rules and functions that are specific to this part of ISO 10303. This clause also specifies modifications to the text for the constructs that are imported from the integrated resources and the AIC's. The definitions and EXPRESS provided in the integrated resources for constructs used in the AIM may include select list items and subtypes that are not imported into the AIM. Requirements stated in the integrated resources that refer to select list items apply exclusively to those items that are imported into the AIM.

```
*)
SCHEMA Ship_arrangement_schema;

    USE FROM aic_non_manifold_surface                -- ISO 10303-508
    (non_manifold_surface_shape_representation);

    USE FROM aic_topologically_bounded_surface       -- ISO 10303-511
    (advanced_face);

    USE FROM action_schema                           -- ISO 10303-41
    (action,
    action_method,
    action_request_solution,
    executed_action,
    versioned_action_request);

   USE FROM application_context_schema               -- ISO 10303-41
   (application_context,
   application_protocol_definition);

    USE FROM approval_schema                         -- ISO 10303-41
   (approval,
    approval_date_time,
    approval_person_organization,
    approval_status);

   USE FROM basic_attribute_schema                   -- ISO 10303-41
   (object_role);

   USE FROM date_time_schema                         -- ISO 10303-41
   (calendar_date,
   date_and_time,
   date_time_role,
   ordinal_date,
   week_of_year_and_day_date);

   USE FROM document_schema                          -- ISO 10303-41
   (document,
    document_representation_type,
    document_usage_constraint);
```

```
  USE FROM external_reference_schema                -- ISO 10303-41
  (external_source,
  external_source_relationship);

 USE FROM effectivity_schema                       -- ISO 10303-41
  (effectivity,
   serial_numbered_effectivity);

 USE FROM geometric_model_schema                    -- ISO 10303-42
  (face_based_surface_model);

 REFERENCE FROM geometry_schema                      -- ISO 10303-42
  (dummy_gri);

 USE FROM geometry_schema                           -- ISO 10303-42
  (axis2_placement_2d,
   axis2_placement_3d,
   bezier_curve,
   bezier_surface,
   b_spline_curve,
   b_spline_curve_with_knots,
   b_spline_surface,
   b_spline_surface_with_knots,
   bounded_curve,
   bounded_pcurve,
   bounded_surface_curve,
   cartesian_point,
   cartesian_transformation_operator_3d,
   circle,
   composite_curve_on_surface,
   conical_surface,
   curve,
   curve_replica,
   cylindrical_surface,
   degenerate_pcurve,
   degenerate_toroidal_surface,
   direction,
   ellipse,
   evaluated_degenerate_pcurve,
   geometric_representation_context,
   geometric_representation_item,
   hyperbola,
   intersection_curve,
   line,
   offset_curve_3d,
   offset_surface,
   oriented_surface,
   point_on_curve,
   point_on_surface,
   parabola,
   pcurve,
   plane,
   polyline,
   quasi_uniform_curve,
   quasi_uniform_surface,
   rational_b_spline_curve,
   rational_b_spline_surface,
   seam_curve,
   spherical_surface,
   surface,
   surface_curve,
   surface_of_linear_extrusion,
   surface_of_revolution,
   surface_replica,
```



```
  swept_surface,
  toroidal_surface,
  uniform_curve,
  uniform_surface,
  vector);

 USE FROM group_schema                                  -- ISO 10303-41
 (group,
  group_relationship);

 USE FROM management_resources_schema                   -- ISO 10303-41
 (action_assignment,
  action_request_assignment,
  approval_assignment,
  classification_assignment,
  classification_role,
  date_and_time_assignment,
  document_reference,
  effectivity_assignment,
  external_identification_assignment,
  group_assignment,
  identification_assignment,
  identification_assignment_relationship,
  identification_role,
  organization_assignment,
  person_assignment,
  person_and_organization_assignment);

 USE FROM material_property_definition_schema           -- ISO 10303-41
 (property_definition_relationship);

USE FROM measure_schema                                 -- ISO 10303-41
  (amount_of_substance_measure,
  area_measure,
  context_dependent_measure,
  context_dependent_unit,
  count_measure,
  derived_unit,
  electric_current_measure,
  derived_unit_element,
  global_unit_assigned_context,
  length_measure,
  length_unit,
  luminous_intensity_measure,
  luminous_intensity_unit,
  mass_measure,
  mass_unit,
  named_unit,
  parameter_value,
  plane_angle_measure,
  plane_angle_unit,
  positive_length_measure,
  positive_plane_angle_measure,
  ratio_measure,
  ratio_unit,
  si_unit,
  solid_angle_measure,
  thermodynamic_temperature_measure,
  thermodynamic_temperature_unit,
  time_measure,
  time_unit,
  volume_measure);

 USE FROM person_organization_schema                    -- ISO 10303-41
 (address,
  person,
```

```
    person_and_organization,
    person_and_organization_role,
    personal_address,
    organization,
    organizational_address,
    organizational_project);

   USE FROM product_definition_schema                -- ISO 10303-41
   (product,
   product_category,
   product_definition,
   product_definition_relationship,
   product_definition_with_associated_documents,
   product_related_product_category);

   USE FROM product_property_definition_schema       -- ISO 10303-41
   (characterized_product_definition,
    characterized_object,
    product_definition_shape,
    property_definition,
    shape_aspect);

   USE FROM product_property_representation_schema   -- ISO 10303-41
  (property_definition_representation,
    shape_definition_representation,
    shape_representation);

   USE FROM qualified_measure_schema                 -- ISO 10303-45
   (descriptive_representation_item);

   USE FROM representation_schema                    -- ISO 10303-43
    (compound_item_definition,
     compound_representation_item,
     global_uncertainty_assigned_context,
     definitional_representation,
     founded_item,
     item_defined_transformation,
     list_representation_item,
     mapped_item,
     parametric_representation_context,
     representation,
     representation_item,
     representation_map,
     representation_relationship,
     set_representation_item,
     uncertainty_measure_with_unit,
     value_representation_item);

   USE FROM support_resource_schema                   -- ISO 10303-41
     (label);

  REFERENCE FROM topology_schema                      -- ISO 10303-42
   (dummy_tri);

   USE FROM topology_schema                           -- ISO 10303-42
   (edge,
    edge_curve,
    edge_loop,
    face_bound,
    face_outer_bound,
    face_surface,
    oriented_edge,
    path,
    poly_loop,
```



```
    oriented_path,
    closed_shell,
    connected_face_set,
    face,
    shell,
    open_shell,
    oriented_face,
    vertex_loop,
    vertex_point);

(*
```

NOTE - The schemas referenced above can be found in the following parts of ISO 10303:

aic_non_manifold_surface	ISO 10303-508
aic_topological_bounded_surface	ISO 10303-511
action_schema	ISO 10303-41
application_context_schema	ISO 10303-41
approval_schema	ISO 10303-41
basic_attribute_schema	ISO 10303-41
date_time_schema	ISO 10303-41
document_schema	ISO 10303-41
effectivity_schema	ISO 10303-41
external_reference_schema	ISO 10303-41
geometric_model_schema	ISO 10303-42
geometry_schema	ISO 10303-42
group_schema	ISO 10303-41
management_resources_schema	ISO 10303-41
material_property_definition_schema	ISO 10303-41
measure_schema	ISO 10303-41
person_organization_schema	ISO 10303-41
product_definition_schema	ISO 10303-41
product_property_definition_schema	ISO 10303-41
product_property_representation_schema	ISO 10303-41
qualified_measure_schema	ISO 10303-45
representation_schema	ISO 10303-43
support_resource_schema	ISO 10303-41
topology_schema	ISO 10303-42

5.2.1 Fundamental concepts and assumptions

5.2.2 Ship arrangement types

5.2.2.1 Ship arrangement type definitions

5.2.2.1.1 action_item

An action_item identifies an **action_request_solution, document, executed_action, group, product**, **product_definition**, **product_definition_relationship**, **product_definition_shape**, **product_related_product_category**, and **property_definition** to which an **action** may be assigned.

EXPRESS specification:

```
*)
TYPE action_item = SELECT
    (action_request_solution,
     document,
     executed_action,
     group,
     product,
      product_definition,
      product_definition_relationship,
      product_definition_shape,
      product_related_product_category,
     property_definition
      );
END_TYPE;
(*
```

5.2.2.1.2 action_request_item

An **action_request_item** identifies an **action** or **executed_action** to which an action_request may be assigned.

EXPRESS specification:

```
*)
TYPE action_request_item = SELECT(
   action, executed_action);
END_TYPE;
(*
```

5.2.2.1.3 approval_item

An **approval_item** identifies a **product_definition**, **product_definition_shape**, **product_related_-product_category**, or **property_definition** to which an approval may be assigned.

EXPRESS specification:

```
*)
TYPE approval_item = SELECT(
   product_definition,
   product_definition_shape,
   product_related_product_category,
   property_definition
   );
END_TYPE;
(*
```


5.2.2.1.4 classification_item

A classification_item identifies an **action**, **action_request_solution**, **applied_action_request_-assignment**, **approval**, **compound_representation_item**, **document**, **executed_action**, **external_source**, **group**, **identification_assignment_relationship**, **product**, **product_definition**, **product_definition_relationship**, **product_definition_shape**, **product_-related_product_category**, **property_definition**, **property_definition_representation,** **representation**, **shape_aspect**, or **versioned_action_request** to which a classification may be assigned.

EXPRESS specification:

```
*)
TYPE classification_item = SELECT(
   action,
   action_request_solution,
   applied_action_request_assignment,
    approval,
   compound_representation_item,
   document,
   executed_action,
   external_source,
   group,
   identification_assignment_relationship,
   product,
   product_definition,
   product_definition_relationship,
   product_definition_shape,
   product_related_product_category,
   property_definition,
   property_definition_representation,
   representation,
   shape_aspect,
      versioned_action_request);
END_TYPE;
(*
```

5.2.2.1.5 date_and_time_item

A date_and_time_item identifies an **action**, **action_request_solution**, **executed_action**, **product_-definition**, **property_definition**, **property_definition_representation**, or **versioned_action_-request** to which a date may be assigned.

EXPRESS specification:

```
*)
TYPE date_and_time_item = SELECT(
   action,
   action_request_solution,
   executed_action ,
   product_definition,
   property_definition,
   property_definition_representation,
   versioned_action_request);
END_TYPE;
(*
```

5.2.2.1.6 document_reference_item

A **document_reference_item** identifies an **action**, **product**, **product_definition**, or **property_-definition** to which a document may be assigned.

EXPRESS specification:

```
*)
TYPE document_reference_item = SELECT (
   action,
   product,
   product_definition,
   property_definition);
END_TYPE;
(*
```

5.2.2.2 effectivity_item

An **effectivity_item** identifies a **product_definition**, **product_definition_shape**, **product_related_-product_category**, or **property_definition** to which an effectivity may be assigned.

EXPRESS specification:

```
*)
TYPE effectivity_item = SELECT(
   product_definition,
   product_definition_shape,
   product_related_product_category,
   property_definition
  );
END_TYPE;

(*
```

5.2.2.2.1 external_identification_item

An **external_identification_item** identifies an **action**, **document**, **product**, **product_definition, property_definition**, or **shape_aspect** to which an external_identification may be assigned.

EXPRESS specification:

```
*)
TYPE external_identification_item = SELECT(
  action,
  document,
  product,
  product_definition,
  property_definition,
  shape_aspect);
END_TYPE;
(*
```

5.2.2.2.2 group_item

A group_item identifies an **applied_external_identification_assignment**, **approval**, **document**, **group**, **identification_assignment_relationship**, **product, product_definition**, **product_-**

definition_relationship, **product_definition_shape**, **product_related_product_category**, or **property_definition** to which a group may be assigned.

EXPRESS specification:

```
*)
TYPE group_item = SELECT(
   applied_external_identification_assignment,
   approval,
   document,
   group,
   identification_assignment_relationship,
   product,
   product_definition,
   product_definition_relationship,
   product_definition_shape,
   product_related_product_category,
   property_definition);
END_TYPE;
(*
```

5.2.2.2.3 identification_item

An **identification_item** identifies an **action**, **action_request_solution**, **document**, **executed_action**, **group**, **product**, **product_definition**, **product_definition_relationship**, **product_definition_shape**, **product_related_product_category**, **property_definition**, or **versioned_action_request** to which an identification may be assigned.

EXPRESS specification:

```
*)
TYPE identification_item = SELECT(
   action,
   action_request_solution,
   document,
   executed_action,
   group,
   product,
   product_definition,
   product_definition_relationship,
   product_definition_shape,
   product_related_product_category,
   property_definition,
   versioned_action_request);
END_TYPE;
(*
```

5.2.2.2.4 organization_item

An **organization_item** identifies a **document**, **product_definition**, or **property_definition** to which an organization may be identified.

EXPRESS specification:

```
*)
TYPE organization_item = SELECT(
   document,
   product_definition,
```

```
   property_definition);
END_TYPE;
(*
```

5.2.2.2.5 person_item

A **person_item** identifies a **document** to which a person may be identified.

EXPRESS specification:

```
*)
TYPE person_item = SELECT(
   document);
END_TYPE;
(*
```

5.2.2.2.6 person_and_organization_item

A **person_and_organization_item** identifies an **action**, **action_request_solution**, **document**, **executed_action**, or **versioned_action_request** to which a person_and_organization may be identified.

EXPRESS specification:

```
*)
TYPE person_and_organization_item = SELECT(
   action,
   action_request_solution,
   document,
   executed_action,
   versioned_action_request);
END_TYPE;
(*
```

5.2.3 Ship arrangement entities

5.2.3.1 Ship arrangement entity definitions

5.2.3.1.1 applied_action_assignment

An **applied_action_assignment** specifies those **action_item**s to which an **action_assignment** is assigned.

EXPRESS specification:

```
*)
ENTITY applied_action_assignment
  SUBTYPE OF (action_assignment);
  items : SET [1:?] OF action_item;
END_ENTITY;
(*
```

Attribute definition:

items: the set of **action_item** for which a particular **action_assignment** is applicable.

5.2.3.1.2 applied_action_request_assignment

An **applied_action_request_assignment** specifies those **action_request_item**s to which a referenced **action_request_assignment** is assigned.

EXPRESS specification:

```
*)
ENTITY applied_action_request_assignment
  SUBTYPE OF (action_request_assignment);
  items: SET[1:?] OF action_request_item;
END_ENTITY;
(*
```

Attribute definition:

items: the set of **action_request_item** for which a particular **action_request_assignment** is applicable.

Associated global rules:

The following global rule defined in this part of ISO 10303 applies to the **applied_action_request_-assignment** entity:

— change_impact_with_versionable_object_change_event (See 5.2.4.28).

5.2.3.1.3 applied_approval_assignment

An **applied_approval_assignment** specifies those **approval_item**s to which a referenced **approval_assignment** is assigned.

EXPRESS specification:

```
*)
ENTITY applied_approval_assignment
  SUBTYPE OF (approval_assignment);
      items : SET [1:?] OF approval_item;
END_ENTITY;
(*
```

Attribute definition:

items: the set of **approval_item** for which a particular **approval_assignment** is applicable.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **applied_approval_-assignment** entity:

— applied_approval_assignment_has_exactly_one_elements (See 5.2.4.7);

— approval_history_approves_same_definition (See 5.2.4.11).

5.2.3.1.4 applied_classification_assignment

An **applied_classification_assignment** specifies those **classification_item**s to which a referenced **classification_assignment** is assigned.

EXPRESS specification:

```
*)
ENTITY applied_classification_assignment
  SUBTYPE OF (classification_assignment);
  items: SET[1:?] OF classification_item;
END_ENTITY;
(*
```

Attribute definition:

items: the set of **classification_item** for which a particular **classification_assignment** is applicable.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **applied_classification_-assignment** entity:

— action_request_solution_with_identification_assignment (See 5.2.4.2);

— action_with_identification_assignment (see 5.2.4.3);

— alternative_version_relationship_has_mandatory_description (See 5.2.4.4);

— alternative_version_relationship_has_unique_versions (See 5.2.4.5);

— approval_event_with_approval_date_time (See 5.2.4.9);

— approval_event_with_approval_person_organization (See 5.2.4.10);

— approval_history_approves_same_definition (See 5.2.4.11);

— approval_history_has_at_least_one_member (See 5.2.4.12);

— approvals_references_approval_history (See 5.2.4.13);

— centre_location_compound_representation_has_specified_name (see 5.2.4.27);

— change_impact_with_versionable_object_change_event (See 5.2.4.28);

— change_plan_has_mandatory_attribute_description (See 5.2.4.29);

— class_and_statutory_designation_has_properties (See 5.2.4.30);

— class_parameters_has_properties (See 5.2.4.32);

— document_has_at_least_one_references (See 5.2.4.36);

— document_reference_with_address_has_at_least_one_references (See 5.2.4.38);

— envisaged_version_creation_has_mandatory_attribute_description (See 5.2.4.39);

— executed_action_with_identification_assignment (See 5.2.4.40);

— floating_position_compound_representation_with_name (See 5.2.4.42);

— freeboard_characteristics_has_properties (See 5.2.4.43);

— global_axis_placement_has_properties (See 5.2.4.44);

— lightship_definition_has_properties (See 5.2.4.48);

— members_is_referenced_by_at_least_one_revision (See 5.2.4.50);

— no_approvals_except_in_approval_history (See 5.2.4.51);

— principal_characteristics_has_properties (See 5.2.4.52);

— product_definition_for_call_sign (See 5.2.4.53);

— product_definition_for_certifying_organization (See 5.2.4.54);

— product_definition_for_class_notation (See 5.2.4.55);

— product_definition_for_expiry_date (See 5.2.4.56);

— product_definition_for_flag_state (See 5.2.4.57);

— product_definition_for_managing_company (See 5.2.4.59);

— product_definition_for_ordering_company (See 5.2.4.60);

— product_definition_for_owning_company (See 5.2.4.61);

— product_definition_for_port_of_registration (See 5.2.4.62);

— product_definition_for_regulation (See 5.2.4.63);

— product_definition_for_shipyard (See 5.2.4.64);

— product_definition_has_references_with_class_loadline (See 5.2.4.58);

— product_definition_relationship_with_identification_assignment (See 5.2.4.66);

— product_definition_shape_for_deck_zone_design (See 5.2.4.67);

— product_definition_shape_with_identification_assignment (See 5.2.4.68);

— product_definition_with_date_freeboard_assigned (See 5.2.4.69);

— product_definition_with_freeboard_assigned_by (See 5.2.4.70);

— product_definition_with_identification_assignment (See 5.2.4.71);

— product_related_product_category_with_identification_assignment (See 5.2.4.72);

— product_with_identification_assignment (See 5.2.4.73);

— property_definition_for_class_bulk_load_requirement_definition (See 5.2.4.74);

— property_definition_for_class_compartment_requirement_definition (See 5.2.4.75);

— property_definition_for_class_deck_load_requirement_definition (See 5.2.4.76);

— property_definition_for_class_notation (See 5.2.4.77);

— property_definition_for_class_society (See 5.2.4.78);

— property_definition_for_class_tank_requirement_definition (See 5.2.4.79);

— property_definition_for_compartment_design_requirement (See 5.2.4.80);

— property_definition_for_compartment_function (See 5.2.4.81);

— property_definition_for_compensated_gross_tonnage (See 5.2.4.82);

— property_definition_for_damage_stability_definition_requires_reference (See 5.2.4.83);

— property_definition_for_date_of_loading (See 5.2.4.84);

— property_definition_for_deck_zone_function (See 5.2.4.85);

— property_definition_for_gross_tonnage (See 5.2.4.86);

— property_definition_for_local_coordinate_system (See 5.2.4.87);

— property_definition_for_local_coordinate_system_with_position (See 5.2.4.88);

— property_definition_for_net_tonnage (See 5.2.4.89);

— property_definition_for_stability_definition_requires_reference (See 5.2.4.90);

— property_definition_for_tonnage_definition (See 5.2.4.91);

— property_definition_for_zone_function (See 5.2.4.92);

— property_definition_has_references_with_name_loadline (See 5.2.4.93);

— property_definition_representation_for_date_of_measurement (See 5.2.4.94);

— property_definition_representation_for_gross_tonnage (See 5.2.4.95);

— property_definition_representation_for_net_tonnage (See 5.2.4.96);

— property_definition_with_identification_assignment (See 5.2.4.97);

— property_definition_with_lightship_weight_item (See 5.2.4.98);

— property_definition_with_weight_and_centre_of_gravity (See 5.2.4.99);

— representation_for_stability_table_restricted (See 5.2.4.171);

— representation_for_stability_table_restricted_by_class_id (See 5.2.4.172);

— representation_item_for_transformation_to_parent (see 5.2.4.184).

— revision_has_mandatory_attribute_description (See 5.2.4.216);

— revision_with_context_referenced_for_context_of_revision (See 5.2.4.217);

— ship_designation_has_one_specified_names (See 5.2.4.219);

— spacing_position_compound_representation_has_name (See 5.2.4.220);

— spacing_position_with_offset_compound_representation_has_class (See 5.2.4.221);

— spacing_position_with_offset_compound_representation_has_name (See 5.2.4.222);

— stability_properties_for_floating_position_has_class (See 5.2.4.223);

— stability_properties_for_floating_position_has_name (See 5.2.4.224);

— stability_property_has_name (See 5.2.4.225);

— tonnage_definition_has_properties (See 5.2.4.226);

— unique_approvals_in_approval_history (See 5.2.4.227);

— version_creation_has_mandatory_attribute_description (See 5.2.4.241);

— version_deletion_has_mandatory_attribute_description (See 5.2.4.242);

— version_history_has_exactly_one_assigned_group (See 5.2.4.243);

— version_history_is_referenced_by_at_least_one_versions (See 5.2.4.244);

— version_history_referenced_by_exactly_one_current_version (See 5.2.4.245);

— version_history_referenced_by_multiple_roles (See 5.2.4.246);

— version_modification_has_mandatory_attribute_description (See 5.2.4.247);

— version_relationship_associates_with_versionable_object (See 5.2.4.248);

— version_relationship_has_mandatory_attribute_description (See 5.2.4.249);

— version_relationship_has_unique_versions (See 5.2.4.250);

— versioned_action_request_with_identification_assignment (See 5.2.4.252);

— versions_is_referenced_by_at_least_one_version_history (See 5.2.4.253).

5.2.3.1.5 applied_date_and_time_assignment

An **applied_date_and_time_assignment** specifies those **date_and_time_item**s to which a referenced **date_and_assignment** is assigned.

EXPRESS specification:

```
*)
ENTITY applied_date_and_time_assignment
  SUBTYPE OF (date_and_time_assignment);
  items: SET[1:?] OF date_and_time_item;
END_ENTITY;
```

(*

Attribute definition:

items: the set of **date_and_time_item** for which a particular **date_and_time_assignment** is applicable.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **applied_date_and_time_-assignment** entity:

— caused_when_for_check (See 5.2.4.22);

— caused_when_for_envisaged_version_creation (See 5.2.4.23);

— caused_when_for_version_creation (See 5.2.4.24);

— caused_when_for_version_deletion (See 5.2.4.25);

— caused_when_for_version_modification (See 5.2.4.26);

— date_time_for_change_plan (See 5.2.4.33);

— date_time_for_change_realization (See 5.2.4.34);

— date_time_for_change_request (See 5.2.4.35);

— product_definition_with_date_freeboard_assigned (See 5.2.4.69);

— product_definition_for_expiry_date (See 5.2.4.56);

— property_definition_for_date_of_loading (See 5.2.4.84);

— property_definition_representation_for_gross_tonnage (See 5.2.4.95);

— property_definition_representation_for_net_tonnage (See 5.2.4.96);

— property_definition_representation_for_date_of_measurement (See 5.2.4.94).

5.2.3.1.6 applied_document_reference

An **applied_document_reference** specifies those **document_reference_item**s to which a referenced **document_reference** is assigned.

EXPRESS specification:

```
*)
ENTITY applied_document_reference
  SUBTYPE OF (document_reference);
  items: SET[1:?] OF document_reference_item;
END_ENTITY;
(*
```

Attribute definition:

items: the set of **document_reference_item** for which a particular **document_reference** is applicable.

5.2.3.1.7 applied_effectivity_assignment

An **applied_effectivity_assignment** specifies those **effectivity_item** to which a referenced **effectivity_assignment** is assigned.

EXPRESS specification:

```
*)
ENTITY applied_effectivity_assignment
SUBTYPE OF (effectivity_assignment);
items : SET [1:?] OF effectivity_item;
END_ENTITY; -- applied_effectivity_assignment
(*
```

Attribute definition:

items: the set of **effectivity_item** for which a particular **effectivity_assignment** is applicable.

5.2.3.1.8 applied_external_identification_assignment

An **applied_external_identification_assignment** specifies those **external_identification_item**s to which a referenced **external_identification_assignment** is assigned.

EXPRESS specification:

```
*)
ENTITY applied_external_identification_assignment
  SUBTYPE OF (external_identification_assignment);
  items: SET[1:?] OF external_identification_item;
END_ENTITY;
(*
```

Attribute definition:

items: the set of **external_identification_item** for which a particular **external_identification_-assignment** is applicable.

Associated global rules:

The following global rule defined in this part of ISO 10303 applies to the **applied_external_-identification_assignment** entity:

— document_reference_with_address_has_at_least_one_references (See 5.2.4.38).

5.2.3.1.9 applied_group_assignment

An **applied_group_assignment** specifies those **group_item**s to which a referenced **group_-assignment** is assigned.

EXPRESS specification:

```
*)
ENTITY applied_group_assignment
  SUBTYPE OF (group_assignment);
  items: SET[1:?] OF group_item;
END_ENTITY;
(*
```

Attribute definition:

items: the set of **group_item** for which a particular **group_assignment** is applicable.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **applied_group_-assignment** entity:

— applied_group_assignment_has_at_least_one_elements (See 5.2.4.8);

— approval_history_approves_same_definition (See 5.2.4.11);

— approval_history_has_at_least_one_member (See 5.2.4.12);

— approvals_references_approval_history (See 5.2.4.13);

— members_is_referenced_by_at_least_one_revision (See 5.2.4.50);

— no_approvals_except_in_approval_history (See 5.2.4.51);

— revision_with_context_referenced_for_context_of_revision (See 5.2.4.217);

— unique_approvals_in_approval_history (See 5.2.4.227);

— version_history_has_exactly_one_assigned_group (See 5.2.4.243);

— version_history_is_referenced_by_at_least_one_versions (See 5.2.4.244);

— version_history_referenced_by_exactly_one_current_version (See 5.2.4.245);

— version_history_referenced_by_multiple_roles (See 5.2.4.246);

— versions_is_referenced_by_at_least_one_version_history (See 5.2.4.253).

5.2.3.1.10 applied_identification_assignment

An **applied_identification_assignment** specifies those **identification_item**s to which a referenced **identification_assignment** is assigned.

EXPRESS specification:

```
*)
ENTITY applied_identification_assignment
  SUBTYPE OF (identification_assignment);
  items: SET[1:?] OF identification_item;
END_ENTITY;
(*
```


Attribute definition:

items: the set of **identification_item** for which a particular **identification_assignment** is applicable.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **applied_identification_-assignment** entity:

— alternative_version_relationship_versionable_object (See 5.2.4.6);

— global_id_is_unique (See 5.2.4.45);

— product_definition_for_call_sign (See 5.2.4.53);

— product_definition_for_flag_state (See 5.2.4.57);

— product_definition_for_port_of_registration (See 5.2.4.62);

— ship_designation_has_one_specified_names (See 5.2.4.219);

— version_relationship_associates_with_versionable_object (See 5.2.4.248);

— versionable_object_has_one_version_id (See 5.2.4.251).

5.2.3.1.11 applied_organization_assignment

An **applied_organization_assignment** specifies those **organization_item**s to which a referenced **organization_assignment** is assigned.

EXPRESS specification:

```
*)
ENTITY applied_organization_assignment
  SUBTYPE OF (organization_assignment);
  items: SET[1:?] OF organization_item;
END_ENTITY;
(*
```

Attribute definition:

items: the set of **organization_item** for which a particular **organization_assignment** is applicable.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **applied_organization_-assignment** entity:

— document_has_exactly_one_author (See 5.2.4.37);

— product_definition_for_certifying_organization (See 5.2.4.54);

— product_definition_with_freeboard_assigned_by (See 5.2.4.70);

— product_definition_for_managing_company (See 5.2.4.59);

— product_definition_for_ordering_company (See 5.2.4.60);

— product_definition_for_owning_company (See 5.2.4.61);

— product_definition_for_shipyard (See 5.2.4.64);

— property_definition_for_class_society (See 5.2.4.78).

5.2.3.1.12 applied_person_and_organization_assignment

An **applied_person_and_organization_assignment** specifies those **person_and_organization_-item**s to which a referenced **person_and_organization_assignment** is assigned.

EXPRESS specification:

```
*)
ENTITY applied_person_and_organization_assignment
  SUBTYPE OF (person_and_organization_assignment);
  items: SET[1:?] OF person_and_organization_item;
END_ENTITY;
(*
```

Attribute definition:

items: the set of **person_and_organization_item** for which a particular **person_and_-organization_assignment** is applicable.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **applied_person_-and_organization_assignment** entity:

— author_for_change_plan (See 5.2.4.14);

— author_for_change_realization (See 5.2.4.15);

— author_for_change_request (See 5.2.4.16);

— caused_by_for_check (See 5.2.4.17);

— caused_by_for_envisaged_version_creation (See 5.2.4.18);

— caused_by_for_version_creation (See 5.2.4.19);

— caused_by_for_version_deletion (See 5.2.4.20);

— caused_by_for_version_modification (See 5.2.4.21);

— document_has_exactly_one_author (See 5.2.4.37);

— initiator_for_change_request (See 5.2.4.47).

5.2.3.1.13 applied_person_assignment

An **applied_person_assignment** specifies those **person_item**s to which a referenced **person_-assignment** is assigned.

EXPRESS specification:

```
*)
ENTITY applied_person_assignment
  SUBTYPE OF (person_assignment);
  items: SET[1:?] OF person_item;
END_ENTITY;
(*
```

Attribute definition:

items: the set of **person_item** for which a particular **person_assignment** is applicable.

Associated global rules:

The following global rule defined in this part of ISO 10303 applies to the **applied_person_-assignment** entity:

— document_has_exactly_one_author (See 5.2.4.37).

5.2.3.1.14 class

A **class** is a type of **group** that specifies a type of classification assignment.

EXPRESS specification:

```
*)
  ENTITY class
    SUBTYPE OF (group);
    WHERE
     WR1:   (SIZEOF(QUERY ( oa <* USEDIN(SELF,
            'SHIP_ARRANGEMENT_SCHEMA.GROUP_ASSIGNMENT.ASSIGNED_GROUP')|
         NOT ('SHIP_ARRANGEMENT_SCHEMA.APPLIED_GROUP_ASSIGNMENT' IN
          TYPEOF(oa))
             )) =0);
  END_ENTITY; -- class
(*
```

Formal proposition:

WR1: Each **class** shall not be referenced by any **applied_group_assignment** through the **assigned_group** attribute.

5.2.3.2 Ship arrangement imported entity modification

5.2.3.2.1 action

The base definition of the **action** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **action** entity:

— action_request_solution_connected_to_action (see 5.2.4.1);

— action_with_identification_assignment (see 5.2.4.3);

— caused_by_for_check (see 5.2.4.17);

— caused_by_for_envisaged_version_creation (see 5.2.4.18);

— caused_by_for_version_creation (see 5.2.4.19);

— caused_by_for_version_deletion (see 5.2.4.20);

— caused_by_for_version_modification (see 5.2.4.21);

— caused_when_for_check (see 5.2.4.22);

— caused_when_for_envisaged_version_creation (see 5.2.4.23);

— caused_when_for_version_creation (see 5.2.4.24);

— caused_when_for_version_deletion (see 5.2.4.25);

— caused_when_for_version_modification (see 5.2.4.26);

— change_impact_with_versionable_object_change_event (see 5.2.4.28);

— envisaged_version_creation_has_mandatory_attribute_description (see 5.2.4.39);

— version_creation_has_mandatory_attribute_description (see 5.2.4.241);

— version_deletion_has_mandatory_attribute_description (see 5.2.4.242);

— version_modification_has_mandatory_attribute_description (see 5.2.4.247).

5.2.3.2.2 action_request_solution

The base definition of the **action_request_solution** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **action_request_solution** entity:

— action_request_solution_connected_to_action (see 5.2.4.1);

— action_request_solution_with_identification_assignment (see 5.2.4.2);

— author_for_change_plan (see 5.2.4.14);

— change_plan_has_mandatory_attribute_description (see 5.2.4.29);

— date_time_for_change_plan (see 5.2.4.33).

5.2.3.2.3 approval

The base definition of the **approval** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **approval** entity:

— approval_event_with_approval_date_time (see 5.2.4.9);

— approval_event_with_approval_person_organization (see 5.2.4.10);

— no_approvals_except_in_approval_history (see 5.2.4.51).

5.2.3.2.4 approval_date_time

The base definition of the **approval_date_time** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rule defined in this part of ISO 10303 applies to the **approval_date_time** entity:

— approval_event_with_approval_date_time (see 5.2.4.9).

5.2.3.2.5 approval_person_organization

The base definition of the **approval_person_organization** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rule defined in this part of ISO 10303 applies to the **approval_person_organization** entity:

— approval_event_with_approval_person_organization (see 5.2.4.10).

5.2.3.2.6 compound_representation_item

The base definition of the **compound_representation_item** entity is given in ISO 10303-43. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **compound_representation_item** entity:

— centre_location_compound_representation_has_specified_name (see 5.2.4.27);

— spacing_position_compound_representation_has_name (see 5.2.4.220);

— spacing_position_with_offset_compound_representation_has_class (see 5.2.4.221);

— spacing_position_with_offset_compound_representation_has_name (see 5.2.4.222).

5.2.3.2.7 date_time_role

The base definition of the **date_time_role** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **date_time_role** entity:

— caused_when_for_check (see 5.2.4.22);

— caused_when_for_envisaged_version_creation (see 5.2.4.23);

— caused_when_for_version_creation (see 5.2.4.24);

— caused_when_for_version_deletion (see 5.2.4.25);

— caused_when_for_version_modification (see 5.2.4.26);

— date_time_for_change_plan (see 5.2.4.33);

— date_time_for_change_realization (see 5.2.4.34);

— date_time_for_change_request (see 5.2.4.35).

5.2.3.2.8 document

The base definition of the **document** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **document** entity:

— document_has_at_least_one_references (see 5.2.4.36);

— document_has_exactly_one_author (see 5.2.4.37);

— document_reference_with_address_has_at_least_one_references (see 5.2.4.38).

5.2.3.2.9 document_representation_type

The base definition of the **document_representation_type** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rule defined in this part of ISO 10303 applies to the **document_representation_type** entity:

— document_has_at_least_one_references (see 5.2.4.36).

5.2.3.2.10 executed_action

The base definition of the **executed_action** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **executed_action** entity:

— author_for_change_realization (see 5.2.4.15);

— date_time_for_change_realization (see 5.2.4.34);

— executed_action_with_identification_assignment (See 5.2.4.40).

5.2.3.2.11 external_source

The base definition of the **external_source** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rule defined in this part of ISO 10303 applies to the **external_source** entity:

— mandatory_entity_type_for_external_instance_reference (see 5.2.4.49).

5.2.3.2.12 external_source_relationship

The base definition of the **external_source_relationship** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rule defined in this part of ISO 10303 applies to the **external_source_relationship** entity:

— mandatory_entity_type_for_external_instance_reference (see 5.2.4.49).

5.2.3.2.13 group

The base definition of the **group** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **group** entity:

— approval_history_has_at_least_one_member (see 5.2.4.12);

— approvals_references_approval_history (see 5.2.4.13);

— centre_location_compound_representation_has_specified_name (see 5.2.4.27);

— class_and_statutory_designation_has_properties (see 5.2.4.30);

— class_parameters_has_properties (see 5.2.4.32);

— floating_position_compound_representation_with_name (see 5.2.4.42);

— freeboard_characteristics_has_properties (see 5.2.4.43);

— global_axis_placement_has_properties (see 5.2.4.44);

— lightship_definition_has_properties (see 5.2.4.48);

— members_is_referenced_by_at_least_one_revision (see 5.2.4.50);

— principal_characteristics_has_properties (see 5.2.4.52);

— product_definition_for_shipyard (see 5.2.4.64);

— product_definition_for_call_sign (see 5.2.4.53);

— product_definition_for_certifying_organization (see 5.2.4.54);

— product_definition_for_class_notation (see 5.2.4.55);

— product_definition_for_expiry_date (see 5.2.4.56);

— product_definition_for_flag_state (see 5.2.4.57);

— product_definition_for_managing_company (see 5.2.4.59);

— product_definition_for_ordering_company (see 5.2.4.60);

— product_definition_for_owning_company (see 5.2.4.61);

— product_definition_for_port_of_registration (see 5.2.4.62);

— product_definition_for_regulation (see 5.2.4.63);

— product_definition_for_shipyard (see 5.2.4.64);

— product_definition_has_references_with_class_loadline (see 5.2.4.58);

— product_definition_shape_for_deck_zone_design (see 5.2.4.67);

— product_definition_with_date_freeboard_assigned (see 5.2.4.69);

— product_definition_with_freeboard_assigned_by (see 5.2.4.70);

— property_definition_for_class_bulk_load_requirement_definition (see 5.2.4.74);

— property_definition_for_class_compartment_requirement_definition (see 5.2.4.75);

— property_definition_for_class_deck_load_requirement_definition (see 5.2.4.76);

— property_definition_for_class_notation (see 5.2.4.77);

— property_definition_for_class_society (see 5.2.4.78);

— property_definition_for_class_tank_requirement_definition (see 5.2.4.79);

— property_definition_for_compartment_design_requirement (see 5.2.4.80);

— property_definition_for_compartment_function (see 5.2.4.81);

— property_definition_for_compensated_gross_tonnage (see 5.2.4.82);

— property_definition_for_damage_stability_definition_requires_reference (see 5.2.4.83);

— property_definition_for_date_of_loading (see 5.2.4.84);

— property_definition_for_deck_zone_function (see 5.2.4.85);

— property_definition_for_gross_tonnage (see 5.2.4.86);

— property_definition_for_local_coordinate_system (see 5.2.4.87);

— property_definition_for_local_coordinate_system_with_position (see 5.2.4.88);

— property_definition_for_net_tonnage (see 5.2.4.89);

— property_definition_for_stability_definition_requires_reference (see 5.2.4.90);

— property_definition_for_tonnage_definition (see 5.2.4.91);

— representation_item_for_transformation_to_parent (see 5.2.4.184);

— property_definition_for_zone_function (see 5.2.4.92);

— property_definition_has_references_with_name_loadline (see 5.2.4.93);

— property_definition_representation_for_date_of_measurement (see 5.2.4.94);

— property_definition_representation_for_gross_tonnage (see 5.2.4.95);

— property_definition_representation_for_net_tonnage (see 5.2.4.96);

— property_definition_with_lightship_weight_item (see 5.2.4.98);

— property_definition_with_weight_and_centre_of_gravity (see 5.2.4.99);

— representation_for_stability_table_restricted (see 5.2.4.171);

— representation_for_stability_table_restricted_by_class_id (see 5.2.4.172);

— representation_item_for_transformation_to_parent (see 5.2.4.184).

— revision_has_mandatory_attribute_description (see 5.2.4.216);

— revision_with_context_referenced_for_context_of_revision (see 5.2.4.217);

— ship_designation_has_one_specified_names (see 5.2.4.219);

— spacing_position_compound_representation_has_name (see 5.2.4.220);

— spacing_position_with_offset_compound_representation_has_class (see 5.2.4.221);

— spacing_position_with_offset_compound_representation_has_name (see 5.2.4.222);

— stability_properties_for_floating_position_has_class (see 5.2.4.223);

— stability_properties_for_floating_position_has_name (see 5.2.4.224);

— stability_property_has_name (see 5.2.4.225);

— unique_approvals_in_approval_history (see 5.2.4.227);

— version_history_has_exactly_one_assigned_group (see 5.2.4.243);

— version_history_is_referenced_by_at_least_one_versions (see 5.2.4.244);

— version_history_referenced_by_exactly_one_current_version (see 5.2.4.245);

— version_history_referenced_by_multiple_roles (see 5.2.4.246);

— versions_is_referenced_by_at_least_one_version_history (see 5.2.4.253).

5.2.3.2.14 identification_assignment_relationship

The base definition of the **identification_assignment_relationship** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **identification_assignment_relationship** entity:

— alternative_version_relationship_has_mandatory_description (see 5.2.4.4);

— alternative_version_relationship_has_unique_versions (see 5.2.4.5);

— alternative_version_relationship_versionable_object (see 5.2.4.6);

— version_relationship_has_mandatory_attribute_description (see 5.2.4.249);

— version_relationship_has_unique_versions (see 5.2.4.250).

5.2.3.2.15 identification_role

The base definition of the **identification_role** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **identification_role** entity:

— identification_role_optional_attribute_description_required (see 5.2.4.46).

5.2.3.2.16 mapped_item

The base definition of the **mapped_item** entity is given in ISO 10303-43. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rule defined in this part of ISO 10303 applies to the **mapped_item** entity:

— representation_item_for_transformation_to_parent (see 5.2.4.184).

5.2.3.2.17 object_role

The base definition of the **object_role** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **object_role** entity:

— applied_approval_assignment_has_exactly_one_elements (see 5.2.4.7);

— applied_group_assignment_has_at_least_one_elements (see 5.2.4.8);

— approvals_references_approval_history (see 5.2.4.13);

— change_impact_with_versionable_object_change_event (see 5.2.4.28).

5.2.3.2.18 person_and_organization_role

The base definition of the **person_and_organization_role** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **person_and_-organization_role** entity:

— author_for_change_plan (see 5.2.4.14);

— author_for_change_realization (see 5.2.4.15);

— author_for_change_request (see 5.2.4.16);

— caused_by_for_check (see 5.2.4.17);

— caused_by_for_envisaged_version_creation (see 5.2.4.18);

— caused_by_for_version_creation (see 5.2.4.19);

— caused_by_for_version_deletion (see 5.2.4.20);

— caused_by_for_version_modification (see 5.2.4.21);

— initiator_for_change_request (see 5.2.4.47).

5.2.3.2.19 product

The base definition of the **product** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **product** entity:

— product_with_identification_assignment (See 5.2.4.73).

5.2.3.2.20 product_definition

The base definition of the **product_definition** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **product_definition** entity:

— class_and_statutory_designation_has_properties (see 5.2.4.30);

— class_parameters_has_properties (see 5.2.4.32);

— freeboard_characteristics_has_properties (see 5.2.4.43);

— global_axis_placement_has_properties (see 5.2.4.44);

— lightship_definition_has_properties (see 5.2.4.48);

— principal_characteristics_has_properties (see 5.2.4.52);

— product_definition_for_call_sign (see 5.2.4.53);

— product_definition_for_certifying_organization (see 5.2.4.54);

— product_definition_for_class_notation (see 5.2.4.55);

— product_definition_for_expiry_date (see 5.2.4.56);

— product_definition_for_flag_state (see 5.2.4.57);

— product_definition_for_managing_company (see 5.2.4.59);

— product_definition_for_ordering_company (see 5.2.4.60);

— product_definition_for_owning_company (see 5.2.4.61);

— product_definition_for_port_of_registration (see 5.2.4.62);

— product_definition_for_regulation (see 5.2.4.63);

— product_definition_for_shipyard (see 5.2.4.64);

— product_definition_has_references_with_class_loadline (see 5.2.4.58);

— product_definition_with_date_freeboard_assigned (see 5.2.4.69);

— product_definition_with_freeboard_assigned_by (see 5.2.4.70);

— product_definition_with_identification_assignment (See 5.2.4.71);

— ship_designation_has_one_specified_names (see 5.2.4.219);

— tonnage_definition_has_properties (see 5.2.4.226).

5.2.3.2.21 product_definition_relationship

The base definition of the **product_definition_relationship** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **product_definition_-relationship** entity:

— product_definition_relationship_with_identification_assignment (See 5.2.4.66).

5.2.3.2.22 product_definition_shape

The base definition of the **product_definition_shape** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **product_definition_shape** entity:

— product_definition_shape_for_deck_zone_design (see 5.2.4.67);

— product_definition_shape_with_identification_assignment (See 5.2.4.68).

5.2.3.2.23 product_related_product_category

The base definition of the **product_related_product_category** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **product_related_-product_category** entity:

— product_related_product_category_with_identification_assignment (See 5.2.4.72).

5.2.3.2.24 property_definition

The base definition of the **property_definition** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **property_definition** entity:

— class_and_statutory_designation_has_properties (see 5.2.4.30);

— class_parameters_has_properties (see 5.2.4.32);

— freeboard_characteristics_has_properties (see 5.2.4.43);

— global_axis_placement_has_properties (see 5.2.4.44);

— lightship_definition_has_properties (see 5.2.4.48);

— principal_characteristics_has_properties (see 5.2.4.52);

— product_definition_for_class_notation (see 5.2.4.55);

— product_definition_for_regulation (see 5.2.4.63);

— product_definition_has_references_with_class_loadline (see 5.2.4.58);

— property_definition_for_class_bulk_load_requirement_definition (see 5.2.4.74);

— property_definition_for_class_compartment_requirement_definition (see 5.2.4.75);

— property_definition_for_class_deck_load_requirement_definition (see 5.2.4.76);

— property_definition_for_class_notation (see 5.2.4.77);

— property_definition_for_class_society (see 5.2.4.78);

— property_definition_for_class_tank_requirement_definition (see 5.2.4.79);

— property_definition_for_compartment_design_requirement (see 5.2.4.80);

— property_definition_for_compartment_function (see 5.2.4.81);

— property_definition_for_compensated_gross_tonnage (see 5.2.4.82);

— property_definition_for_damage_stability_definition_requires_reference (see 5.2.4.83);

— property_definition_for_date_of_loading (see 5.2.4.84);

— property_definition_for_deck_zone_function (see 5.2.4.85);

— property_definition_for_gross_tonnage (see 5.2.4.86);

— property_definition_for_local_coordinate_system (see 5.2.4.87);

— property_definition_for_local_coordinate_system_with_position (see 5.2.4.88);

— property_definition_for_net_tonnage (see 5.2.4.89);

— property_definition_for_stability_definition_requires_reference (see 5.2.4.90);

— property_definition_for_tonnage_definition (see 5.2.4.91);

— property_definition_for_zone_function (see 5.2.4.92);

— property_definition_has_references_with_name_loadline (see 5.2.4.93);

— property_definition_with_identification_assignment (See 5.2.4.97);

— property_definition_with_lightship_weight_item (see 5.2.4.98);

— property_definition_with_weight_and_centre_of_gravity (see 5.2.4.99);

— representation_item_for_transformation_to_parent (see 5.2.4.184);

— tonnage_definition_has_properties (see 5.2.4.226).

5.2.3.2.25 property_definition_representation

The base definition of the **property_definition_representation** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **property_definition_-representation** entity:

— class_and_statutory_designation_has_properties (see 5.2.4.30);

— class_parameters_has_properties (see 5.2.4.32);

— freeboard_characteristics_has_properties (see 5.2.4.43);

— global_axis_placement_has_properties (see 5.2.4.44);

— lightship_definition_has_properties (see 5.2.4.48);

— principal_characteristics_has_properties (see 5.2.4.52);

— product_definition_shape_for_deck_zone_design (see 5.2.4.67);

— property_definition_for_class_bulk_load_requirement_definition (see 5.2.4.74);

— property_definition_for_class_compartment_requirement_definition (see 5.2.4.75);

— property_definition_for_class_deck_load_requirement_definition (see 5.2.4.76);

— property_definition_for_class_notation (see 5.2.4.77);

— property_definition_for_class_tank_requirement_definition (see 5.2.4.79);

— property_definition_for_compartment_design_requirement (see 5.2.4.80);

— property_definition_for_compartment_function (see 5.2.4.81);

— property_definition_for_compensated_gross_tonnage (see 5.2.4.82);

— property_definition_for_deck_zone_function (see 5.2.4.85);

— property_definition_for_gross_tonnage (see 5.2.4.86);

— property_definition_for_local_coordinate_system (see 5.2.4.87);

— property_definition_for_local_coordinate_system_with_position (see 5.2.4.88);

— property_definition_for_net_tonnage (see 5.2.4.89);

— property_definition_for_tonnage_definition (see 5.2.4.91);

— property_definition_for_zone_function (see 5.2.4.92);

— property_definition_has_references_with_name_loadline (see 5.2.4.93);

— property_definition_representation_for_date_of_measurement (see 5.2.4.94);

— property_definition_representation_for_gross_tonnage (see 5.2.4.95);

— property_definition_representation_for_net_tonnage (see 5.2.4.96);

— property_definition_with_lightship_weight_item (see 5.2.4.98);

— property_definition_with_weight_and_centre_of_gravity (see 5.2.4.99);

— tonnage_definition_has_properties (see 5.2.4.226).

5.2.3.2.26 representation

The base definition of the **representation** entity is given in ISO 10303-43. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **representation** entity:

— class_notation_with_named_representation_items (see 5.2.4.31);

— representation_for_absolute_cargo (see 5.2.4.100);

— representation_for_adjacent_space_surface_area (see 5.2.4.101);

— representation_for_bulk_cargo (see 5.2.4.102);

— representation_for_bulk_cargo_assignment (see 5.2.4.103);

— representation_for_capacity_properties (see 5.2.4.104);

— representation_for_cargo_compartment_property (see 5.2.4.105);

— representation_for_cargo_footprint (see 5.2.4.106);

— representation_for_class_and_statutory_designation (see 5.2.4.107);

— representation_for_class_bulk_load_requirement_definition (see 5.2.4.108);

— representation_for_class_compartment_requirement_definition (see 5.2.4.109);

— representation_for_class_notation (see 5.2.4.110);

— representation_for_class_parameters (see 5.2.4.111);

— representation_for_class_tank_requirement_definition (see 5.2.4.112);

— representation_for_coating (see 5.2.4.113);

— representation_for_coating_level (see 5.2.4.114);

— representation_for_compartment_abbreviated_name (see 5.2.4.115);

— representation_for_compartment_acceleration (see 5.2.4.116);

— representation_for_compartment_access_authorization (see 5.2.4.117);

— representation_for_compartment_air_circulation_rate (see 5.2.4.118);

— representation_for_compartment_cargo_assignment (see 5.2.4.119);

— representation_for_compartment_coating (see 5.2.4.120);

— representation_for_compartment_design_requirement (see 5.2.4.121);

— representation_for_compartment_function (see 5.2.4.122);

— representation_for_compartment_group (see 5.2.4.123);

— representation_for_compartment_horizontal_cross_sectional_area (see 5.2.4.124);

— representation_for_compartment_illumination (see 5.2.4.125);

— representation_for_compartment_insulation (see 5.2.4.126);

— representation_for_compartment_noise_category (see 5.2.4.127);

— representation_for_compartment_nuclear_classification (see 5.2.4.128);

— representation_for_compartment_occupancy (see 5.2.4.129);

— representation_for_compartment_safety_class (see 5.2.4.130);

— representation_for_compartment_security_classification (see 5.2.4.131);

— representation_for_compartment_stiffened_surface_area_property (see 5.2.4.132);

— representation_for_compartment_tightness (see 5.2.4.133);

— representation_for_compartment_unstiffened_surface_area_property (see 5.2.4.134);

— representation_for_compartment_vertical_longitudinal_sectional_area (see 5.2.4.135);

— representation_for_compartment_vertical_transverse_sectional_area (see 5.2.4.136);

— representation_for_compartment_volume_permeability_property (see 5.2.4.137);

— representation_for_compartment_volume_property (see 5.2.4.138);

— representation_for_compartment_ziplist_number (see 5.2.4.139);

— representation_for_compensated_gross_tonnage (see 5.2.4.140);

— representation_for_corrosion_control_coating (see 5.2.4.141);

— representation_for_corrosion_protection (see 5.2.4.142);

— representation_for_damage_case (see 5.2.4.143);

— representation_for_damage_position (see 5.2.4.144);

— representation_for_dangerous_goods_code (see 5.2.4.145);

— representation_for_deck_cargo_assignment (see 5.2.4.146);

— representation_for_deck_zone_function (see 5.2.4.147);

— representation_for_dry_cargo (see 5.2.4.148);

— representation_for_fire_safe_coating (see 5.2.4.149);

— representation_for_freeboard_characteristics (see 5.2.4.150);

— representation_for_gaseous_cargo (see 5.2.4.151);

— representation_for_gaseous_cargo_assignment (see 5.2.4.152);

— representation_for_global_axis_placement (see 5.2.4.153);

— representation_for_gross_tonnage (see 5.2.4.154);

— representation_for_lightship_definition (see 5.2.4.155);

— representation_for_lightship_weight_item (see 5.2.4.156);

— representation_for_liquid_cargo (see 5.2.4.157);

— representation_for_liquid_cargo_assignment (see 5.2.4.158);

— representation_for_loading_condition_design_definition (see 5.2.4.159);

— representation_for_loading_condition_operating_definition (see 5.2.4.160);

— representation_for_loadline (see 5.2.4.161);

— representation_for_local_co_ordinate_system (see 5.2.4.162);

— representation_for_moment_3d_restricts_representation_item (see 5.2.4.163);

— representation_for_moments_of_inertia (see 5.2.4.164);

— representation_for_net_tonnage (see 5.2.4.165);

— representation_for_person_group (see 5.2.4.166);

— representation_for_primer_coating (see 5.2.4.167);

— representation_for_principal_characteristics (see 5.2.4.168);

— representation_for_space_adjacency_relationship (see 5.2.4.169);

— representation_for_space_positional_relationship (see 5.2.4.170);

— representation_for_stability_table_restricted (see 5.2.4.171);

— representation_for_stability_table_restricted_by_class_id (see 5.2.4.172);

— representation_for_tank_compartment_property (see 5.2.4.173);

— representation_for_tonnage_definition (see 5.2.4.174);

— representation_for_tonnage_measurement (see 5.2.4.175);

— representation_for_unit_cargo (see 5.2.4.176);

— representation_for_unit_cargo_assignment (see 5.2.4.177);

— representation_for_unit_cargo_bounding_box (see 5.2.4.178);

— representation_for_unit_cargo_group (see 5.2.4.179);

— representation_for_vehicle_load_description (see 5.2.4.180);

— representation_for_zone_function (see 5.2.4.181);

— representation_has_global_uncertainty_assigned_context (see 5.2.4.182);

— representation_has_global_unit_assigned_context (see 5.2.4.183);

— representation_item_for_transformation_to_parent (see 5.2.4.184);

— representation_items_optional_for_bulk_cargo (see 5.2.4.185);

— representation_items_optional_for_capacity_properties (see 5.2.4.186);

— representation_items_optional_for_class_deck_load_requirement_definition (see 5.2.4.187);

— representation_items_optional_for_class_notation (see 5.2.4.188);

— representation_items_optional_for_compartment_access_authorization (see 5.2.4.189);

— representation_items_optional_for_compartment_design_requirement (see 5.2.4.190);

— representation_items_optional_for_compartment_function (see 5.2.4.191);

— representation_items_optional_for_compartment_insulation (see 5.2.4.192);

— representation_items_optional_for_compartment_noise_category (see 5.2.4.193);

— representation_items_optional_for_compartment_safety_class (see 5.2.4.194);

— representation_items_optional_for_compartment_security (see 5.2.4.195);

— representation_items_optional_for_compartment_tightness (see 5.2.4.196);

— representation_items_optional_for_corrosion_control_coating (see 5.2.4.197);

— representation_items_optional_for_damage_case (see 5.2.4.198);

— representation_items_optional_for_deck_zone_function (see 5.2.4.199);

— representation_items_optional_for_detailed_cargo_material_properties (see 5.2.4.200);

— representation_items_optional_for_dry_cargo (see 5.2.4.201);

— representation_items_optional_for_gaseous_cargo (see 5.2.4.202);

— representation_items_optional_for_general_cargo_material_properties (see 5.2.4.203);

— representation_items_optional_for_liquid_cargo (see 5.2.4.204);

— representation_items_optional_for_loading_condition_operating_definition (see 5.2.4.205);

— representation_items_optional_for_owner_designation (see 5.2.4.206);

— representation_items_optional_for_principal_characteristics (see 5.2.4.207);

— representation_items_optional_for_space_connection_relationship (see 5.2.4.208);

— representation_items_optional_for_tank_geometric_parameters (see 5.2.4.209);

— representation_items_optional_for_tank_piping_design_properties (see 5.2.4.210);

— representation_items_optional_for_unit_cargo (see 5.2.4.211);

— representation_items_optional_for_vehicle_load_description (see 5.2.4.212);

— representation_items_optional_for_zone_function (see 5.2.4.213);

— representation_local_coordinate_system_with_position_reference (see 5.2.4.214);

— representation_restricted_weight_and_centre_of_gravity (see 5.2.4.215);

— user_def_cargo_description_required_for_cargo_type (see 5.2.4.228);

— user_def_cargo_description_required_for_type_of (see 5.2.4.229);

— user_def_function_description_required (see 5.2.4.230);

— user_defined_capacity_context_description_required_for_capacity_context (see 5.2.4.231);

— user_defined_description_required_for_damage_cause (see 5.2.4.232);

— user_defined_type_description_required_for_type_of (see 5.2.4.233);

— user_defined_value_description_required_for_authorization_classification (see 5.2.4.234);

— user_defined_value_description_required_for_compartment_insulation (see 5.2.4.235);

— user_defined_value_description_required_for_compartment_noise_category (see 5.2.4.236);

— user_defined_value_description_required_for_compartment_safety_class (see 5.2.4.237);

— user_defined_value_description_required_for_compartment_security (see 5.2.4.238);

— user_defined_value_description_required_for_compartment_tightness (see 5.2.4.239);

— user_defined_value_description_required_for_requirement_type (see 5.2.4.240).

5.2.3.2.27 shape_representation

The base definition of the **shape_representation** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rule defined in this part of ISO 10303 applies to the **shape_representation** entity:

— representation_has_global_uncertainty_asssigned_context (see 5.2.4.182);

— shape_representation_subtype_exclusiveness (see 5.2.4.218).

5.2.3.2.28 versioned_action_request

The base definition of the **versioned_action_request** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **versioned_action_request** entity:

— author_for_change_request (see 5.2.4.16);

— date_time_for_change_request (see 5.2.4.35);

— initiator_for_change_request (see 5.2.4.47);

— versioned_action_request_with_identification_assignment (See 5.2.4.252).

5.2.4 Ship arrangement rule definitions

5.2.4.1 action_request_solution_connected_to_action

The **action_request_solution_connected_to_action** rule specifies that each instance of type **action_request_solution** with class 'change plan' shall be connected to an instance of type **action** with class 'change' via an instance of **action_method**.

EXPRESS specification:

```
*)
RULE action_request_solution_connected_to_action
FOR(action_request_solution, action);
  LOCAL
    t1_set:  SET OF action_request_solution := [];
    t2_set:  SET OF action := [];
    set_3 :  SET OF ACTION_METHOD := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

(* get all instances of action_request_solution with class 'change plan' *)
t1_set := QUERY(a <* action_request_solution |
 VALUE_IN(WHICH_CLASS(a), 'change plan'));

(* get all instances of action with class 'change' *)
t2_set := QUERY(b <* action | VALUE_IN(WHICH_CLASS(b), 'change'));

(* for all instances found above *)
REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
   set_3 := [];
   REPEAT j := 1 TO HIINDEX(t2_set);
      set_3 := set_3 + [t2_set[j].chosen_method];
   END_REPEAT;

   violate := VALUE_IN(set_3, t1_set[i].method);
END_REPEAT;

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

action_request_solution: the set of all instances of **action_request_solution** entities.

action: the set of all instances of **action** entities.

Formal propositions:

WR1: Every instance of **action_method** that is the **chosen_method** of an instance of **action** shall at the same time be the **method** of an instance of **action_request_solution**.

5.2.4.2 action_request_solution_with_identification_assignment

The **action_request_solution_with_identification_assignment** rule specifies a list of entities that require an identification. The identification is defined by the **applied_identification_assignment** entity.

EXPRESS specification:

```
*)
RULE action_request_solution_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF action_request_solution := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := ['change plan'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                    i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
     t2_set := QUERY ( j <* t2_set | j.role.name =
'globally unambiguous identifier');

      violation := NOT (SIZEOF(t2_set) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment**.

Formal propositions:

WR1: Every instance of **action_request_solution** that is referenced by an **applied_classification_-assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'change plan' shall require an **applied_identification_assignment** to define the instance identifier.

5.2.4.3 action_with_identification_assignment

The **action_with_identification_assignment** rule specifies a list of entities that require an identification. The identification is defined by the **applied_identification_assignment** entity.

EXPRESS specification:

```
*)
RULE action_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF action := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := ['change',
'versionable object change event', 'check'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances*)
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
  'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
     t2_set := QUERY ( j <* t2_set | j.role.name =
'globally unambiguous identifier');
      violation := NOT (SIZEOF(t2_set) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment .**

Formal propositions:

WR1: Every instance of **action** that is referenced by an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to ‘change’, ‘versionable object change event’, or ‘check’ shall require an **applied_identification_assignment** to define the instance identifier.

5.2.4.4 alternative_version_relationship_has_mandatory_description

The **alternative_version_relationship_has_mandatory_description** rule specifies that for an instance of **identification_assignment_relationship** with class id 'alternative version relationship' the optional attribute **description** is instantiated.

EXPRESS specification:

```
*)
RULE alternative_version_relationship_has_mandatory_description
FOR (identification_assignment_relationship);
LOCAL
    t1_set:  SET OF identification_assignment_relationship := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

(* get all instances of identification_assignment_relationship *)
(*being classified as 'alternative version relationship' *)
t1_set := QUERY(i <* identification_assignment_relationship |
  VALUE_IN(WHICH_CLASS(i), 'alternative version relationship'));

(* from all instances found above:
   find those for which attribute description is not instantiated
*)
violate := (SIZEOF(QUERY(k <* t1_set | NOT EXISTS (k.description))) > 0);

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

identification_assignment_relationship: the set of all instances of **identification_assignment_relationship** entities.

Formal propositions:

WR1: The optional attribute **description** shall exist for every instance of **identification_assignment_relationship** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'alternative version relationship'.

5.2.4.5 alternative_version_relationship_has_unique_versions

The **alternative_version_relationship_has_unique_versions** rule specifies that for all instances of **identification_assignment_relationship** with class equal to 'alternative version relationship', versionable objects alternative_1 and alternative_2 must be different, which means that they must have different version_ids.

EXPRESS specification:

```
*)
RULE alternative_version_relationship_has_unique_versions
FOR (identification_assignment_relationship);

LOCAL
    t1_set:  SET OF identification_assignment_relationship := [];
    violate: LOGICAL := FALSE;
END_LOCAL;

(* get all instances of identification_assignment_relationship with
class 'alternative version relationship' *)
   t1_set := QUERY(a <* identification_assignment_relationship |
      VALUE_IN(WHICH_CLASS(a), 'alternative version relationship'));
```



```
REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
    violate :=
          ( t1_set[i].relating_identification_assignment.assigned_id =
            t1_set[i].related_identification_assignment.assigned_id );
END_REPEAT;

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

identification_assignment_relationship: the set of all instances of **identification_assignment_-relationship** entities.

Formal propositions:

WR1: The **assigned_id**s of the **related_identification_assignment** and the **relating_-identification_assignment** of every instance of **identification_assignment_relationship** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'alternative version relationship' shall be distinct.

5.2.4.6 alternative_version_relationship_versionable_object

The **alternative_version_relationship_versionable_object** rule specifies an instance of **identification_assignment_relationship** with class 'alternative version relationship' only relates instances of type **applied_identification_assignment** whose attribute **items** references instances of class ‘versionable object’.

EXPRESS specification:

```
*)
RULE alternative_version_relationship_versionable_object
FOR (applied_identification_assignment,
     identification_assignment_relationship);
LOCAL
 violate: LOGICAL := FALSE;
END_LOCAL;

(* get all instances of applied_identification_assignment *)
REPEAT i := 1 TO HIINDEX(applied_identification_assignment) BY 1 WHILE NOT
violate;

 IF ( (SIZEOF(USEDIN(applied_identification_assignment[i],

('SHIP_ARRANGEMENT_SCHEMA.IDENTIFICATION_ASSIGNMENT_RELATIONSHIP.'
             +'RELATING_IDENTIFICATION_ASSIGNMENT'))) > 0) OR
      (SIZEOF(USEDIN(applied_identification_assignment[i],

('SHIP_ARRANGEMENT_SCHEMA.IDENTIFICATION_ASSIGNMENT_RELATIONSHIP.'
             +'RELATED_IDENTIFICATION_ASSIGNMENT'))) > 0) ) THEN
  REPEAT j := 1 to HIINDEX(applied_identification_assignment[i].items) BY 1
WHILE NOT violate;
 violate := NOT VALUE_IN(which_class(
applied_identification_assignment [i].items[j]),'versionable object');
  END_REPEAT;
 END_IF;

END_REPEAT;
```

```
WHERE
wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_identification_assignment: the set of all instances of **applied_identification_assignment** entities.

Formal propositions:

WR1: Every instance of **identification_assignment_relationship** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'alternative version relationship' shall only reference or be referenced by instances of **applied_identification_assignment** that have an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'versionable object'.

5.2.4.7 applied_approval_assignment_has_exactly_one_elements

The **applied_approval_assignment_has_exactly_one_elements** rule specifies the aggregate type attribute **items** of "assignment" entity type **applied_approval_assignment** with the **object_role** that identifies entities with attribute **name** = 'subject' to have exactly one elements.

EXPRESS specification:

```
*)
RULE applied_approval_assignment_has_exactly_one_elements
FOR (object_role, applied_approval_assignment);
WHERE
    WR1: SIZEOF(QUERY(ass_inst <* applied_approval_assignment |
           NOT((ass_inst.role.name = 'subject')
               AND
               (SIZEOF(ass_inst.items) = 1)
              )
           )) = 0;
END_RULE;
(*
```

Argument definitions:

applied_approval_assignment: the set of all instances of **applied_approval_assignment** entities.

object_role: the set of all instances of **object_role**.

Formal propositions:

WR1: Every instance of **applied_approval_assignment** that has an attribute **role.name** equal to ‘subject' shall collect one elements in its **items** attribute.

5.2.4.8 applied_group_assignment_has_at_least_one_elements

The **applied_group_assignment_has_at_least_one_elements** rule specifies the aggregate type attribute **items** of "assignment" entity type **applied_group_assignment** with the **object_role**

identifies an entity with attribute **name** = 'approvals' to have at least one elements all of which shall be of type **approval**.

EXPRESS specification:

```
*)
RULE
applied_group_assignment_has_at_least_one_elements
FOR (object_role, applied_group_assignment);
WHERE
    WR1: SIZEOF(QUERY(ass_inst <* applied_group_assignment |
           NOT((ass_inst.role.name = 'approvals')
               AND
               (SIZEOF(ass_inst.items) >= 1)
               AND
               (SIZEOF(QUERY(item <* ass_inst.items |
                  NOT('SHIP_ARRANGEMENT_SCHEMA.APPROVAL'
                  IN TYPEOF(item)))) = 0)
              )
           )) = 0;
END_RULE;
(*
```

Argument definitions:

applied_group_assignment: the set of all instances of **applied_group_assignment** entities.

object_role: the set of all instances of **object_role**.

Formal propositions:

WR1: Every **applied_group_assignment** whose attribute **role** is an entity with **name** equals 'approvals' shall collect one or more instances of **approval** in its aggregate type attribute **items**.

5.2.4.9 approval_event_with_approval_date_time

The **approval_event_with_approval_date_time** rule specifies that an instance of **approval** with class id 'approval event' is referenced by exactly one instance of **approval_date_time** via **dated_approval**.

EXPRESS specification:

```
*)
RULE approval_event_with_approval_date_time
FOR(approval);
LOCAL
    t1_set:  SET OF approval := [];
    t2_set:  SET OF approval_date_time := [];
    violate: LOGICAL := FALSE;

  END_LOCAL;

t1_set := QUERY(i <* approval |
 VALUE_IN(WHICH_CLASS(i), 'approval event'));

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
    t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPROVAL_DATE_TIME.' +
'DATED_APPROVAL'));
(* stop, if there total number is not equal to 1 *)
    violate := NOT (SIZEOF(t2_set) = 1);
```

```
  END_REPEAT;

  WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

approval: the set of all instances of **approval** entities.

Formal propositions:

WR1: Every instance of **approval** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'approval event' is referenced by exactly one instance of **approval_date_time** through **dated_approval**.

5.2.4.10 approval_event_with_approval_person_organization

The **approval_event_with_approval_person_organization** rule specifies that an instance of **approval** with class id 'approval event' is referenced by exactly one instance of **approval_person_-organization** via **authorized_approval**.

EXPRESS specification:

```
*)
RULE approval_event_with_approval_person_organization
FOR(approval);
LOCAL
    t1_set:  SET OF approval := [];
    t2_set:  SET OF approval_person_organization := [];
    violate: LOGICAL := FALSE;

  END_LOCAL;

t1_set := QUERY(i <* approval |
 VALUE_IN(WHICH_CLASS(i), 'approval event'));

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
    t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPROVAL_PERSON_ORGANIZATION.' +
'AUTHORIZED_APPROVAL'));

    violate := NOT (SIZEOF(t2_set) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

approval: the set of all instances of **approva**l entities.

Formal propositions:

WR1: Every instance of **approval** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'approval event' is referenced by exactly one instance of **approval_person_organization** through **authorized_approval**.

5.2.4.11 approval_history_approves_same_definition

The **approval_history_approves_same_definition** rule specifies that all members in a **group** with class id 'approval history' shall approve the same instance with class 'definition'.

EXPRESS specification:

```
*)
RULE approval_history_approves_same_definition
FOR (applied_group_assignment, applied_approval_assignment);
LOCAL
    t2_set:  SET OF APPLIED_GROUP_ASSIGNMENT := [];
    t3_set:  SET OF APPROVAL :=[];
    t4_set:  SET OF group_item :=[];
    t5_set:  SET OF APPLIED_APPROVAL_ASSIGNMENT := [];
    violate: LOGICAL := FALSE;
END_LOCAL;

t2_set := QUERY(a <* APPLIED_GROUP_ASSIGNMENT |
                VALUE_IN(WHICH_CLASS(a.ASSIGNED_GROUP),
          'approval history'));

t3_set := QUERY(b <* t2_set[1].items |
 'SHIP_ARRANGEMENT_SCHEMA.APPROVAL' IN TYPEOF(b));

t4_set := QUERY(b <* t2_set[1].items |
 VALUE_IN(WHICH_CLASS(b), 'definition'));

violate := NOT(SIZEOF(t4_set) = 1);

REPEAT i := 1 TO HIINDEX(t3_set) WHILE NOT violate;
    t5_set := QUERY(a <* APPLIED_APPROVAL_ASSIGNMENT |
(a.ASSIGNED_APPROVAL = t3_set[i]) AND
(NOT (VALUE_IN(a.ITEMS, t4_set[1]))));

    violate := (SIZEOF(t5_set) > 0);
  END_REPEAT;

WHERE
    wr1: NOT violate;
    wr2: (SIZEOF(t4_set) = 1);
END_RULE;

(*
```

Argument definitions:

applied_group_assignment: the set of all instances of **applied_group_assignment** entities.

applied_approval_assignment: the set of all instances of **applied_approval_assignment** entities.

Formal propositions:

WR1: Every instance of **approval** that is member of the **items** attribute of an instance of **applied_group_assignment** whose **assigned_group** is an instance of **group** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to

'approval history' shall be **assigned_approval** of an instance of **applied_approval_assignment** that collects the same instance that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'definition' in its **items** attribute.

5.2.4.12 approval_history_has_at_least_one_member

The **approval_history_has_at_least_one_member** rule specifies that each **group** with class id 'approval history' is used by exactly one **applied group assignment**.

EXPRESS specification:

```
*)
RULE approval_history_has_at_least_one_member
FOR (GROUP, APPLIED_GROUP_ASSIGNMENT);
LOCAL
    t1_set:  SET OF GROUP := [];
    t2_set:  SET OF APPLIED_GROUP_ASSIGNMENT := [];
    violate: LOGICAL := FALSE;
END_LOCAL;

t1_set := QUERY(i <* group |
 VALUE_IN(WHICH_CLASS(i), 'approval history'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
    t2_set := QUERY(a <* APPLIED_GROUP_ASSIGNMENT |
      a.ASSIGNED_GROUP = t1_set[i]);

    violate := NOT(SIZEOF(t2_set) = 1);
  END_REPEAT;

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_group_assignment: the set of all instances of **applied_group_assignment** entities.

group: the set of all instances of **group** entities.

Formal propositions:

WR1: Every instance of **group** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'approval history' shall be the **assigned_group** in exactly one instance of **applied_group_assignment**.

5.2.4.13 approvals_references_approval_history

The **approvals_references_approval_history** rule specifies that each instance of type **group** with class 'approval history' shall only be referenced by assignments of type **applied_group_assignment** with role 'approvals' via attribute **assigned_group**.

EXPRESS specification:

```
*)
```



```
RULE approvals_references_approval_history
FOR(applied_group_assignment, group);
  LOCAL
    t1_set:  SET OF group := [];
    a_set:  SET OF applied_group_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

t1_set := QUERY(a <* group |
 VALUE_IN(WHICH_CLASS(a), 'approval history'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_group_assignment |
   NOT ((b.assigned_group = t1_set[i]) AND (b.role.name = 'approvals')));

    violate := SIZEOF(a_set) > 0;
  END_REPEAT;

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_group_assignment: the set of all instances of **applied_group_assignment** entities.

group: the set of all instances of **group** entities.

Formal propositions:

WR1: Every instance of **group** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'approval history' shall only be referenced by instances of type **applied_group_assignment** whose role equals 'approvals' through attribute **assigned_group**.

5.2.4.14 author_for_change_plan

The **author_for_change_plan** rule specifies that an instance of **action_request_solution** of class 'change plan' is referenced by exactly one assignment instance of type **applied_person_and_organization_assignment** that has the **role** 'author'.

EXPRESS specification:

```
*)
RULE author_for_change_plan
FOR(applied_person_and_organization_assignment, action_request_solution);
  LOCAL
    t1_set:  SET OF action_request_solution := [];
    a_set:  SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action_request_solution |
             VALUE_IN(WHICH_CLASS(a), 'change plan'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
               (VALUE_IN(b.items, t1_set[i]) AND
               (b.role.name = 'author')));
    violate := SIZEOF(a_set) <> 1;
```

```
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_person_and_organization_assignment: the set of all instances of **applied_person_and_-organization_assignment** entities.

action_request_solution: the set of all instances of **action_request_solution** entities.

Formal propositions:

WR1: Every instance of **action_request_solution** of class 'change plan' is referenced by exactly one instance of **applied_person_and_organization_assignment** whose **role** equals 'author'.

5.2.4.15 author_for_change_realization

The **author_for_change_realization** rule specifies that an instance of **executed_action** of class 'change realization' is referenced by exactly one assignment instance of type **applied_-person_and_organization_assignment** that has the **role** 'author'.

EXPRESS specification:

```
*)
RULE author_for_change_realization
FOR(applied_person_and_organization_assignment,executed_action);
  LOCAL
    t1_set:  SET OF executed_action := [];
    a_set:  SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* executed_action |
             VALUE_IN(WHICH_CLASS(a), 'change realization'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'author')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_person_and_organization_assignment: the set of all instances of **applied_person_and_-organization_assignment** entities.

executed_action: the set of all instances of **executed_action** entities.

Formal propositions:

WR1: Every instance of **executed_action** of class 'change realization' is referenced by exactly one instance of **applied_person_and_organization_assignment** whose **role** equals 'author'.

5.2.4.16 author_for_change_request

The **author_for_change_request** rule specifies that an instance of **versioned_action_request** of class 'change request' is referenced by exactly one assignment instances of type **applied_person_and_organization_assignment** that has the **role** 'author'.

EXPRESS specification:

```
*)
RULE author_for_change_request
FOR(applied_person_and_organization_assignment, versioned_action_request);
  LOCAL
    t1_set:  SET OF versioned_action_request := [];
    a_set:  SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* versioned_action_request |
             VALUE_IN(WHICH_CLASS(a), 'change request'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'author')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_person_and_organization_assignment: the set of all instances of **applied_person_and_-organization_assignment** entities.

versioned_action_request: the set of all instances of **versioned_action_request** entities.

Formal propositions:

WR1: Every instance of **versioned_action_request** of class ‘change request’ is referenced by exactly one instance of **applied_person_and_organization_assignment** whose **role** equals 'author'.

5.2.4.17 caused_by_for_check

The **caused_by_for_check** rule specifies that an instance of **action** of class 'check' is referenced by exactly one assignment instance of type **applied_person_and_organization_assignment** that has the **role** 'caused by'.

EXPRESS specification:

```
*)
RULE caused_by_for_check
FOR(applied_person_and_organization_assignment, action);
  LOCAL
```

```
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action | VALUE_IN(WHICH_CLASS(a), 'check'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused by')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_person_and_organization_assignment: the set of all instances of **applied_person_and_-organization_assignment** entities.

action: the set of all instances of **action** entities.

Formal propositions:

WR1: Every instance of **action** of class 'check' is referenced by exactly one instance of **applied_person_and_organization_assignment** whose **role** equals 'caused by'.

5.2.4.18 caused_by_for_envisaged_version_creation

The **caused_by_for_envisaged_version_creation** rule specifies that an instance of **action** of class 'envisaged version creation' is referenced by exactly one assignment instance of type **applied_person_and_organization_assignment** that has the **role** 'caused by'.

EXPRESS specification:

```
*)
RULE caused_by_for_envisaged_version_creation
FOR(applied_person_and_organization_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action |
             VALUE_IN(WHICH_CLASS(a), 'envisaged version creation'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused by')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```


Argument definitions:

applied_person_and_organization_assignment: the set of all instances of **applied_person_and_-organization_assignment** entities.

action: the set of all instances of **action** entities.

Formal propositions:

WR1: Every instance of **action** of class 'envisaged version creation' is referenced by exactly one instance of **applied_person_and_organization_assignment** whose **role** equals 'caused by'.

5.2.4.19 caused_by_for_version_creation

The **caused_by_for_version_creation** rule specifies that an instance of **action** of class 'version creation' is referenced by exactly one assignment instance of type **applied_person_and_-organization_assignment** that has the **role** 'caused by'.

EXPRESS specification:

```
*)
RULE caused_by_for_version_creation
FOR(applied_person_and_organization_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:   SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action |
             VALUE_IN(WHICH_CLASS(a), 'version creation'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused by')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_person_and_organization_assignment: the set of all instances of **applied_person_and_-organization_assignment** entities.

action: the set of all instances of **action** entities.

Formal propositions:

WR1: Every instance of **action** of class 'version creation' is referenced by exactly one instance of **applied_person_and_organization_assignment** whose **role** equals 'caused by'.

5.2.4.20 caused_by_for_version_deletion

The **caused_by_for_version_deletion** rule specifies that an instance of **action** of class ‘version deletion’ is referenced by exactly one assignment instance of type **applied_person_and_organization_assignment** that has the **role** 'caused by'.

EXPRESS specification:

```
*)
RULE caused_by_for_version_deletion
FOR(applied_person_and_organization_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action |
             VALUE_IN(WHICH_CLASS(a), 'version deletion'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused by')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_person_and_organization_assignment: the set of all instances of **applied_person_and_-organization_assignment** entities.

action: the set of all instances of **action** entities.

Formal propositions:

WR1: Every instance of **action** of class ‘version deletion’ is referenced by exactly one instance of **applied_person_and_organization_assignment** whose **role** equals 'caused by'.

5.2.4.21 caused_by_for_version_modification

The **caused_by_for_version_modification** rule specifies that an instance of **action** of class ‘version modification’ is referenced by exactly one assignment instance of type **applied_person_and_-organization_assignment** that has the **role** 'caused by'.

EXPRESS specification:

```
*)
RULE caused_by_for_version_modification
FOR(applied_person_and_organization_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:   SET OF applied_person_and_organization_assignment := [];
```



```
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action |
             VALUE_IN(WHICH_CLASS(a), 'version modification'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused by')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_person_and_organization_assignment: the set of all instances of **applied_person_and_-organization_assignment** entities.

action: the set of all instances of **action** entities.

Formal propositions:

WR1: Every instance of **action** of class ‘version modification’ is referenced by exactly one instance of **applied_person_and_organization_assignment** whose **role** equals 'caused by'.

5.2.4.22 caused_when_for_check

The **caused_when_for_check** rule specifies that an instance of **action** of class ‘check’ is referenced by exactly one assignment instance of type **applied_date_and_time_assignment** that has the **role** 'caused when'.

EXPRESS specification:

```
*)
RULE caused_when_for_check
FOR(applied_date_and_time_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action | VALUE_IN(WHICH_CLASS(a), 'check'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused when')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_date_and_time_assignment: the set of all instances of **applied_date_and_time_-assignment** entities.

action: the set of all instances of **action** entities.

Formal propositions:

WR1: Every instance of **action** of class ‘check’ is referenced by exactly one instance of **applied_date_and_time_assignment** whose **role** equals 'caused when'.

5.2.4.23 caused_when_for_envisaged_version_creation

The **caused_when_for_envisaged_version_creation** rule specifies that an instance of **action** of class ‘envisaged version creation’ is referenced by exactly one assignment instance of type **applied_date_and_time_assignment** that has the **role** 'caused when'.

EXPRESS specification:

```
*)
RULE caused_when_for_envisaged_version_creation
FOR(applied_date_and_time_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action | VALUE_IN(WHICH_CLASS(a),
             'envisaged version creation'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused when')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_date_and_time_assignment: the set of all instances of **applied_date_and_time_-assignment** entities.

action: the set of all instances of **action** entities.

Formal propositions:

WR1: Every instance of **action** of class ‘envisaged version creation’ is referenced by exactly one instance of **applied_date_and_time_assignment** whose **role** equals 'caused when'.

5.2.4.24 caused_when_for_version_creation

The **caused_when_for_version_creation** rule specifies that an instance of **action** of class 'version creation' is referenced by exactly one assignment instance of type **applied_date_and_time_-assignment** that has the **role** 'caused when'.

EXPRESS specification:

```
*)
RULE caused_when_for_version_creation
FOR(applied_date_and_time_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action |
             VALUE_IN(WHICH_CLASS(a), 'version creation'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused when')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_date_and_time_assignment: the set of all instances of **applied_date_and_time_-assignment** entities.

action: the set of all instances of **action** entities.

Formal propositions:

WR1: Every instance of **action** of class 'version creation' is referenced by exactly one instance of **applied_date_and_time_assignment** whose **role** equals 'caused when'.

5.2.4.25 caused_when_for_version_deletion

The **caused_when_for_version_deletion** rule specifies that an instance of **action** of class 'version deletion' is referenced by exactly one assignment instance of type **applied_date_and_-time_assignment** that has the **role** 'caused when'.

EXPRESS specification:

```
*)
RULE caused_when_for_version_deletion
FOR(applied_date_and_time_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
```

```
  END_LOCAL;
   t1_set := QUERY(a <* action |
             VALUE_IN(WHICH_CLASS(a), 'version deletion'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused when')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_date_and_time_assignment: the set of all instances of **applied_date_and_time_-assignment** entities.

action: the set of all instances of **action** entities.

Formal propositions:

WR1: Every instance of **action** of class ‘version deletion’ is referenced by exactly one instance of **applied_date_and_time_assignment** whose **role** equals 'caused when'.

5.2.4.26 caused_when_for_version_modification

The **caused_when_for_version_modification** rule specifies that an instance of **action** of class ‘version modification’ is referenced by exactly one assignment instance of type **applied_date_and_time_assignment** that has the **role** 'caused when'.

EXPRESS specification:

```
*)

RULE caused_when_for_version_modification
FOR(applied_date_and_time_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action |
             VALUE_IN(WHICH_CLASS(a), 'version modification'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused when')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```


Argument definitions:

applied_date_and_time_assignment: the set of all instances of **applied_date_and_time_-assignment** entities.

action: the set of all instances of **action** entities.

Formal propositions:

WR1: Every instance of **action** of class ‘version modification’ is referenced by exactly one instance of **applied_date_and_time_assignment** whose **role** equals 'caused when'.

5.2.4.27 centre_location_compound_representation_has_specified_name

The **centre_location_compound_representation_has_specified_name** rule specifies the **item_-element** attribute of a **compound_representation_item** with the class id 'centre location' to have in the **list_representation_item** for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list.

EXPRESS specification:

```
*)

RULE centre_location_compound_representation_has_specified_name
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF Compound_representation_item := [];
    t2_set:  SET OF representation_item := [];
    arg_list:  LIST OF STRING := ['longitudinal location',
   'transversal location', 'vertical location'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'centre location'
*)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'centre location');

  (* get all instances of compound_representation_item that have class id
'centre location' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* iterate over all compound_representation_item found above; stop, if
one of them has not exactly one rep_item for each name in the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
   violation := (SIZEOF(QUERY(items <* t2_set | items.name = arg_list[j]))
<> 1);
 END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;

END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **compound_representation_item** that has an **applied_classification_-assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'centre location' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall for each value out of 'longitudinal location', 'transversal location', 'vertical location' collect exactly one instance of **representation_item** whose **name** attribute equals that value.

5.2.4.28 change_impact_with_versionable_object_change_event

The **change_impact_with_versionable_object_change_event** rule specifies that each assignment of type **applied_action_request_assignment** with **role** 'change impact' shall refer to at least one element with class 'versionable object change event' through its attribute **items**.

EXPRESS specification:

```
*)
RULE change_impact_with_versionable_object_change_event
FOR(applied_action_request_assignment);
  LOCAL
    t1_set:  SET OF applied_action_request_assignment := [];
    a_set:   SET OF action := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

   t1_set := QUERY(b <* applied_action_request_assignment |
                   (b.role.name= 'change impact'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
          a_set := QUERY(b <* t1_set[i].items |
          ('SHIP_ARRANGEMENT_SCHEMA.ACTION' IN TYPEOF(b)) AND
             VALUE_IN(WHICH_CLASS(b), 'versionable object change event'));
         violate := SIZEOF(a_set) = 0;
      END_REPEAT;

WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_action_request_assignment: the set of all instances of **applied_action_request_-assignment** entities.

Formal propositions:

WR1: Every instance of **applied_action_request_assignment** whose **role** equals 'change impact' shall reference at least one instance that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'versionable object change event' through its attribute **items**.

5.2.4.29 change_plan_has_mandatory_attribute_description

The **change_plan_has_mandatory_attribute_description** rule specifies that for an instance of **action_request_solution** with class of 'change plan' the optional attribute **description** is instantiated.

EXPRESS specification:

```
*)
RULE change_plan_has_mandatory_attribute_description
FOR (action_request_solution);
LOCAL
    t1_set:  SET OF action_request_solution := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
  t1_set := QUERY(i <* action_request_solution |
  VALUE_IN(WHICH_CLASS(i), 'change plan'));
  violate := (SIZEOF(QUERY(k <* t1_set |
  NOT EXISTS (k.description))) > 0);
WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

action_request_solution: the set of all instances of **action_request_solution** entities.

Formal propositions:

WR1: The optional attribute **description** shall exist for every instance of **action_request_solution** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'change plan'.

5.2.4.30 class_and_statutory_designation_has_properties

The **class_and_statutory_designation_has_properties** rule specifies that a **product_definition** with a class of 'class and statutory designation' is referenced by one **property_definition_-representation** with the **name** 'class and statutory designation' via a **property_definition**.

EXPRESS specification:

```
*)
RULE class_and_statutory_designation_has_properties
FOR (property_definition_representation,
applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_list: LIST OF product_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    t3_list: LIST OF property_definition := [];
    t4_list: LIST OF product_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME =
                  'class and statutory designation');
```

```
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_list := t1_list + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
                  i.NAME = 'class and statutory designation');
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_list := t3_list + t2_set[i].definition;
  END_REPEAT;
  REPEAT i := 1 TO HIINDEX(t3_list);
    t4_list := t4_list + t3_list[i].definition;
  END_REPEAT;
  violation := t1_list <> t4_list;
  WHERE
    WR1: NOT violation;

END_RULE;

(*
```

Argument definitions:

property_definition_representation: the set of all instances of **property_definition_-representation** entities.

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **product_definition** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'class and statutory designation' shall be referenced by exactly one instance of **property_definition** through attribute **definition** that in turn is referenced by an instance of **property_definition_representation** through attribute **definition** whose attribute **name** equals 'class and statutory designation'.

5.2.4.31 class_notation_with_named_representation_items

The **class_notation_with_named_representation_items** rule specifies the **items** attribute of a **representation** to have for each entry in the list one or more **representation_item**s whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'class notation'.

EXPRESS specification:

```
*)
RULE class_notation_with_named_representation_items
FOR (representation);
LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['class notations hull',
                                  'class notations machinery'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  reps := QUERY(
            temp_rep <* representation |
```



```
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
               'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
               'USED_REPRESENTATION')) |
              (temp_prop_def_rep.name = 'class notation')
                    )
               ) > 0
            );

  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) < 1);
    END_REPEAT;
  END_REPEAT;
  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every instance of **representation** that is the **used_representation** in an instance of **property_definition_representation** whose attribute **name** equals 'class notation' shall for each value out of 'class notations hull', or 'class notations machinery' collect at least one instance of **representation_item** in its attribute **items** whose **name** attribute equals that value.

5.2.4.32 class_parameters_has_properties

The **class_parameters_has_properties** rule specifies that a **product_definition** with a class of 'class parameters' is referenced by one **property_definition_representation** with the **name** 'class parameters' via a **property_definition**.

EXPRESS specification:

```
*)
RULE class_parameters_has_properties
FOR (property_definition_representation,
applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set: LIST OF product_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    t3_set: LIST OF property_definition := [];
    t4_set: LIST OF product_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME =
                  'class parameters');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
```

```
    END_REPEAT;
  END_REPEAT;

  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
                  i.NAME = 'class parameters');
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
  END_REPEAT;
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;
  violation := t1_set <> t4_set;
  WHERE
    WR1: NOT violation;

END_RULE;

(*
```

Argument definitions:

property_definition_representation: the set of all instances of **property_definition_-representation** entities.

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **product_definition** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'class parameters' shall be referenced by exactly one instance of **property_definition** through attribute **definition** that in turn is referenced by an instance of **property_definition_representation** through attribute **definition** whose attribute **name** equals 'class parameters'.

5.2.4.33 date_time_for_change_plan

The **date_time_for_change_plan** rule specifies that an instance of **action_request_solution** of class 'change plan' is referenced by exactly one assignment instance of type **applied_date_and_time_-assignment** that has the **role** 'date time'.

EXPRESS specification:

```
*)
RULE date_time_for_change_plan
FOR(applied_date_and_time_assignment,  action_request_solution);
  LOCAL
    t1_set:  SET OF action_request_solution := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action_request_solution |
             VALUE_IN(WHICH_CLASS(a), 'change plan'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'date time')));
    violate := SIZEOF(a_set) <> 1;
```



```
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_date_and_time_assignment: the set of all instances of **applied_date_and_time_-assignment** entities.

action_request_solution: the set of all instances of **action_request_solution** entities.

Formal propositions:

WR1: Every instance of **action_request_solution** of class 'change plan' is referenced by exactly one instance of **applied_date_and_time_assignment** whose **role** equals 'date time'.

5.2.4.34 date_time_for_change_realization

The **date_time_for_change_realization** rule specifies that an instance of **executed_action** of class ‘change realization’ is referenced by exactly one assignment instance of type **applied_date_and_-time_assignment** that has the role 'date time'.

EXPRESS specification:

```
*)
RULE date_time_for_change_realization
FOR(applied_date_and_time_assignment, executed_action);
  LOCAL
    t1_set:  SET OF executed_action := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* executed_action |
             VALUE_IN(WHICH_CLASS(a), 'change realization'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'date time')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_date_and_time_assignment: the set of all instances of **applied_date_and_time_-assignment** entities.

executed_action: the set of all instances of **executed_action** entities.

Formal propositions:

WR1: Every instance of **executed_action** of class ‘change realization’ is referenced by exactly one instance of **applied_date_and_time_assignment** whose **role** equals 'date time'.

5.2.4.35 date_time_for_change_request

The **date_time_for_change_request** rule specifies that an instance of **versioned_action_request** of class ‘change request’ is referenced by exactly one assignment instance of type **applied_date_-and_time_assignment** that has the **role** 'date time'.

EXPRESS specification:

```
*)
RULE date_time_for_change_request
FOR(applied_date_and_time_assignment, versioned_action_request);
  LOCAL
    t1_set:  SET OF versioned_action_request := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* versioned_action_request |
             VALUE_IN(WHICH_CLASS(a), 'change request'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'date time')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_date_and_time_assignment: the set of all instances of **applied_date_and_time_-assignment** entities.

versioned_action_request: the set of all instances of **versioned_action_request** entities.

Formal propositions:

WR1: Every instance of **versioned_action_request** of class ‘change request’ is referenced by exactly one instance of **applied_date_and_time_assignment** whose **role** equals 'date time'.

5.2.4.36 document_has_at_least_one_references

The **document_has_at_least_one_references** rule specifies that an instance of **document** of class 'document' is referenced by at least one instance of **document_representation_type** via **represented_document**.

EXPRESS specification:

```
*)
RULE document_has_at_least_one_references
FOR(document);
```



```
  LOCAL
    t1_set:  SET OF document := [];
    t2_set:  SET OF document_representation_type := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
  t1_set := QUERY(i <* document | VALUE_IN(WHICH_CLASS(i),
                                  'document'));

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
    t2_set := bag_to_set(USEDIN(t1_set[i],
      'SHIP_ARRANGEMENT_SCHEMA.DOCUMENT_REPRESENTATION_TYPE.'+
      'REPRESENTED_DOCUMENT'));
    violate := SIZEOF(t2_set) < 1;
END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

document: the set of all instances of **document** entities.

Formal propositions:

WR1: Every instance of **document** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'document' shall be referenced by one or more instances of **document_representation_type** through attribute **represented_document**.

5.2.4.37 document_has_exactly_one_author

The **document_has_exactly_one_author** rule specifies that each instance of type **document** shall be referenced by exactly one instance of type **applied_person_assignment**, **applied_organization_-assignment**, or **applied_person_and_organization_assignment**, where the latter instance has the **role** 'author'.

EXPRESS specification:

```
*)
RULE document_has_exactly_one_author
FOR(document);
  LOCAL
    bag_1: BAG OF applied_person_assignment := [];
    bag_2: BAG OF applied_person_and_organization_assignment := [];
    bag_3: BAG OF applied_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
REPEAT i := 1 TO SIZEOF(document) WHILE (NOT violate);
  bag_1 := USEDIN(document[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
         'APPLIED_PERSON_ASSIGNMENT.ITEMS');
     bag_1 := QUERY( assign <* bag_1 | assign.role.name = ‘author’);
   bag_2 := USEDIN(document[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
         'APPLIED_PERSON_AND_ORGANIZATION_ASSIGNMENT.ITEMS');
  bag_2 := QUERY( assign <* bag_2 | assign.role.name = ‘author’);
  bag_3 := USEDIN(document[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
         'APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS');

  bag_3 := QUERY( assign <* bag_3 | assign.role.name = ‘author’);
    violate := NOT ((SIZEOF( bag_1 ) + SIZEOF( bag_2 )+ SIZEOF( bag_3 ))=
1);
END_REPEAT;
```

```
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

document: the set of all instances of **document** entities.

Formal propositions:

WR1: Every instance of **document** shall be referenced by exactly one instance of **applied_organization_assignment**, **applied_person_assignment**, or **applied_person_and_organization_assignment** whose role equals 'author'.

5.2.4.38 document_reference_with_address_has_at_least_one_references

The **document_reference_with_address_has_at_least_one_references** rule specifies that an instance of **document** with class id 'document reference with address' is referenced by at least one instance of **applied_external_identification_assignment** via **items**.

EXPRESS specification:

```
*)
RULE document_reference_with_address_has_at_least_one_references
FOR(document);
  LOCAL
    t1_set:  SET OF document := [];
    t2_set:  SET OF applied_external_identification_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
t1_set := QUERY(i <* document | VALUE_IN(WHICH_CLASS(i),
                                 'document reference with address'));

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
    t2_set := bag_to_set(USEDIN(t1_set[i],

'SHIP_ARRANGEMENT_SCHEMA.APPLIED_EXTERNAL_IDENTIFICATION_ASSIGNMENT.ITEMS')
);
    violate := SIZEOF(t2_set) < 1;
END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

document: the set of all instances of **document** entities.

Formal propositions:

WR1: Every instance of **document** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'document reference with address' shall be referenced by one or more instances of **applied_external_identification_assignment** through attribute **items**.

5.2.4.39 envisaged_version_creation_has_mandatory_attribute_description

The **envisaged_version_creation_has_mandatory_attribute_description** rule specifies that for an instance of **action** with class id 'envisaged version creation' the optional attribute **description** is instantiated.

EXPRESS specification:

```
*)
RULE envisaged_version_creation_has_mandatory_attribute_description
FOR (action);
LOCAL
    t1_set:  SET OF action := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(i <* action | VALUE_IN(WHICH_CLASS(i),
   'envisaged version creation'));
   violate := (SIZEOF(QUERY(k <* t1_set |
   NOT EXISTS (k.description))) > 0);
WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

action: the set of all instances of **action** entities.

Formal propositions:

WR1: The optional attribute **description** shall exist for every instance of **action** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'envisaged version creation'.

5.2.4.40 executed_action_with_identification_assignment

The **executed_action_with_identification_assignment** rule specifies a list of entities that require an identification. The identification is defined by the **applied_identification_assignment** entity.

EXPRESS specification:

```
*)
RULE executed_action_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF executed_action := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := [ 'change realization'];
    violation: LOGICAL := FALSE;
  END_LOCAL;


    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
```

```
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
  t2_set := QUERY ( j <* t2_set | j.role.name =
  'globally unambiguous identifier');
      violation := NOT (SIZEOF(T2_SET) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;
```

(*

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment**.

Formal propositions:

WR1: Every instance of **executed_action** that is referenced by an **applied_classification_-assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'change realization' shall require an **applied_identification_assignment** to define the instance identifier.

5.2.4.41 external_instance_reference_has_same_identifier

The **external_instance_reference_has_same_identifier** rule verifies the global identifier stored in the **assigned_id** attribute of an **applied_external_identification_assignment** is identical to the global identifier stored in the **assigned_id** attribute of an **applied_identification_assignment** for every entity which is referred to by both an **applied_external_identification_assignment** and an **applied_identification_assignment**.

EXPRESS specification:

```
*)
RULE external_instance_reference_has_same_identifier
FOR (applied_external_identification_assignment);
LOCAL
 violation    : LOGICAL := FALSE;
 extref_set   : SET OF applied_external_identification_assignment := [];
 aia_set      : SET OF applied_identification_assignment := [];
END_LOCAL;

 extref_set := QUERY ( i <* applied_external_identification_assignment |
  (i.role.name = 'external instance reference'));

 REPEAT i := 1 TO HIINDEX(extref_set) BY 1 WHILE NOT violation;
   aia_set := USEDIN(extref_set[i].items[1],
       'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS');
   violation := NOT (aia_set[1].assigned_id = extref_set[i].assigned_id);
 END_REPEAT;
 WHERE
   wr1: NOT violation;
 END_RULE;
```



```
(*
```

Argument definitions:

applied_external_identification_assignment: the set of all instances of **applied_external_-identification_assignment** entities.

Formal propositions:

WR1: For every instance that is referred to by both an **applied_external_identification_assignment** in the role 'external instance reference' and an **applied_identification_assignment**, the value stored in the **assigned_id** attribute of each of these assignment entities should be the same.

5.2.4.42 floating_position_compound_representation_with_name

The **floating_position_compound_representation_with_name** rule specifies the **item_element** attribute of a **compound_representation_item** with the class id 'floating position' to have in the **list_representation_item** for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list.

EXPRESS specification:

```
*)
RULE floating_position_compound_representation_with_name
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF Compound_representation_item := [];
    t2_set:  SET OF representation_item := [];
    arg_list:  LIST OF STRING := ['moulded form displacement',
     'draught at amidships', 'length of waterline', 'breadth of waterline',
     'angle of trim', 'angle of heel'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.Assigned_class.name = 'floating position');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
  violation := (SIZEOF(QUERY(items <* t2_set | items.name = arg_list[j]))
<> 1);
 END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;

END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **compound_representation_item** that has an **applied_classification_-assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'floating position' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall for each value of 'moulded form displacement', 'draught at amidships', 'length of waterline', 'breadth of waterline', 'angle of trim', and 'angle of heel' collect exactly one instance of **representation_item** whose **name** attribute equals that value.

5.2.4.43 freeboard_characteristics_has_properties

The **freeboard_characteristics_has_properties** rule specifies that a **product_definition** with a class id 'freeboard characteristics' is referenced by one **property_definition_representation** with the **name** 'freeboard characteristics' via a **property_definition**.

EXPRESS specification:

```
*)
RULE freeboard_characteristics_has_properties
FOR (property_definition_representation,
applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set: LIST OF product_definition := [];
    t2_set: SET OF property_definition_representation := [];
    t3_set: LIST OF property_definition := [];
    t4_set: LIST OF product_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME =
                  'freeboard characteristics');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
                  i.NAME = 'freeboard characteristics');
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
  END_REPEAT;
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;
  violation := t1_set <> t4_set;
  WHERE
    WR1: NOT violation;

END_RULE;

(*
```


Argument definitions:

property_definition_representation: the set of all instances of **property_definition_representation** entities.

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **product_definition** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'freeboard characteristics' shall be referenced by exactly one instance of **property_definition** through attribute **definition** that in turn is referenced by an instance of **property_definition_representation** through attribute **definition** whose attribute **name** equals 'freeboard characteristics'.

5.2.4.44 global_axis_placement_has_properties

The **global_axis_placement_has_properties** rule specifies that a **product_definition** with a class of 'global axis placement' is referenced by one **property_definition_representation** with the **name** 'global axis placement' via a **property_definition**.

EXPRESS specification:

```
*)
RULE global_axis_placement_has_properties
FOR (property_definition_representation,
group, applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set: LIST OF product_definition := [];
    t2_set: SET OF property_definition_representation := [];
    t3_set: LIST OF property_definition := [];
    t4_set: LIST OF product_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =
                   'global axis placement');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
                   i.NAME = 'global axis placement');
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
  END_REPEAT;
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;
  violation := t1_set <> t4_set;
  WHERE
    WR1: NOT violation;

END_RULE;
```

```
(*
```

Argument definitions:

property_definition_representation: the set of all instances of **property_definition_-representation** entities.

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **product_definition** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'global axis placement' shall be referenced by exactly one instance of **property_definition** through attribute **definition** that in turn is referenced by an instance of **property_definition_representation** through attribute **definition** whose attribute **name** equals 'global axis placement'.

5.2.4.45 global_id_is_unique

The **global_id_is_unique** rule specifies that the global identifiers of definable objects are unique, and that each identifier is only assigned to one definable object.

EXPRESS specification:

```
*)
RULE global_id_is_unique
FOR (APPLIED_IDENTIFICATION_ASSIGNMENT);
  LOCAL
    set_1:  SET OF APPLIED_IDENTIFICATION_ASSIGNMENT:= [];
    bag_2:     BAG OF STRING := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all instances of guid  *)


  set_1 := QUERY(i <* APPLIED_IDENTIFICATION_ASSIGNMENT |
          (i.role.name = 'globally unambiguous identifier'));

REPEAT i := 1 TO HIINDEX(set_1);
    bag_2 := bag_2 + [set_1[i].assigned_id];

END_REPEAT;
violation := SIZEOF (QUERY(i <* set_1 | (SIZEOF(i.items) = 1))) <>
SIZEOF(set_1);

WHERE
    WR1: VALUE_UNIQUE(bag_2);
    WR2: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_identification_assignment: the set of all instances of **applied_identification_assignment** entities.

Formal propositions:

WR1: Every **applied_identification_assignment** that has an atttribute **role** that references an **identification_role** with **name** of 'globally unambiguous identifier' shall have a unique value for **assigned_id**.

WR2: Every **applied_identification_assignment** that has an atttribute **role** that references an **identification_role** with **name** = 'globally unambiguous identifier' shall reference only one definable object in attribute **items**.

5.2.4.46 identification_role_optional_attribute_description_required

The **identification_role_optional_attribute_description_required** rule specifies that for all instances of type **identification_role** the optional attribute **description** is present if attribute **name** of that instance has a value 'external reference'.

EXPRESS specification:

```
*)
RULE identification_role_optional_attribute_description_required
FOR (identification_role);
WHERE
 wr1: SIZEOF(QUERY(i <* identification_role |
     ((i.name = 'external reference')
     AND NOT(EXISTS (i.description))))) = 0;
END_RULE;

(*
```

Argument definitions:

identification_role: the set of all instances of **identification_role** entities.

Formal propositions:

WR1: Optional attribute **description** of every instance of **identification_role** exists if **name** equals 'external reference'.

5.2.4.47 initiator_for_change_request

The **initiator_for_change_request** rule specifies that an instance of **versioned_action_request** of class ‘change request’ is referenced by exactly one assignment instance of type **applied_person_-and_organization_assignment** that has the **role** 'initiator'.

EXPRESS specification:

```
*)
RULE initiator_for_change_request
FOR(applied_person_and_organization_assignment,
    versioned_action_request);
  LOCAL
    t1_set:  SET OF versioned_action_request := [];
    a_set:  SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
```

```
  END_LOCAL;
   t1_set := QUERY(a <* versioned_action_request |
             VALUE_IN(WHICH_CLASS(a), 'change request'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'initiator')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_person_and_organization_assignment: the set of all instances of **applied_person_and_-organization_assignment** entities.

versioned_action_request: the set of all instances of **versioned_action_request** entities.

Formal propositions:

WR1: Every instance of **versioned_action_request** whose class equals ‘change request’ is referenced by exactly one instance of **applied_person_and_organization_assignment** whose **role** equals 'initiator'.

5.2.4.48 lightship_definition_has_properties

The **lightship_definition_has_properties** rule specifies that a **product_definition** with a class of 'lightship definition' is referenced by one **property_definition_representation** with the **name** 'lightship definition parameters' via a **property_definition**.

EXPRESS specification:

```
*)
RULE lightship_definition_has_properties

FOR (property_definition_representation,
group, applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set: LIST OF product_definition := [];
    t2_set: SET OF property_definition_representation := [];
    t3_set: LIST OF property_definition := [];
    t4_set: LIST OF product_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME =
                  'lightship definition');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;
```



```
  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
                  i.NAME = 'lightship definition parameters');
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
  END_REPEAT;
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;
  violation := t1_set <> t4_set;
  WHERE
    WR1: NOT violation;

END_RULE;

(*
```

Argument definitions:

property_definition_representation: the set of all instances of **property_definition_-representation** entities.

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **product_definition** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'lightship definition' shall be referenced by exactly one instance of **property_definition** through attribute **definition** that in turn is referenced by an instance of **property_definition_representation** through attribute **definition** whose attribute **name** equals 'lightship definition parameters'.

5.2.4.49 mandatory_entity_type_for_external_instance_reference

The **mandatory_entity_type_for_external_instance_reference** rule specifies that every instance of **external_source** whose **description** equals 'schema name' must be the **relating_source** of an instance of **external_source_relationship** whose **related_source** is an instance of **external_source** that has a **description** equal to 'entity type'.

EXPRESS specification:

```
*)
RULE mandatory_entity_type_for_external_instance_reference
FOR(external_source, external_source_relationship);
  LOCAL
    bag_1: BAG OF external_source := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
bag_1 := QUERY(a <* external_source | a.description = 'schema name');

REPEAT i := 1 TO SIZEOF(bag_1) WHILE (NOT violate);
violate := (SIZEOF( QUERY(
    a <* external_source_relationship | (a.relating_source :=: bag_1[i])
AND (a.related_source.description = 'entity type'))) = 0 );
END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;
```

```
(*
```

Argument definitions:

external_source: the set of all instances of **external_source** entities.

external_source_relationship: the set of all instances of **external_source_relationship** entities.

Formal propositions:

WR1: Every instance of **external_source** whose **description** equals 'schema name' must be the **relating_source** of an instance of **external_source_relationship** whose **related_source** is an instance of **external_source** that has a **description** equal to 'entity type'.

5.2.4.50 members_is_referenced_by_at_least_one_revision

The **members_is_referenced_by_at_least_one_revision** rule specifies that each instance of type **group** of class 'revision' shall be referenced by at least one assignment of type **applied_group_-assignment** with role 'members' via attribute **assigned_group**

EXPRESS specification:

```
*)
RULE members_is_referenced_by_at_least_one_revision
FOR(applied_group_assignment, group);
  LOCAL
    t1_set:  SET OF group := [];
    a_set:  SET OF applied_group_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

t1_set := QUERY(a <* group | VALUE_IN(WHICH_CLASS(a), 'revision'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
   a_set := QUERY(b <* applied_group_assignment |
       (b.assigned_group :=: t1_set[i]) AND (b.role.name = 'members'));

    violate := SIZEOF(a_set) < 1;
  END_REPEAT;

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_group_assignment: the set of all instances of **applied_group_assignment** entities.

group: the set of all instances of **group** entities

Formal propositions:

WR1: Every instance of **group** that has an **applied_classification_assignment** whose **assigned_class** identifies an entity with attribute **name** equal 'revision' shall be referenced by one or more instances of type **applied_group_assignment** whose role equals 'members' through attribute **assigned_group**.

5.2.4.51 no_approvals_except_in_approval_history

The **no_approvals_except_in_approval_history** rule specifies that there shall be no instance of type **approval** of class 'approval event', that is not part of a **group** of class 'approval history'.

EXPRESS specification:

```
*)
RULE no_approvals_except_in_approval_history
FOR (approval);
LOCAL
    t1_set:  SET OF approval := [];
    t2_set:  SET OF APPLIED_GROUP_ASSIGNMENT := [];
    violate: LOGICAL := FALSE;
END_LOCAL;
   t1_set := QUERY(a <* approval |
      VALUE_IN(WHICH_CLASS(a), 'approval event'));
REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
   t2_set := bag_to_set(USEDIN(t1_set[i],
   'SHIP_ARRANGEMENT_SCHEMA.APPLIED_GROUP_ASSIGNMENT.ITEMS'));
    violate := (SIZEOF(t2_set) = 0);
   REPEAT k := 1 TO HIINDEX(t2_set) WHILE NOT violate;
      violate := NOT (VALUE_IN(WHICH_CLASS(t2_set[k].ASSIGNED_GROUP),
      'approval history'));
   END_REPEAT;
END_REPEAT;
WHERE
    WR1: NOT violate;
END_RULE;
(*
```

Argument definitions:

approval: the set of all instances of **approval** entities.

Formal propositions:

WR1: Every instance of **approval** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal 'approval event' shall be collected in the **items** of an instance of **applied_group_assignment** whose **assigned_group** has an **applied_-classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal 'approval history'.

5.2.4.52 principal_characteristics_has_properties

The **principal_characteristics_has_properties** rule specifies that a **product_definition** with a class id 'principal characteristics' is referenced by one **property_definition_representation** with the **name** 'principal characteristics' via a **property_definition**.

EXPRESS specification:

```
*)
RULE principal_characteristics_has_properties
FOR (property_definition_representation,
applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set: LIST OF product_definition := [];
    t2_set: SET OF property_definition_representation := [];
    t3_set: LIST OF property_definition := [];
```

```
    t4_set: LIST OF product_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.assigned_class.NAME = 'principal characteristics');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
                  i.NAME = 'principal characteristics');
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
  END_REPEAT;
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;
  violation := t1_set <> t4_set;
  WHERE
    WR1: NOT violation;

END_RULE;

(*
```

Argument definitions:

property_definition_representation: the set of all instances of **property_definition_-representation** entities.

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **product_definition** that has an **applied_classification_assignment** whose attribute **assigned_class** is a **group** with attribute **name** equal to 'principal characteristics' shall be referenced by exactly one instance of **property_definition** through attribute **definition** that in turn is referenced by an instance of **property_definition_representation** through attribute **definition** whose attribute **name** equals 'principal characteristics'.

5.2.4.53 product_definition_for_call_sign

The **product_definition_for_call_sign** rule specifies that an instance of **product_definition** with class id 'ship designation' is referenced by exactly one instance of **applied_identification_-assignment** via **items** whose attribute **role.name** has the value 'call sign'.

EXPRESS specification:

```
*)
RULE product_definition_for_call_sign
FOR(applied_classification_assignment);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_identification_assignment := [];
```

```
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME = 'ship designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
     t2_inst.role.name = 'call sign')) = 1);
  END_REPEAT;
  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **product_definition** that is referenced by a **applied_classification_assignment** of class 'ship designation', is referenced by exactly one **applied_identification_assignment** attribute **items**, where **applied_identification_assignment** attribute **role.name** = 'call sign'.

5.2.4.54 product_definition_for_certifying_organization

The **product_definition_for_certifying_organization** rule specifies that an instance of **product_definition** with class id 'coating certification' is referenced by exactly one instance of **applied_organization_assignment** via **items** whose attribute **role.name** has the value 'certifying organization'.

EXPRESS specification:

```
*)
RULE product_definition_for_certifying_organization
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
             i.assigned_class.name = 'coating certification');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
```

```
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
   'APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
   t2_inst.role.name = 'certifying organization')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **product_definition** that is referenced by an **applied_classification_assignment** of class ‘coating certification’ is referenced by exactly one instance of **applied_organization_-assignment** via **items** whose attribute **role.name** has the value ‘certifying organization’.

5.2.4.55 product_definition_for_class_notation

The **product_definition_for_class_notation** rule specifies that an instance of **product_definition** with class id 'class and statutory designation' is referenced by exactly one instance of **property_-definition** with class id 'class notation' via **definition**.

EXPRESS specification:

```
*)
RULE product_definition_for_class_notation
FOR(applied_classification_assignment);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF property_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =
                   'class and statutory designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
      'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
      'class notation' IN WHICH_CLASS(t2_inst))) = 1);
END_REPEAT;
  WHERE
```



```
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **product_definition** that has an **applied_classification_assignment** whose attribute **assigned_class** has an entity with **name** attribute of value 'class and statutory designation' shall be referenced by exactly one instance of **property_definition** that has an **applied_classification_assignment** whose **assigned_class** is an entity with **name** attribute of value 'class notation' through attribute **definition**.

5.2.4.56 product_definition_for_expiry_date

The **product_definition_for_expiry_date** rule specifies that an instance of **product_definition** with class id 'coating certification' is referenced by exactly one instance of **applied_date_and_time_assignment** via **items** whose attribute **role** has the value 'expiry date'.

EXPRESS specification:

```
*)
RULE product_definition_for_expiry_date
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_date_and_time_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'coating
certification' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'coating certification');

  (* get all instances of T1 that have class id 'coating certification' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of product_definition in t1_set:
     get the applied_date_and_time_assignment instances that are
referencing a product_definition instance via items, filter out those
applied_date_and_time_assignment instances whose attribute role has the
value ‘expiry date’; check if their number equals 1
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.role.name = 'expiry date')) = 1);
  END_REPEAT;
```

```
  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **product_definition** that is referenced by an **applied_classification_assignment** with **name** of ‘coating certification’, is referenced by exactly one **applied_date_and_time_-assignment.items**, where **applied_date_and_time_assignment.role** equals ‘expiry date’.

5.2.4.57 product_definition_for_flag_state

The **product_definition_for_flag_state** rule specifies that an instance of **product_definition** with class id 'ship designation' is referenced by exactly one instance of **applied_identification_-assignment** via **items** whose attribute **role.name** has the value 'flag state'.

EXPRESS specification:

```
*)
RULE product_definition_for_flag_state
FOR(applied_classification_assignment);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_identification_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME = 'ship designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
    'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
    'flag state')) = 1);
  END_REPEAT;
  WHERE
    wr1: NOT violation;
END_RULE;

(*
```


Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **product_definition** that is referenced by an **applied_classification_assignment** of class 'ship designation', is referenced by exactly one **applied_identification_assignment** attribute **items**, where **applied_identification_assignment** attribute **role.name** = 'flag state'.

5.2.4.58 product_definition_for_loadline

The **product_definition_for_loadline** rule specifies that an instance of **product_definition** with class id 'freeboard characteristics' is referenced by exactly one instance of **property_definition** with class id 'loadline' via **definition**.

EXPRESS specification:

```
*)
RULE product_definition_for_loadline
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF property_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'freeboard characteristics');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
   'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
   'loadline' IN WHICH_CLASS(t2_inst))) = 1);
END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **product_definition** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'freeboard characteristics' shall be referenced

by exactly one instance of **property_definition** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'loadline' through attribute **definition**.

5.2.4.59 product_definition_for_managing_company

The **product_definition_for_managing_company** rule specifies that an instance of **product_definition** with class id 'owner designation' is referenced by exactly one instance of **applied_organization_assignment** via **items** whose attribute **role.name** has the value 'managing company'.

EXPRESS specification:

```
*)
RULE product_definition_for_managing_company
FOR(applied_classification_assignment);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME =
                   'owner designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
    'APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
    'managing company')) = 1);
  END_REPEAT;
  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **product_definition** that is referenced by an **applied_classification_assignment** of class 'owner designation' is referenced by exactly one **applied_organization_assignment** attribute **items**, where **applied_organization_assignment** attribute **role.name** = 'managing company'.

5.2.4.60 product_definition_for_ordering_company

The **product_definition_for_ordering_company** rule specifies that an instance of **product_definition** with class id 'owner designation' is referenced by exactly one instance of **applied_organization_assignment** via **items** whose attribute **role.name** has the value 'ordering company'.

EXPRESS specification:

```
*)
RULE product_definition_for_ordering_company
FOR(applied_classification_assignment);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME = 'owner designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
    'APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
    'ordering company')) = 1);
  END_REPEAT;
  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **product_definition** that is referenced by an **applied_classification_assignment** of class 'owner designation' is referenced by exactly one **applied_organization_assignment** attribute **items**, where **applied_organization_assignment** attribute **role.name** = 'ordering company'.

5.2.4.61 product_definition_for_owning_company

The **product_definition_for_owning_company** rule specifies that an instance of **product_-definition** with class id 'owner designation' is referenced by exactly one instance of **applied_-organization_assignment** via **items** whose attribute **role.name** has the value 'owning company'.

EXPRESS specification:

```
*)
RULE product_definition_for_owning_company
FOR(applied_classification_assignment);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;
```

```
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =
                   'owner designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
     'SHIP_ARRANGEMENT_SCHEMA.APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
    'owning company')) = 1);
  END_REPEAT;
  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **product_definition** that is referenced by an **applied_classification_assignment** of class 'owner designation', is referenced by exactly one **applied_organization_assignment** attribute **items**, where **applied_organization_assignment** attribute **role.name** = 'owning company'.

5.2.4.62 product_definition_for_port_of_registration

The **product_definition_for_port_of_registration** rule specifies that an instance of **product_-definition** with class id 'ship designation' is referenced by exactly one instance of **applied_-identification_assignment** via **items** whose attribute **role.name** has the value 'port of registration'.

EXPRESS specification:

```
*)
RULE product_definition_for_port_of_registration
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_identification_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME = 'ship designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;
```



```
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
    'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
     'port of registration')) = 1);
  END_REPEAT;
  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **product_definition** that is referenced by an **applied_classification_assignment** of class with 'ship designation', is referenced by exactly one **applied_identification_assignment** attribute **items**, where **applied_identification_assignment** attribute **role.name** = 'port of registration'.

5.2.4.63 product_definition_for_regulation

The **product_definition_for_regulation** rule specifies that an instance of **product_definition** with class id 'class and statutory designation' is referenced by exactly one instance of **property_definition** with class id 'regulation' via **definition.**

EXPRESS specification:

```
*)
RULE product_definition_for_regulation
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF property_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
             i.assigned_class.NAME = 'class and statutory designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
     'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
     'regulation' IN WHICH_CLASS(t2_inst))) = 1);
END_REPEAT;
  WHERE
    WR1: NOT violation;
END_RULE;
```

```
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **product_definition** that has an **applied_classification_assignment** whose **assigned_class** is an entity with **name** attribute of value 'class and statutory designation' shall be referenced by exactly one instance of **property_definition** that has an **applied_classification_-assignment** whose **assigned_class** is an entity with **name** attribute of value 'regulation' through attribute **definition**.

5.2.4.64 product_definition_for_shipyard

The **product_definition_for_shipyard** rule specifies that an instance of **product_definition** with class id 'shipyard designation' is referenced by exactly one instance of **applied_organization_-assignment** via **items** whose attribute **role.name** has the value 'shipyard'.

EXPRESS specification:

```
*)
RULE product_definition_for_shipyard
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =
                   'shipyard designation');
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
    'SHIP_ARRANGEMENT_SCHEMA.APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
    'shipyard')) = 1);
  END_REPEAT;
  WHERE
    wr1: NOT violation;
END_RULE;

(*
```


Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **product_definition** that is referenced by an **applied_classification_assignment** of class 'shipyard designation', is referenced by exactly one **applied_organization_assignment** attribute **items**, where **applied_organization_assignment** attribute **role.name** = 'shipyard'.

5.2.4.65 product_definition_relationship_references_are_distinct

The **product_definition_relationship_references_are_distinct** rule specifies the instances that are related by the attributes **related_product_definition** and **relating_product_definition** of the relationship **product_definition_relationship** shall be different.

EXPRESS specification:

```
*)
RULE product_definition_relationship_references_are_distinct
FOR (product_definition_relationship);
  LOCAL
    cyclic_relationship: LOGICAL := FALSE;
  END_LOCAL;

  REPEAT i := 1 TO HIINDEX(product_definition_relationship)
        WHILE NOT cyclic_relationship;
    cyclic_relationship:=
product_definition_relationship[i].related_product_definition :=:
product_definition_relationship[i].relating_product_definition;
  END_REPEAT;

  WHERE
    wr1: NOT cyclic_relationship;
END_RULE;

(*
```

Argument definitions:

product_definition_relationship: the set of all instances of **product_definition_relationship** entities.

Formal propositions:

WR1: The entity instances referenced by attributes **related_product_definition** and **relating_-product_definition** instance of a **product_definition_relationship** must not be identical instances.

5.2.4.66 product_definition_relationship_with_identification_assignment

The **product_definition_relationship_with_identification_assignment** rule specifies a list of entities that require an identification. The identification is defined by the **applied_identification_-assignment** attribute.

EXPRESS specification:

```
*)
RULE product_definition_relationship_with_identification_assignment
```

```
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition_relationship := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := ['space arrangement relationship'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id in arg_list *)
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  (* get all instances of property_definition_representation *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
   t2_set := QUERY ( j <* t2_set |
  j.role.name = 'globally unambiguous identifier');
     violation := NOT (SIZEOF(T2_SET) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment**.

Formal propositions:

WR1: Every instance of **product_definition_relationship** that is referenced by an **applied_-classification_assignment** whose **assigned_class** has a **name** attribute of value 'space arrangement relationship' shall require an **applied_identification_assignment** to define the instance identifier.

5.2.4.67 product_definition_shape_for_deck_zone_design

The **product_definition_shape_for_deck_zone_design** rule specifies that an instance of **product_definition_shape** with class id 'deck zone design definition' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'deck zone design parameters'.

EXPRESS specification:

```
*)
RULE product_definition_shape_for_deck_zone_design
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
```



```
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition_shape := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'deck zone design
definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.name = 'deck zone design definition');

  (* get all instances of T1 that have class id 'deck zone design
definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of product_definition_shape in t1_set:
     get the property_definition_representation instances that are
referencing a product_definition_shape instance via definition, filter out
those property_definition_representation instances whose attribute name has
the value ‘deck zone design parameters’; check if their number equals 1
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
   t2_inst.name = 'deck zone design parameters')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **product_definition_shape** that is referenced by an **applied_classification_assignment** with **name** of ‘deck zone design definition’, is referenced by exactly one **property_definition_-representation.definition**, where **property_definition_representation.name** equals ‘deck zone design parameters’.

5.2.4.68 product_definition_shape_with_identification_assignment

The **product_definition_shape_with_identification_assignment** rule specifies a list of entities that require an identification. The identification is defined by the **applied_identification_assignment** attribute.

EXPRESS specification:

```
*)
RULE product_definition_shape_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
```

```
 c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
 t1_set: SET OF product_definition_shape := [];
 t2_set: SET OF applied_identification_assignment := [];
 arg_list: LIST OF STRING := ['design definition'];
 violation: LOGICAL := FALSE;
END_LOCAL;

(* get all classification_assignment instances with id in arg_list *)
REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
        i.assigned_class.NAME = arg_LIST[j]);
END_REPEAT;

(* get all instances of product_definition_shape that have class id *)
REPEAT i := 1 TO HIINDEX(c_a_set);
 REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
  t1_set := t1_set + c_a_set[i].items[j];
 END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
 t2_set := bag_to_set(USEDIN(t1_set[i],
    'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
 t2_set := QUERY ( j <* t2_set |
          j.role.name = 'globally unambiguous identifier');
 violation := NOT (SIZEOF(T2_SET) = 1);
END_REPEAT;
WHERE
wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment**.

Formal propositions:

WR1: Every instance of **product_definition_shape** that is referenced by an **applied_classification_-assignment** whose **assigned_class** has a **name** attribute of value 'design definition' shall require an **applied_identification_assignment** to define the instance identifier.

5.2.4.69 product_definition_with_date_freeboard_assigned

The **product_definition_with_date_freeboard_assigned** rule specifies that an instance of **product_definition** with class id 'freeboard characteristics' is referenced by exactly one instance of **applied_date_and_time_assignment** via **items** whose attribute **role** has the value 'date freeboard assigned'.

EXPRESS specification:

```
*)
RULE product_definition_with_date_freeboard_assigned
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_date_and_time_assignment := [];
    violation: LOGICAL := FALSE;
```



```
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
               i.assigned_class.name = 'freeboard characteristics');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
   t2_inst.role.name = 'date freeboard assigned')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **product_definition** that is referenced by an **applied_classification_assignment** with **name** of 'freeboard characteristics' is referenced by exactly one **applied_date_and_time_assignment.items**, where **applied_date_and_time_assignment.role** has a value of 'date freeboard assigned'.

5.2.4.70 product_definition_with_freeboard_assigned_by

The **product_definition_with_freeboard_assigned_by** rule specifies that an instance of **product_definition** with class id 'freeboard characteristics' is referenced by exactly one instance of **applied_organization_assignment** via **items** whose attribute **role** has the value 'freeboard assigned by'.

EXPRESS specification:

```
*)
RULE product_definition_with_freeboard_assigned_by
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
               i.assigned_class.name = 'freeboard characteristics');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
```

```
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
   'SHIP_ARRANGEMENT_SCHEMA.APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
      t2_inst.role.name = 'freeboard assigned by')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **product_definition** that is referenced by an **applied_classification_assignment** with **name** of 'freeboard characteristics' is referenced by exactly one **applied_organization_-assignment.items**, where **applied_organization_assignment.role** has a value of 'freeboard assigned by'.

5.2.4.71 product_definition_with_identification_assignment

The **product_definition_with_identification_assignment** rule specifies a list of entities that require an identification. The identification is defined by the **applied_identification_assignment** attribute.

EXPRESS specification:

```
*)
RULE product_definition_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := [ 'definition',
                                   'definable object'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances  *)
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  (* get all instances of product_definition that have class id *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;
```



```
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
      t2_set := QUERY ( j <* t2_set |
          j.role.name = 'globally unambiguous identifier');
violation := NOT (SIZEOF(T2_SET) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment**.

Formal propositions:

WR1: Every instance of **product_definition** that is referenced by an **applied_classification_-assignment** whose **assigned_class** has a **name** attribute of value 'definition' or ‘definable object’ shall require an **applied_identification_assignment** to define the instance identifier.

5.2.4.72 product_related_product_category_with_identification_assignment

The **product_related_product_category_with_identification_assignment** rule specifies a list of entities that require an identification. The identification is defined by the **applied_identification_-assignment** entity.

EXPRESS specification:

```
*)
RULE product_related_product_category_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_related_product_category := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := ['shiptype'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id  *)
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  (* get all instances that have class id *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
      t2_set := QUERY ( j <* t2_set |
```

```
              j.role.name = 'globally unambiguous identifier');
   violation := NOT (SIZEOF(T2_SET) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment**.

Formal propositions:

WR1: Every instance of **product_related_product_categor** that is referenced by an **applied_-classification_assignment** whose **assigned_class** has a **name** attribute of value 'shiptype' shall require an **applied_identification_assignment** to define the instance identifier.

5.2.4.73 product_with_identification_assignment

The **product_with_identification_assignment** rule specifies a list of entities that require an identification. The identification is defined by the **applied_identification_assignment** entity.

EXPRESS specification:

```
*)
RULE product_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := ['ship'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances *)
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
      t2_set := QUERY ( j <* t2_set |
                j.role.name = 'globally unambiguous identifier');
violation := NOT (SIZEOF(T2_SET) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
```



```
END_RULE;
```

(*

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment**.

Formal propositions:

WR1: Every instance of **product** that is referenced by an **applied_classification_assignment** whose **assigned_class** has a **name** attribute of value 'ship' shall require an **applied_identification_-assignment** to define the instance identifier.

5.2.4.74 property_definition_for_class_bulk_load_requirement_definition

The **property_definition_for_class_bulk_load_requirement_definition** rule specifies that an instance of **property_definition** with class id 'class bulk load requirement definition' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'class bulk load requirement definition parameters'.

EXPRESS specification:

```
*)
RULE property_definition_for_class_bulk_load_requirement_definition
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'class bulk load
requirement definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
       i.assigned_class.name = 'class bulk load requirement definition');

  (* get all instances of T1 that have class id 'class bulk load
requirement definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ‘class bulk load requirement definition parameters’; check if their
number equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.name = 'class bulk load requirement definition parameters')) = 1);
  END_REPEAT;

  WHERE
```

```
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **name** of ‘class bulk load requirement definition’, is referenced by exactly one **property_definition_representation.definition**, where **property_definition_representation.name** equals ‘class bulk load requirement definition parameters’.

5.2.4.75 property_definition_for_class_compartment_requirement_definition

The **property_definition_for_class_compartment_requirement_definition** rule specifies that an instance of **property_definition** with class id 'class compartment requirement definition' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'class compartment requirement definition parameters'.

EXPRESS specification:

```
*)
RULE property_definition_for_class_compartment_requirement_definition
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'class compartment
requirement definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
      i.assigned_class.name = 'class compartment requirement definition');

  (* get all instances of T1 that have class id 'class compartment
requirement definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ‘class compartment requirement definition parameters’; check if their
number equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
```



```
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
t2_inst.name = 'class compartment requirement definition parameters')) =
1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **name** of ‘class compartment requirement definition’, is referenced by exactly one **property_definition_representation.definition**, where **property_definition_representation.name** equals ‘class compartment requirement definition parameters’.

5.2.4.76 property_definition_for_class_deck_load_requirement_definition

The **property_definition_for_class_deck_load_requirement_definition** rule specifies that an instance of **property_definition** with class id 'class deck load requirement definition' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'class deck load requirement definition parameters'.

EXPRESS specification:

```
*)
RULE property_definition_for_class_deck_load_requirement_definition
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'class deck load
requirement definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
      i.assigned_class.name = 'class deck load requirement definition');

  (* get all instances of T1 that have class id 'class deck load
requirement definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ‘class deck load requirement definition parameters’; check if their
number equals 1.  *)
```

```
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
t2_inst.name = 'class deck load requirement definition parameters')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **name** of 'class deck load requirement definition', is referenced by exactly one **property_-definition_representation.definition**, where **property_definition_representation.name** equals 'class deck load requirement definition parameters'.

5.2.4.77 property_definition_for_class_notation

The **property_definition_for_class_notation** rule specifies that an instance of **property_definition** with class id 'class notation' is referenced by exactly one instance of **property_definition_-representation** via **definition** whose attribute **name** has the value 'class notation'.

EXPRESS specification:

```
*)
RULE property_definition_for_class_notation
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'class notation'
*)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'class notation');

  (* get all instances of T1 that have class id 'class notation' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value 'class notation'; check if their number equals 1.  *)
```



```
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.name = 'class notation')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **name** of 'class notation', is referenced by exactly one **property_definition_representation.definition** with **property_definition_representation.name** of 'class notation'.

5.2.4.78 property_definition_for_class_society

The **property_definition_for_class_society** rule specifies that an instance of **property_definition** with class id 'class notation' is referenced by exactly one instance of **applied_organization_assignment** via **items** whose attribute **role** has the value 'class society'.

EXPRESS specification:

```
*)
RULE property_definition_for_class_society
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'class notation'
*)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.name = 'class notation');

  (* get all instances of T1 that have class id 'class notation' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the applied_organization_assignment instances that are referencing
a property_definition instance via items, filter out those
applied_organization_assignment instances whose attribute role has the
value ‘class society’; check if their number equals 1
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
```

```
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.role.name = 'class society')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **name** of 'class notation' is referenced by exactly one **applied_organization_assignment.items**, where **applied_organization_assignment.role** is 'class society'.

5.2.4.79 property_definition_for_class_tank_requirement_definition

The **property_definition_for_class_tank_requirement_definition** rule specifies that an instance of **property_definition** with class id 'class tank requirement definition' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'class tank requirement definition parameters'.

EXPRESS specification:

```
*)
RULE property_definition_for_class_tank_requirement_definition
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'class tank
requirement definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
        i.assigned_class.name = 'class tank requirement definition');

  (* get all instances of T1 that have class id 'class tank requirement
definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
```



```
value ‘class tank requirement definition parameters’; check if their number
equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.name = 'class tank requirement definition parameters')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **name** of ‘class tank requirement definition’, is referenced by exactly one **property_definition_-representation.definition**, where **property_definition_representation.name** equals ‘class tank requirement definition parameters’.

5.2.4.80 property_definition_for_compartment_design_requirement

The **property_definition_for_compartment_design_requirement** rule specifies that an instance of **property_definition** with class id 'compartment design requirement' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'compartment design requirement parameters'.

EXPRESS specification:

```
*)
RULE property_definition_for_compartment_design_requirement
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'compartment
design requirement' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
      i.assigned_class.name = 'compartment design requirement');

  (* get all instances of T1 that have class id 'compartment design
requirement' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
```

```
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ‘compartment design requirement parameters’; check if their number
equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.name = 'compartment design requirement parameters')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **name** of ‘compartment design requirement’, is referenced by exactly one **property_definition_-representation.definition**, where **property_definition_representation.name** equals ‘compartment design requirement parameters’.

5.2.4.81 property_definition_for_compartment_function

The **property_definition_for_compartment_function** rule specifies that an instance of **property_definition** with class id 'compartment functional definition' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'compartment function parameters'.

EXPRESS specification:

```
*)
RULE property_definition_for_compartment_function
FOR(property_definition, property_definition_representation,
APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'compartment
functional definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
     i.assigned_class.name = 'compartment functional definition');

  (* get all instances of T1 that have class id 'compartment functional
definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
```

```
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ‘compartment function parameters’; check if their number equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.name = 'compartment function parameters')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **name** of ‘compartment functional definition’ is referenced by exactly one **property_definition_-representation.definition**, where **property_definition_representation.name** equals ‘compartment function parameters’.

5.2.4.82 property_definition_for_compensated_gross_tonnage

The **property_definition_has references_with_class_compensated_gross_tonnage** rule specifies that an instance of **property_definition** with class id 'tonnage definition' is referenced by exactly one instance of **property_definition_representation** with class id 'compensated gross tonnage' via **definition.**

EXPRESS specification:

```
*)
RULE property_definition_for_compensated_gross_tonnage
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'tonnage
definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
         i.assigned_class.name = 'tonnage definition');

  (* get all instances of property_definition that have class id 'tonnage
definition' *)
```

```
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose class id is 'compensated
gross tonnage'; check if their number equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 'compensated gross tonnage' IN WHICH_CLASS(t2_inst))) = 1);
END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **property_definition** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'tonnage definition' shall be referenced by exactly one instance of **property_definition_representation** that has an **applied_classification_-assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'compensated gross tonnage' through attribute **definition**.

5.2.4.83 property_definition_for_damage_stability_definition_requires_reference

The **property_definition_for_damage_stability_definition_requires_reference** rule specifies that a **property_definition** with a class id 'damage stability definition' is referenced by one or more **representation**s with a class id 'stability table' via a **property_definition_representation.**

EXPRESS specification:

```
*)
RULE property_definition_for_damage_stability_definition_requires_reference
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF representation := [];
    t3_set:  SET OF property_definition_representation := [];
    t4_set:  SET OF property_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'damage stability
definition' *)
```



```
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'damage stability definition');

  (* get all instances of property_definition that have class id 'damage
stability definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

(* get all classification_assignment instances with id 'stability table' *)
  c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'stability table');

  (* get all instances of representation that have class id 'stability
table' *)
  REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
      t2_set := t2_set + c2_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

(* get all property_definition_representation instances which have as the
.used_representation the representation instances that have class id
'stability table' *)
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + bag_to_set(USEDIN(t2_set[i],
          'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
         'USED_REPRESENTATION'));
  END_REPEAT;

(* get all property_definition instances which are the .definition of the
property_definition *)
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;

(* compare both lists with product_definition instances which have to be
identical *)
  violation := t1_set <> t4_set;

  WHERE
    wr1: NOT violation;

END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **property_definition** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'damage stability definition' shall be the **definition** of one or more instances of **property_definition_representation** whose **used_-representation** is an instance of **representation** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'stability table'.

5.2.4.84 property_definition_for_date_of_loading

The **property_definition_for_date_of_loading** rule specifies that an instance of **property_definition** with class id 'loading condition operating definition' is referenced by exactly one instance of **applied_date_and_time_assignment** via **items** whose attribute **role** has the value 'date of loading'.

EXPRESS specification:

```
*)
RULE property_definition_for_date_of_loading
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF applied_date_and_time_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'loading condition
operating definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
         i.ASSIGNED_CLASS.NAME = 'loading condition operating definition');

  (* get all instances of T1 that have class id 'loading condition
operating definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the applied_date_and_time_assignment instances that are
referencing a property_definition instance via items, filter out those
applied_date_and_time_assignment instances whose attribute role has the
value 'date of loading'; check if their number equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.role.name = 'date of loading')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **name** of 'loading condition operating definition', is referenced by exactly one **applied_date_-**

and_time_assignment.items, where **applied_date_and_time_assignment.role** equals 'date of loading'.

5.2.4.85 property_definition_for_deck_zone_function

The **property_definition_for_deck_zone_function** rule specifies that an instance of **property_definition** with class id 'deck zone functional definition' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'deck zone function parameters'.

EXPRESS specification:

```
*)
RULE property_definition_for_deck_zone_function
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'deck zone
functional definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
         i.assigned_class.name = 'deck zone functional definition');

  (* get all instances of T1 that have class id 'deck zone functional
definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ‘deck zone function parameters’; check if their number equals 1
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.name = 'deck zone function parameters')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **name** of 'deck zone functional definition', is referenced by exactly one **property_definition_-representation.definition**, where **property_definition_representation.name** equals 'deck zone function parameters'.

5.2.4.86 property_definition_for_gross_tonnage

The **property_definition_has references_with_class_gross_tonnage** rule specifies that an instance of **property_definition** with class id 'tonnage definition' is referenced by exactly one instance of **property_definition_representation** with class id 'gross tonnage' via **definition**.

EXPRESS specification:

```
*)
RULE property_definition_for_gross_tonnage
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'tonnage
definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
              i.assigned_class.name = 'tonnage definition');

  (* get all instances of property_definition that have class id 'tonnage
definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose class id is 'gross
tonnage'; check if their number equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 'gross tonnage' IN WHICH_CLASS(t2_inst))) = 1);
END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **property_definition** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'tonnage definition' shall be referenced by exactly one instance of **property_definition_representation** that has an **applied_classification_-assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'gross tonnage' through attribute **definition**.

5.2.4.87 property_definition_for_local_coordinate_system

The **property_definition_for_local_coordinate_system** rule specifies that an instance of **property_definition** with class id 'local co ordinate system' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'local coordinate system'.

EXPRESS specification:

```
*)
RULE property_definition_for_local_coordinate_system
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'local co ordinate
system' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'local co ordinate system');

  (* get all instances that have class id 'local co ordinate system' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set: get the
property_definition_representation instances that are referencing a
property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value 'local coordinate system'; check if their number equals 1
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
   t2_inst.name = 'local coordinate system')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **assigned_class.name** = 'local co ordinate system', is referenced by exactly one **property_definition_representation.definition**, where **property_definition_representation.name** equals 'local coordinate system'.

5.2.4.88 property_definition_for_local_coordinate_system_with_position

The **property_definition_for_local_coordinate_system_with_position** rule specifies that an instance of **property_definition** with class id 'local co ordinate system with position reference' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'local coordinate system with position reference'.

EXPRESS specification:

```
*)
RULE property_definition_for_local_coordinate_system_with_position
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =
                   'local co ordinate system with position reference');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],

'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name =
      'local coordinate system with position reference')) = 1);
  END_REPEAT;
  WHERE
    WR1: NOT violation;
END_RULE;

(*
```


Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **assigned_class.name** = 'local co ordinate system with position reference', is referenced by exactly one **property_definition_representation** attribute **definition**, where **property_definition_-representation** attribute **name** equals 'local coordinate system with position reference'.

5.2.4.89 property_definition_for_net_tonnage

The **property_definition_for_net_tonnage** rule specifies that an instance of **property_definition** with class id 'tonnage definition' is referenced by exactly one instance of **property_-definition_representation** with class id 'net tonnage' via **definition**.

EXPRESS specification:

```
*)
RULE property_definition_for_net_tonnage
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'tonnage
definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'tonnage definition');

  (* get all instances of property_definition that have class id 'tonnage
definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose class id is 'net
tonnage'; check if their number equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
   'net tonnage' IN WHICH_CLASS(t2_inst))) = 1);
END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **property_definition** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'tonnage definition' shall be referenced by exactly one instance of **property_definition_representation** that has an **applied_classification_-assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'net tonnage' through attribute **definition**.

5.2.4.90 property_definition_for_stability_definition_requires_reference

The **property_definition_for_stability_definition_requires_reference** rule specifies that a **property_definition** with a class id 'stability definition' is referenced by one or more **representation**s with a class id 'stability table' via a **property_definition_representation.**

EXPRESS specification:

```
*)
RULE property_definition_for_stability_definition_requires_reference
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF representation := [];
    t3_set:  SET OF property_definition_representation := [];
    t4_set:  SET OF property_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'stability
definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'stability definition');

  (* get all instances of property_definition that have class id 'stability
definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

(* get all classification_assignment instances with id 'stability table' *)
  c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'stability table');

  (* get all instances of representation that have class id 'stability
table' *)
  REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
      t2_set := t2_set + c2_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

(* get all property_definition_representation instances which have as the
```



```
.used_representation the representation instances that have class id
'stability table' *)
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + bag_to_set(USEDIN(t2_set[i],
            'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'));
  END_REPEAT;

(* get all property_definition instances which are the .definition of the
property_definition *)
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;

(* compare both lists with product_definition instances which have to be
identical *)
  violation := t1_set <> t4_set;

  WHERE
    wr1: NOT violation;

END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **property_definition** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'stability definition' shall be the **definition** of one or more instances of **property_definition_representation** whose **used_representation** is an instance of **representation** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'stability table'.

5.2.4.91 property_definition_for_tonnage_definition

The **property_definition_for_tonnage_definition** rule specifies that an instance of **property_-definition** with class id 'tonnage definition' is referenced by exactly one instance of **property_-definition_representation** via **definition** whose attribute **name** has the value 'tonnage definition parameters'.

EXPRESS specification:

```
*)
RULE property_definition_for_tonnage_definition
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'tonnage
definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
```

```
                    i.assigned_class.name = 'tonnage definition');

  (* get all instances of T1 that have class id 'tonnage definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ‘tonnage definition parameters’; check if their number equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.name = 'tonnage definition parameters')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **assigned_class.name** of ‘tonnage definition’, is referenced by exactly one **property_definition_-representation.definition**, where **property_definition_representation.name** equals ‘tonnage definition parameters’.

5.2.4.92 property_definition_for_zone_function

The **property_definition_for_zone_function** rule specifies that an instance of **property_definition** with class id 'zone functional definition' is referenced by exactly one instance of **property_-definition_representation** via **definition** whose attribute **name** has the value 'zone function parameters'.

EXPRESS specification:

```
*)
RULE property_definition_for_zone_function
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;
```



```
  (* get all classification_assignment instances with id 'zone functional
definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'zone functional definition');

  (* get all instances of T1 that have class id 'zone functional
definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ‘zone function parameters’; check if their number equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.name = 'zone function parameters')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **assigned_class.name** of ‘zone functional definition’, is referenced by exactly one **property_definition_representation.definition**, where p**roperty_definition_representation.name** equals ‘zone function parameters’.

5.2.4.93 property_definition_has_references_with_name_loadline

The **property_definition_has_references_with_name_loadline** rule specifies that an instance of **property_definition** with class id 'loadline' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'loadline'.

EXPRESS specification:

```
*)
RULE property_definition_has_references_with_name_loadline
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;
```

```
  (* get all classification_assignment instances with id 'loadline' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.name = 'loadline');

  (* get all instances of T1 that have class id 'loadline' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ‘loadline’; check if their number equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.name = 'loadline')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **assigned_class.name** of 'loadline' is referenced by exactly one **property_definition_representation.definition**, where **property_definition_representation.name** has a value of 'loadline'.

5.2.4.94 property_definition_representation_for_date_of_measurement

The **property_definition_representation_for_date_of_measurement** rule specifies that an instance of **property_definition_representation** with class id 'tonnage measurement' is referenced by exactly one instance of **applied_date_and_time_assignment** via **items** whose attribute **role** has the value 'date of measurement'.

EXPRESS specification:

```
*)
RULE property_definition_representation_for_date_of_measurement
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition_representation := [];
    t2_set:  SET OF applied_date_and_time_assignment := [];
    violation: LOGICAL := FALSE;
```



```
  END_LOCAL;

  (* get all classification_assignment instances with id 'tonnage
measurement' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'tonnage measurement');

  (* get all instances of T1 that have class id 'tonnage measurement' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition_representation in t1_set:
     get the applied_date_and_time_assignment instances that are
referencing a property_definition_representation instance via items,
     filter out those applied_date_and_time_assignment instances whose
attribute role has the value 'date of measurement'; check if their number
equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
'date of  measurement')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition_representation** that is referenced by an **applied_classification_-assignment** of class 'tonnage measurement', is referenced by exactly one **applied_date_and_time_-assignment.items**, where **applied_date_and_time_assignment.role** equals 'date of measurement'.

5.2.4.95 property_definition_representation_for_gross_tonnage

The **property_definition_representation_for_gross_tonnage** rule specifies that an instance of **property_definition_representation** with class id 'gross tonnage' is referenced by exactly one instance of **applied_date_and_time_assignment** via **items** whose attribute **role** has the value 'date of measurement'.

EXPRESS specification:

```
*)
RULE property_definition_representation_for_gross_tonnage
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition_representation := [];
    t2_set:  SET OF applied_date_and_time_assignment := [];
    violation: LOGICAL := FALSE;
```

```
  END_LOCAL;

  (* get all classification_assignment instances with id 'gross tonnage' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'gross tonnage');

  (* get all instances of T1 that have class id 'gross tonnage' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition_representation in t1_set:
     get the applied_date_and_time_assignment instances that are
referencing a property_definition_representation instance via items, filter
out those applied_date_and_time_assignment instances whose attribute role
has the value 'date of measurement'; check if their number equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
      'SHIP_ARRANGEMENT_SCHEMA.APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
'date of measurement')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition_representation** that is referenced by an **applied_classification_-assignment** of class 'gross tonnage', is referenced by exactly one **applied_date_and_time_-assignment.items**, where **applied_date_and_time_assignment.role** equals 'date of measurement'.

5.2.4.96 property_definition_representation_for_net_tonnage

The **property_definition_representation_for_net_tonnage** rule specifies that an instance of **property_definition_representation** with class id 'net tonnage' is referenced by exactly one instance of **applied_date_and_time_assignment** via **items** whose attribute **role** has the value 'date of measurement'.

EXPRESS specification:

```
*)
RULE property_definition_representation_for_net_tonnage
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition_representation := [];
    t2_set:  SET OF applied_date_and_time_assignment := [];
```

```
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'net tonnage' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'net tonnage');

  (* get all instances of T1 that have class id 'net tonnage' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition_representation in t1_set:
     get the applied_date_and_time_assignment instances that are
referencing a property_definition_representation instance via items,
filter out those applied_date_and_time_assignment instances whose attribute
role has the value 'date of measurement'; check if their number equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
'date of  measurement')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition_representation** that is referenced by an **applied_classification_-assignment** of class 'net tonnage', is referenced by exactly one **applied_date_and_time_-assignment.items**, where **applied_date_and_time_assignment.role** equals 'date of measurement'.

5.2.4.97 property_definition_with_identification_assignment

The **property_definition_with_identification_assignment** rule specifies a list of entities that require an identification. The identification is defined by the **applied_identification_assignment** entity.

EXPRESS specification:

```
*)
RULE property_definition_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING :=
['cargo bay definition', 'compartment functional definition',
```

```
'deck zone functional definition', 'design requirement',
'loading condition definition', 'local co ordinate system',
'spacing table', 'stability definition', 'tonnage definition',
'zone functional definition'];
        violation: LOGICAL := FALSE;
  END_LOCAL;


    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
      t2_set := QUERY ( j <* t2_set |
                j.role.name = 'globally unambiguous identifier');
violation := NOT (SIZEOF(T2_SET) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;
```

(*

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment**.

Formal propositions:

WR1: Every instance of **property_definition** that is referenced by an **applied_classification_-assignment** whose **assigned_class** has a **name** attribute of value of 'cargo bay definition', 'compartment functional definition', 'deck zone functional definition', 'design requirement', 'loading condition definition', 'local co ordinate system', 'spacing table', 'stability definition', 'tonnage definition', or 'zone functional definition' shall require an **applied_identification_assignment** to define the instance identifier.

5.2.4.98 property_definition_with_lightship_weight_item

The **property_definition_with_lightship_weight_item** rule specifies that an instance of **property_definition** with class id 'lightship weight item' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'lightship weight item'.

EXPRESS specification:

```
*)
RULE property_definition_with_lightship_weight_item
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
```



```
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'lightship weight
item' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.name = 'lightship weight item');

  (* get all instances of T1 that have class id 'lightship weight item' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ‘lightship weight item’; check if their number equals 1
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name =
'lightship weight item')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** of class 'lightship weight item', is referenced by exactly one **property_definition_-representation.definition**, where **property_definition_representation.name** has a value of 'lightship weight item'.

5.2.4.99 property_definition_with_weight_and_centre_of_gravity

The **property_definition_with_weight_and_centre_of_gravity** rule specifies that an instance of **property_definition** with class id 'weight and centre of gravity' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'weight and centre of gravity'.

EXPRESS specification:

```
*)
RULE property_definition_with_weight_and_centre_of_gravity
```

```
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'weight and centre
of gravity' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.name = 'weight and centre of gravity');

  (* get all instances of T1 that have class id 'weight and centre of
gravity' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of property_definition in t1_set:
     get the property_definition_representation instances that are
referencing a property_definition instance via definition, filter out those
property_definition_representation instances whose attribute name has the
value ‘weight and centre of gravity’; check if their number equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name =
'weight and centre of gravity')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** of class 'weight and centre of gravity' is referenced by exactly one **property_definition_-representation.definition**, where **property_definition_representation.name** has a value of 'weight and centre of gravity'.

5.2.4.100 representation_for_absolute_cargo

The **representation_for_absolute_cargo** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'absolute cargo position parameters'.

EXPRESS specification:


```
*)
RULE representation_for_absolute_cargo
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['position', 'orientation'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'absolute
cargo position parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'absolute cargo position parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'absolute cargo position parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'position' or 'orientation'.

5.2.4.101 representation_for_adjacent_space_surface_area

The **representation_for_adjacent_space_surface_area** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'adjacent space surface area parameters'.

EXPRESS specification:

```
*)
RULE representation_for_adjacent_space_surface_area
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['surface area'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used by a
property_definition_representation with name equal to 'adjacent space
surface area parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'adjacent space surface area parameters')
                )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘adjacent space surface area parameters’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of 'surface area'.

5.2.4.102 representation_for_bulk_cargo

The **representation_for_bulk_cargo** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_-representation** whose **name** attribute has a value 'bulk cargo parameters'.

EXPRESS specification:

```
*)
RULE representation_for_bulk_cargo
```



```
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['natural angle of repose',
 'pollution code', 'required carriage temperature', 'type of',
 'un type code'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used by a
property_definition_representation with name equal to 'bulk cargo
parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'bulk cargo parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'bulk cargo parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'natural angle of repose', 'pollution code', 'required carriage temperature', 'type of', or 'un type code'.

5.2.4.103 representation_for_bulk_cargo_assignment

The **representation_for_bulk_cargo_assignment** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'bulk cargo assignment parameters'.

EXPRESS specification:

```
*)
RULE representation_for_bulk_cargo_assignment
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['actual angle of repose',
'assignment context', 'cargo height', 'cargo identifier', 'trimmed'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'bulk cargo
assignment parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))
     | (temp_prop_def_rep.name = 'bulk cargo assignment parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'bulk cargo assignment parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'actual angle of repose', 'assignment context', 'cargo height', 'cargo identifier', or 'trimmed'.

5.2.4.104 representation_for_capacity_properties

The **representation_for_capacity_properties** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'capacity properties'.

EXPRESS specification:


```
*)
RULE representation_for_capacity_properties
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['capacity level origin',
'capacity centre', 'capacity level', 'capacity trim angle',
'capacity heel angle', 'capacity volume', 'capacity context'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'capacity
properties' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))
       | (temp_prop_def_rep.name = 'capacity properties')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘capacity properties’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_-item** with a **name** of either 'capacity level origin', 'capacity centre', 'capacity level', 'capacity trim angle', 'capacity heel angle', 'capacity volume', or 'capacity context'.

5.2.4.105 representation_for_cargo_compartment_property

The **representation_for_cargo_compartment_property** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'cargo compartment property'.

EXPRESS specification:

```
*)
RULE representation_for_cargo_compartment_property
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'design stowage density'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'cargo
compartment property' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.name = 'cargo compartment property')
                  )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                      rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘cargo compartment property’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'design stowage density'.

5.2.4.106 representation_for_cargo_footprint

The **representation_for_cargo_footprint** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_-definition_representation** whose **name** attribute has a value 'cargo footprint'.

EXPRESS specification:


```
*)
RULE representation_for_cargo_footprint
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['contact material', 'shape',
 'transferred mass'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'cargo
footprint' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'cargo footprint')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'cargo footprint', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'contact material', 'shape', or 'transferred mass'.

5.2.4.107 representation_for_class_and_statutory_designation

The **representation_for_class_and_statutory_designation** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'class and statutory designation'.

EXPRESS specification:

```
*)
RULE representation_for_class_and_statutory_designation
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['class number'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'class and statutory designation')
                   )
             ) > 0
          );

  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **property_definition_representation** with name of 'class and statutory designation', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a name of 'class number'.

5.2.4.108 representation_for_class_bulk_load_requirement_definition

The **representation_for_class_bulk_load_requirement_definition** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'class bulk load requirement definition parameters'.

EXPRESS specification:

```
*)
RULE representation_for_class_bulk_load_requirement_definition
FOR (representation);
  LOCAL
```



```
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['ambient temperature', 'angle of repose',
'bulk cargo mass', 'cargo density', 'cargo height', 'coating',
'damage waterline', 'max pressure', 'max temperature', 'min pressure',
'min temperature', 'permeability', 'top of hatch'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'class bulk
load requirement definition parameters' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
 'class bulk load requirement definition parameters')
                  )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘class bulk load requirement definition parameters’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'ambient temperature', 'angle of repose', 'bulk cargo mass', 'cargo density', 'cargo height', 'coating', 'damage waterline', 'max pressure', 'max temperature', 'min pressure', 'min temperature', 'permeability', or 'top of hatch'.

5.2.4.109 representation_for_class_compartment_requirement_definition

The **representation_for_class_compartment_requirement_definition** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'class compartment requirement definition parameters'.

EXPRESS specification:

```
*)
RULE representation_for_class_compartment_requirement_definition
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['ambient temperature', 'cargo density',
'cargo height', 'coating', 'damage waterline', 'max pressure',
'max temperature', 'min pressure', 'min temperature'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'class
compartment requirement definition parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
 'class compartment requirement definition parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘class compartment requirement definition parameters’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'ambient temperature', 'cargo density', 'cargo height', 'coating', 'damage waterline', 'max pressure', 'max temperature', 'min pressure', or 'min temperature'.

5.2.4.110 representation_for_class_notation

The **representation_for_class_notation** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value

given in the list if the **representation** is the **used_representation** in a **property_definition_-representation** whose **name** attribute has a value 'class notation'.

EXPRESS specification:

```
*)
RULE representation_for_class_notation
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['service area',
'approval required for oil cargo',
'approval required for loading unloading aground',
'approval required for unloading grabs'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'class
notation' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'class notation')
                )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'class notation', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_-item** with a **name** of either 'service area', 'approval required for oil cargo', 'approval required for loading unloading aground', or 'approval required for unloading grabs'.

5.2.4.111 representation_for_class_parameters

The **representation_for_class_parameters** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'class parameters'.

EXPRESS specification:

```
*)
RULE representation_for_class_parameters
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['length class', 'length solas',
'scantlings draught', 'block coefficient class',
 'design speed ahead','design speed astern'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'class
parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'class parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'class parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_-**

item with a **name** of either 'length class', 'length solas', 'scantlings draught', 'block coefficient class', 'design speed ahead', or 'design speed astern'.

5.2.4.112 representation_for_class_tank_requirement_definition

The **representation_for_class_tank_requirement_definition** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'class tank requirement definition parameters'.

EXPRESS specification:

```
*)
RULE representation_for_class_tank_requirement_definition
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['ambient temperature', 'cargo density',
'cargo height', 'coating', 'damage waterline', 'max pressure',
'max temperature', 'min pressure', 'min temperature', 'overflow height',
'partial filling', 'pressure relief setting'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'class tank
requirement definition parameters' *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
 'class tank requirement definition parameters')
                 )
              ) > 0
           );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘class tank requirement definition parameters’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'ambient temperature', 'cargo density', 'cargo height', 'coating', 'damage waterline', 'max pressure', 'max temperature', 'min pressure', 'min temperature', 'overflow height', 'partial filling',or 'pressure relief setting'.

5.2.4.113 representation_for_coating

The **representation_for_coating** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_-representation** whose **name** attribute has a value 'coating parameters'.

EXPRESS specification:

```
*)
RULE representation_for_coating
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['dry film thickness', 'number of coats',
'surface preparation'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'coating
parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'coating parameters')
                   )
                ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                      rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘coating parameters’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'dry film thickness', 'number of coats', or 'surface preparation'.

5.2.4.114 representation_for_coating_level

The **representation_for_coating_level** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'coating level'.

EXPRESS specification:

```
*)
RULE representation_for_coating_level
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['lower extent', 'upper extent'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'coating
level' *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'coating level')
                    )
              ) > 0
           );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘coating level’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_-item** with a **name** of either 'lower extent' or 'upper extent'.

5.2.4.115 representation_for_compartment_abbreviated_name

The **representation_for_compartment_abbreviated_name** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment abbreviated_name'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_abbreviated_name
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'name'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment abbreviated name' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment abbreviated name')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment abbreviated name’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a name of either 'context' or 'name'.

5.2.4.116 representation_for_compartment_acceleration

The **representation_for_compartment_acceleration** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment acceleration'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_acceleration
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'acceleration g force'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment acceleration' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment acceleration')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment acceleration’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'acceleration g force'.

5.2.4.117 representation_for_compartment_access_authorization

The **representation_for_compartment_access_authorization** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment access authorization'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_access_authorization
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context',
 'authorization classification'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment access authorization' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment access authorization')
                )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment access authorization’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'authorization classification'.

5.2.4.118 representation_for_compartment_air_circulation_rate

The **representation_for_compartment_air_circulation_rate** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment air circulation rate'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_air_circulation_rate
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'air circulation rate'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment air circulation rate' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment air circulation rate')
                  )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment air circulation rate’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'air circulation rate'.

5.2.4.119 representation_for_compartment_cargo_assignment

The **representation_for_compartment_cargo_assignment** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment cargo assignment parameters'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_cargo_assignment
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['assignment context',
  'cargo identifier'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment cargo assignment parameters' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment cargo assignment parameters')
                  )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'compartment cargo assignment parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'assignment context' or 'cargo identifier'.

5.2.4.120 representation_for_compartment_coating

The **representation_for_compartment_coating** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment coating'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_coating
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment coating' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment coating')
                  )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment coating’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of 'context'.

5.2.4.121 representation_for_compartment_design_requirement

The **representation_for_compartment_design_requirement** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment design requirement parameters'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_design_requirement
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['description', 'requirement type'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment design requirement parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment design requirement parameters')
                )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment design requirement parameters’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'description' or 'requirement type'.

5.2.4.122 representation_for_compartment_function

The **representation_for_compartment_function** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment function parameters'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_function
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['used for'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment function parameters' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment function parameters')
                  )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment function parameters’ shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of 'used for'.

5.2.4.123 representation_for_compartment_group

The **representation_for_compartment_group** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment group parameters'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_group
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['tonnage volume'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment group parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment group parameters')
                )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'compartment group parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of 'tonnage volume'.

5.2.4.124 representation_for_compartment_horizontal_cross_sectional_area

The **representation_for_compartment_horizontal_cross_sectional_area** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment horizontal cross sectional area property'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_horizontal_cross_sectional_area
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context',
 'horizontal cross sectional area'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment horizontal cross sectional area property' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
 'compartment horizontal cross sectional area property')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment horizontal cross sectional area property’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'horizontal cross sectional area'.

5.2.4.125 representation_for_compartment_illumination

The **representation_for_compartment_illumination** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment illumination'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_illumination
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'illumination value'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment illumination' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment illumination')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment illumination’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'illumination value'.

5.2.4.126 representation_for_compartment_insulation

The **representation_for_compartment_insulation** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment insulation'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_insulation
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'insulation category'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment insulation' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment insulation')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment insulation’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'insulation category'.

5.2.4.127 representation_for_compartment_noise_category

The **representation_for_compartment_noise_category** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment noise category'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_noise_category
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'noise category'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment noise category' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment noise category')
                  )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment noise category’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'noise category'.

5.2.4.128 representation_for_compartment_nuclear_classification

The **representation_for_compartment_nuclear_classification** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment nuclear classification'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_nuclear_classification
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'nuclear classification'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment nuclear classification' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment nuclear classification')
                  )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment nuclear classification’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'nuclear classification'.

5.2.4.129 representation_for_compartment_occupancy

The **representation_for_compartment_occupancy** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment occupancy'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_occupancy
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'occupancy'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment occupancy' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment occupancy')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment occupancy’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'occupancy'.

5.2.4.130 representation_for_compartment_safety_class

The **representation_for_compartment_safety_class** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment safety class'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_safety_class
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'safety category'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment safety class' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment safety class')
                  )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment safety class’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'safety category'.

5.2.4.131 representation_for_compartment_security_classification

The **representation_for_compartment_security_classification** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment security classification'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_security_classification
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'security classification'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment security classification' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment security classification')
                )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment security classification’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'security classification'.

5.2.4.132 representation_for_compartment_stiffened_surface_area_property

The **representation_for_compartment_stiffened_surface_area_property** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment stiffened surface area property'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_stiffened_surface_area_property
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'stiffened surface area'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment stiffened surface area property' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
 'compartment stiffened surface area property')
                        )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment stiffened surface area property’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'stiffened surface area'.

5.2.4.133 representation_for_compartment_tightness

The **representation_for_compartment_tightness** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment tightness'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_tightness
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context',
 'required bulkhead tightness'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment tightness' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment tightness')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment tightness’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'required bulkhead tightness'.

5.2.4.134 representation_for_compartment_unstiffened_surface_area_property

The **representation_for_compartment_unstiffened_surface_area_property** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment unstiffened surface area property'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_unstiffened_surface_area_property
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'unstiffened surface area'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment unstiffened surface area property' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
 'compartment unstiffened surface area property')
                  )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment unstiffened surface area property’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'unstiffened surface area'.

5.2.4.135 representation_for_compartment_vertical_longitudinal_sectional_area

The **representation_for_compartment_vertical_longitudinal_sectional_area** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment vertical longitudinal sectional area property'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_vertical_longitudinal_sectional_area
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context',
 'vertical longitudinal cross sectional area'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment vertical longitudinal sectional area property' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
 'compartment vertical longitudinal sectional area property')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment vertical longitudinal sectional area property’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'vertical longitudinal cross sectional area'.

5.2.4.136 representation_for_compartment_vertical_transverse_sectional_area

The **representation_for_compartment_vertical_transverse_sectional_area** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment vertical transverse sectional area property'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_vertical_transverse_sectional_area
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context',
 'vertical transverse cross sectional area'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment vertical transverse sectional area property' *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
 'compartment vertical transverse sectional area property')
                    )
              ) > 0
           );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment vertical transverse sectional area property’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'vertical transverse cross sectional area'.

5.2.4.137 representation_for_compartment_volume_permeability_property

The **representation_for_compartment_volume_permeability_property** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment volume permeability property'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_volume_permeability_property
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'permeability'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment volume permeability property' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
 'compartment volume permeability property')
                      )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```


Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment volume permeability property’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'permeability'.

5.2.4.138 representation_for_compartment_volume_property

The **representation_for_compartment_volume_property** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment volume property'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_volume_property
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['centre of volume', 'context', 'volume'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment volume property' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
 'compartment volume property')
                      )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment volume property’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'centre of volume', 'context', or 'volume'.

5.2.4.139 representation_for_compartment_ziplist_number

The **representation_for_compartment_ziplist_number** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment ziplist number'.

EXPRESS specification:

```
*)
RULE representation_for_compartment_ziplist_number
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'department ziplist number',
'division ziplist number'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment ziplist number' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment ziplist number')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```


Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘compartment ziplist number’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context', 'department ziplist number', or 'division ziplist number'.

5.2.4.140 representation_for_compensated_gross_tonnage

The **representation_for_compensated_gross_tonnage** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compensated gross tonnage'.

EXPRESS specification:

```
*)
RULE representation_for_compensated_gross_tonnage
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['compensation factor', 'tonnage value'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compensated gross tonnage' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compensated gross tonnage')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'compensated gross tonnage', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'compensation factor' or 'tonnage value'.

5.2.4.141 representation_for_corrosion_control_coating

The **representation_for_corrosion_control_coating** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'corrosion control coating parameters'.

EXPRESS specification:

```
*)
RULE representation_for_corrosion_control_coating
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['applicability', 'dry film thickness',
'number of coats', 'surface preparation', 'type of'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'corrosion
control coating parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'corrosion control coating parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```


Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘corrosion control coating parameters’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'applicability', 'dry film thickness', 'number of coats', 'surface preparation', or 'type of'.

5.2.4.142 representation_for_corrosion_protection

The **representation_for_corrosion_protection** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'corrosion protection'.

EXPRESS specification:

```
*)
RULE representation_for_corrosion_protection
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['cathodic protection'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'corrosion
protection' *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'corrosion protection')
                    )
              ) > 0
           );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘corrosion protection’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of 'cathodic protection'.

5.2.4.143 representation_for_damage_case

The **representation_for_damage_case** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'damage case parameters'.

EXPRESS specification:

```
*)
RULE representation_for_damage_case
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['damage cause',
 'relative damage position'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'damage
case parameters' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'damage case parameters')
                  )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'damage case parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'damage cause' or 'relative damage position'.

5.2.4.144 representation_for_damage_position

The **representation_for_damage_position** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'damage position parameters'.

EXPRESS specification:

```
*)
RULE representation_for_damage_position
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['centre of damage', 'position accuracy'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'damage
position parameters' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'damage position parameters')
                  )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'damage position parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'centre of damage' or 'position accuracy'.

5.2.4.145 representation_for_dangerous_goods_code

The **representation_for_dangerous_goods_code** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'dangerous goods code parameters'.

EXPRESS specification:

```
*)
RULE representation_for_dangerous_goods_code
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['class', 'subsidiary risks'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'dangerous
goods code parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'dangerous goods code parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'dangerous goods code parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'class' or 'subsidiary risks'.

5.2.4.146 representation_for_deck_cargo_assignment

The **representation_for_deck_cargo_assignment** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'deck cargo assignment parameters'.

EXPRESS specification:

```
*)
RULE representation_for_deck_cargo_assignment
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['assignment context',
  'cargo identifier'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'deck cargo
assignment parameters' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'deck cargo assignment parameters')
                  )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'deck cargo assignment parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'assignment context' or 'cargo identifier'.

5.2.4.147 representation_for_deck_zone_function

The **representation_for_deck_zone_function** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'deck zone function parameters'.

EXPRESS specification:

```
*)
RULE representation_for_deck_zone_function
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['used for'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'deck zone
function parameters' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'deck zone function parameters')
                  )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘deck zone function parameters’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of 'used for'.

5.2.4.148 representation_for_dry_cargo

The **representation_for_dry_cargo** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_-representation** whose **name** attribute has a value 'dry cargo parameters'.

EXPRESS specification:

```
*)
RULE representation_for_dry_cargo
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['pollution code', 'un type code'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'dry cargo
parameters' *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'dry cargo parameters')
                    )
              ) > 0
           );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'dry cargo parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_-item** with a **name** of either 'pollution code', or 'un type code'.

5.2.4.149 representation_for_fire_safe_coating

The **representation_for_fire_safe_coating** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'fire safe coating parameters'.

EXPRESS specification:

```
*)
RULE representation_for_fire_safe_coating
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['dry film thickness', 'low flame spread',
'nitro cellulose based', 'number of coats', 'surface preparation'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'fire safe
coating parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'fire safe coating parameters')
                )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘fire safe coating parameters’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'dry film thickness', 'low flame spread', 'nitro cellulose based', 'number of coats', or 'surface preparation'.

5.2.4.150 representation_for_freeboard_characteristics

The **representation_for_freeboard_characteristics** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'freeboard characteristics'.

EXPRESS specification:

```
*)
RULE representation_for_freeboard_characteristics
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['assigned code','freeboard'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'freeboard
characteristics' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'freeboard characteristics')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'freeboard characteristics', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a name of ‘assigned code’ or 'freeboard'.

5.2.4.151 representation_for_gaseous_cargo

The **representation_for_gaseous_cargo** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_-representation** whose **name** attribute has a value 'gaseous cargo parameters'.

EXPRESS specification:

```
*)
RULE representation_for_gaseous_cargo
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := [‘cargo type’, 'carried in liquid state',
'pollution code', 'un type code'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'gaseous
cargo parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'gaseous cargo parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'gaseous cargo parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'cargo type', 'carried in liquid state', 'pollution code', or 'un type code'.

5.2.4.152 representation_for_gaseous_cargo_assignment

The **representation_for_gaseous_cargo_assignment** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'gaseous cargo assignment parameters'.

EXPRESS specification:

```
*)
RULE representation_for_gaseous_cargo_assignment
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['assignment context',
  'cargo identifier'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'gaseous
cargo assignment parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'gaseous cargo assignment parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'gaseous cargo assignment parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'assignment context' or 'cargo identifier'.

5.2.4.153 representation_for_global_axis_placement

The **representation_for_global_axis_placement** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'global axis placement'.

EXPRESS specification:

```
*)
RULE representation_for_global_axis_placement
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['global axes and origin',
 'after perpendicular offset', 'orientation'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'global axis placement')
                     )
               ) > 0
            );

  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'global axis placement' shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'global axes and origin', 'after perpendicular offset', or 'orientation'.

5.2.4.154 representation_for_gross_tonnage

The **representation_for_gross_tonnage** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value

given in the list if the **representation** is the **used_representation** in a **property_definition_-representation** whose **name** attribute has a value 'gross tonnage'.

EXPRESS specification:

```
*)
RULE representation_for_gross_tonnage
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['overdeck tonnage', 'tonnage value',
'underdeck tonnage'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'gross
tonnage' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'gross tonnage')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'gross tonnage', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'overdeck tonnage', 'tonnage value', or 'underdeck tonnage'.

5.2.4.155 representation_for_lightship_definition

The **representation_for_lightship_definition** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a

property_definition_representation whose **name** attribute has a value 'lightship definition parameters'.

EXPRESS specification:

```
*)
RULE representation_for_lightship_definition
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['lightship weight',
 'lightship centre of gravity'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'lightship
definition parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'lightship definition parameters')
                )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'lightship definition parameters' shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'lightship weight', or 'lightship centre of gravity'.

5.2.4.156 representation_for_lightship_weight_item

The **representation_for_lightship_weight_item** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name**

attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'lightship weight item'.

EXPRESS specification:

```
*)
RULE representation_for_lightship_weight_item
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['aft weight extent','fwd weight extent'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'lightship
weight item' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'lightship weight item')
                )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'lightship weight item' shall use a **representation** that has exactly one item in its set of **items** that shall be of type **representation_item** with a **name** of either 'aft weight extent', or 'fwd weight extent'.

5.2.4.157 representation_for_liquid_cargo

The **representation_for_liquid_cargo** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_-representation** whose **name** attribute has a value 'liquid cargo parameters'.

EXPRESS specification:

```
*)
RULE representation_for_liquid_cargo
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['cargo type', 'pollution code',
 'un type code'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'liquid
cargo parameters' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'liquid cargo parameters')
                     )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'liquid cargo parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_-item** with a **name** of either 'cargo type', 'pollution code', or 'un type code'.

5.2.4.158 representation_for_liquid_cargo_assignment

The **representation_for_liquid_cargo_assignment** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'liquid cargo assignment parameters'.

EXPRESS specification:

```
*)
RULE representation_for_liquid_cargo_assignment
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['assignment context', 'cargo height',
'cargo identifier'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'liquid
cargo assignment parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'liquid cargo assignment parameters')
                )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'liquid cargo assignment parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'assignment context', 'cargo height', or 'cargo identifier'.

5.2.4.159 representation_for_loading_condition_design_definition

The **representation_for_loading_condition_design_definition** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'loading condition design definition parameters'.

EXPRESS specification:

```
*)
RULE representation_for_loading_condition_design_definition
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['type of'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'loading
condition design definition parameters' *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
 'loading condition design definition parameters')
                  )
              ) > 0
           );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'loading condition design definition parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'type of'.

5.2.4.160 representation_for_loading_condition_operating_definition

The **representation_for_loading_condition_operating_definition** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'loading condition operating definition parameters'.

EXPRESS specification:

```
*)
RULE representation_for_loading_condition_operating_definition
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['type of'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'loading
condition operating definition parameters' *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
 'loading condition operating definition parameters')
                 )
              ) > 0
           );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'loading condition operating definition parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of 'type of'.

5.2.4.161 representation_for_loadline

The **representation_for_loadline** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_-representation** whose **name** attribute has a value 'loadline'.

EXPRESS specification:

```
*)
RULE representation_for_loadline
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['load line length',
'load line depth', 'load line displacement',
'load line block coefficient','load line regulation'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'loadline'
*)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) | (temp_prop_def_rep.name = 'loadline')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'loadline' shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'load line length', 'load line depth', 'load line displacement', 'load line block coefficient', or 'load line regulation'.

5.2.4.162 representation_for_local_coordinate_system

The **representation_for_local_coordinate_system** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'local coordinate system'.

EXPRESS specification:

```
*)
RULE representation_for_local_coordinate_system
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['local axes and origin'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'local coordinate system')
                )
             ) > 0
          );

  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'local coordinate system' shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_-item** with a **name** of 'local axes and origin'.

5.2.4.163 representation_for_moment_3d_restricts_representation_item

The **representation_for_moment_3d_restricts_representation_item** rule specifies the **items** attribute of a **representation** with the **name** 'moment 3d' to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list.

EXPRESS specification:

```
*)
RULE representation_for_moment_3d_restricts_representation_item
FOR (representation);
LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['longitudinal moment',
 'transverse moment', 'vertical moment', 'origin'];
    violation: LOGICAL := FALSE;
```

```
  END_LOCAL;

  (* find all instances of representation with name equal to 'moment 3d'
   *)
  reps := QUERY(i <* representation |
       i.NAME = 'moment 3d');

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item for each name in the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every instance of **representation** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'moment 3d' shall for each value of 'longitudinal moment', 'transverse moment', 'vertical moment', and 'origin' collect one instance of **representation_item** in its attribute **items** whose name attribute equals that value.

5.2.4.164 representation_for_moments_of_inertia

The **representation_for_moments_of_inertia** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'moments of inertia'.

EXPRESS specification:

```
*)
RULE representation_for_moments_of_inertia
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['long moment of inertia',
 'trans moment of inertia'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'moments of
inertia' *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
```



```
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'moments of inertia')
                   )
               ) > 0
           );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘moments of inertia’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_-item** with a **name** of either 'long moment of inertia' or 'trans moment of inertia'.

5.2.4.165 representation_for_net_tonnage

The **representation_for_net_tonnage** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_-representation** whose **name** attribute has a value 'net tonnage'.

EXPRESS specification:

```
*)
RULE representation_for_net_tonnage
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['tonnage value'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'net
tonnage' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) | (temp_prop_def_rep.name = 'net tonnage')
```

```
                    )
               ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'net tonnage', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_-item** with a **name** of 'tonnage value'.

5.2.4.166 representation_for_person_group

The **representation_for_person_group** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_-representation** whose **name** attribute has a value 'person group parameters'.

EXPRESS specification:

```
*)
RULE representation_for_person_group
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['number of people', 'person type',
'volume'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'person
group parameters' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'person group parameters')
```



```
                    )
               ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'person group parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a name of either 'number of people', 'person type', or 'volume'.

5.2.4.167 representation_for_primer_coating

The **representation_for_primer_coating** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'primer coating parameters'.

EXPRESS specification:

```
*)
RULE representation_for_primer_coating
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['dry film thickness', 'number of coats',
'surface preparation'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'primer
coating parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'primer coating parameters')
                   )
```

```
            ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘primer coating parameters’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'dry film thickness', 'number of coats', or 'surface preparation'.

5.2.4.168 representation_for_principal_characteristics

The **representation_for_principal_characteristics** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'principal characteristics'.

EXPRESS specification:

```
*)
RULE representation_for_principal_characteristics
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['length between perpendiculars',
 'moulded breadth', 'moulded depth'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'principal
characteristics' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'principal characteristics')
```



```
                    )
                ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with name of 'principal characteristics' shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a name of either 'length between perpendiculars', 'moulded breadth', 'moulded depth', 'design draught', 'design deadweight', 'min draught at fp', 'max draught at fp', 'min draught at ap', or 'max draught at ap'.

5.2.4.169 representation_for_space_adjacency_relationship

The **representation_for_space_adjacency_relationship** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'space adjacency relationship parameters'.

EXPRESS specification:

```
*)
RULE representation_for_space_adjacency_relationship
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['adjacency access', 'adjacency type'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'space
adjacency relationship parameters' *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
```

```
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'space adjacency relationship parameters')
                  )
              ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘space adjacency relationship parameters’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'adjacency access' or 'adjacency type'.

5.2.4.170 representation_for_space_positional_relationship

The **representation_for_space_positional_relationship** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'space positional relationship parameters'.

EXPRESS specification:

```
*)
RULE representation_for_space_positional_relationship
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['relationship type'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'space
positional relationship parameters' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
```



```
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'space positional relationship parameters')
                  )
              ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘space positional relationship parameters’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of 'relationship type'.

5.2.4.171 representation_for_stability_table_restricted

The **representation_for_stability_table_restricted** rule specifies the **items** attribute of a **representation** with the class id 'stability table' to have one or more **representation_items** whose class id is 'stability properties for one floating position'.

EXPRESS specification:

```
*)
RULE representation_for_stability_table_restricted
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF representation := [];
    t2_set:  SET OF representation_item := [];
    t3_set:  SET OF representation_item := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'stability table'
*)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.Assigned_class.name = 'stability table');

  (* get all instances of representation that have class id 'stability
table' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
```

```
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

(* get all classification_assignment instances with id 'stability
properties for one floating position' *)
  c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.Assigned_class.name = 'stability properties for one floating position');

  (* get all instances of representation_item that have class id 'stability
properties for one floating position' *)
  REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
      t2_set := t2_set + c2_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

(* get all representation_item instances which are the .items of the
representation
instances that have class id 'stability table' *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].items);
(* compare both lists with representation_item instances and the
intersection has to be
greater 0 *)
      t3_set := t3_set + t1_set[i].items[j];
    END_REPEAT;
  violation := (SIZEOF(t3_set* t2_set) < 1);
  t3_set:= [];
  END_REPEAT;

WHERE
    wr1: NOT violation;
  END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **representation** that has an **applied_classification_assignment** whose attribute **assigned_class.name** equals 'stability table' shall collect at least one instance of **representation_item** that has an **applied_classification_assignment** whose **assigned_class.name** equals 'stability properties for one floating position'.

5.2.4.172 representation_for_stability_table_restricted_by_class_id

The **representation_for_stability_table_restricted_by_class_id** rule specifies the **items** attribute of a **representation** with the class id 'stability table' to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list.

EXPRESS specification:

```
*)
RULE representation_for_stability_table_restricted_by_class_id
FOR (applied_classification_assignment);
LOCAL
```

```
  c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
  t1_set: SET OF REPRESENTATION := [];
  arg_list: LIST OF STRING := ['mean shell thickness'];
  violation: LOGICAL := FALSE;
END_LOCAL;

(* get all classification_assignment instances with id
 'stability table'*)

c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
  i.assigned_class.NAME = 'stability table');

(* get all instances of representation that have class id *)
(* 'stability table'*)

REPEAT i := 1 TO HIINDEX(c_a_set);
  REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
    t1_set := t1_set + c_a_set[i].items[j];
  END_REPEAT;
END_REPEAT;

(*iterate over all representation instances found above; stop, *)
(* if one of them has not exactly one rep_item for each name in *)
(* the arg_list*)

REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
  REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
    violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
    rep_item.name = arg_list[j])) <> 1);
  END_REPEAT;
END_REPEAT;
WHERE
  WR1: NOT violation;
END_RULE;
(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment**.

Formal propositions:

WR1: Every instance of **representation** that has an **applied_classification_assignment** whose attribute **assigned_class.name** equals 'stability table' shall have in its attribute **items** one **representation_item** whose **name** attribute has the value `mean shell thickness`.

5.2.4.173 representation_for_tank_compartment_property

The **representation_for_tank_compartment_property** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'tank compartment property'.

EXPRESS specification:

```
*)
RULE representation_for_tank_compartment_property
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['context', 'design stowage density'];
```

```
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'tank
compartment property' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.name = 'tank compartment property')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘tank compartment property’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'context' or 'design stowage density'.

5.2.4.174 representation_for_tonnage_definition

The **representation_for_tonnage_definition** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'tonnage definition'.

EXPRESS specification:

```
*)
RULE representation_for_tonnage_definition
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['tonnage regulation'];
    violation: LOGICAL := FALSE;
```



```
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal TO 'tonnage
definition' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'tonnage definition')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘tonnage definition’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_-item** with a **name** of 'tonnage regulation'.

5.2.4.175 representation_for_tonnage_measurement

The **representation_for_tonnage_measurement** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'tonnage measurement'.

EXPRESS specification:

```
*)

RULE representation_for_tonnage_measurement
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['tonnage value'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
```

```
     by a property_definition_representation with name equal to 'tonnage
measurement' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.name = 'tonnage measurement')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'tonnage measurement', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of 'tonnage value'.

5.2.4.176 representation_for_unit_cargo

The **representation_for_unit_cargo** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'unit cargo parameters'.

EXPRESS specification:

```
*)
RULE representation_for_unit_cargo
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['cargo type', 'pollution code',
 'un type code'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
```



```
     by a property_definition_representation with name equal to 'unit cargo
parameters' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'unit cargo parameters')
                     )
               ) > 0
            );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'unit cargo parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'cargo type', 'pollution code', or 'un type code'.

5.2.4.177 representation_for_unit_cargo_assignment

The **representation_for_unit_cargo_assignment** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'unit cargo assignment parameters'.

EXPRESS specification:

```
*)
RULE representation_for_unit_cargo_assignment
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['assignment context',
  'cargo identifier'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
```

```
     by a property_definition_representation with name equal to 'unit cargo
assignment parameters' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'unit cargo assignment parameters')
                )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'unit cargo assignment parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'assignment context' or 'cargo identifier'.

5.2.4.178 representation_for_unit_cargo_bounding_box

The **representation_for_unit_cargo_bounding_box** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'unit cargo bounding box'.

EXPRESS specification:

```
*)
RULE representation_for_unit_cargo_bounding_box
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['point max',  'point min'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'unit cargo
bounding box' *)
```



```
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'unit cargo bounding box')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'unit cargo bounding box', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'point max' or 'point min'.

5.2.4.179 representation_for_unit_cargo_group

The **representation_for_unit_cargo_group** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'unit cargo group parameters'.

EXPRESS specification:

```
*)
RULE representation_for_unit_cargo_group
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['volume'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'unit cargo
group parameters' *)
  reps := QUERY(
          temp_rep <* representation |
```

```
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'unit cargo group parameters')
                 )
              ) > 0
           );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'unit cargo group parameters', shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of 'volume'.

5.2.4.180 representation_for_vehicle_load_description

The **representation_for_vehicle_load_description** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'vehicle load description'.

EXPRESS specification:

```
*)
RULE representation_for_vehicle_load_description
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['load handling', 'load per wheel',
'number of wheels', 'type of vehicle'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'vehicle
load description' *)
  reps := QUERY(
            temp_rep <* representation |
```



```
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'vehicle load description')
                    )
              ) > 0
           );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘vehicle load description’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'load handling', 'load per wheel', 'number of wheels', or 'type of vehicle'.

5.2.4.181 representation_for_zone_function

The **representation_for_zone_function** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_-representation** whose **name** attribute has a value 'zone function parameters'.

EXPRESS specification:

```
*)
RULE representation_for_zone_function
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['used for'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'zone
function parameters' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
```

```
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'zone function parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of ‘zone function parameters’ shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of 'used for'.

5.2.4.182 representation_has_global_uncertainty_assigned_context

The **representation_has_global_uncertainty_assigned_context** rule specifies that the type list for the entity referenced by **shape_representation.context_of_items** includes **global_uncertainty_assigned_context**.

EXPRESS specification:

```
*)
RULE representation_has_global_uncertainty_assigned_context
     FOR (SHAPE_REPRESENTATION);
LOCAL
 has_gunac: LOGICAL := TRUE;
END_LOCAL;

REPEAT i := 1 TO HIINDEX(SHAPE_REPRESENTATION) WHILE has_gunac;
 has_gunac :=
('SHIP_ARRANGEMENT_SCHEMA.GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT' IN
                TYPEOF(SHAPE_REPRESENTATION[i].CONTEXT_OF_ITEMS));
END_REPEAT;

WHERE
 WR1: has_gunac;
END_RULE;

(*
```


Argument definitions:

shape_representation: the set of all instances of **shape_representation** entities.

Formal propositions:

WR1: Every **shape_representation** must point to a **representation_context** via its **context_of_items** attribute that is of type **global_uncertainty_assigned_context.**

5.2.4.183 representation_has_global_unit_assigned_context

The **representation_has_global_unit_assigned_context** rule specifies that a **representation** has a **global_unit_assigned_context** if it has **representation_item**s of type **value_representation_item** or **geometric_representation_item**.

EXPRESS specification:

```
*)
RULE representation_has_global_unit_assigned_context
     FOR (REPRESENTATION);
  LOCAL
    has_guac: LOGICAL := TRUE;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE has_guac;
    REPEAT j := 1 TO SIZEOF(REPRESENTATION[i].ITEMS) WHILE has_guac;
      IF (('SHIP_ARRANGEMENT_SCHEMA.VALUE_REPRESENTATION_ITEM'
          IN TYPEOF(REPRESENTATION[i].ITEMS[j])) OR
          ('SHIP_ARRANGEMENT_SCHEMA.GEOMETRIC_REPRESENTATION_ITEM'
           IN TYPEOF(REPRESENTATION[i].ITEMS[j]))) THEN
      has_guac := ('SHIP_ARRANGEMENT_SCHEMA.GLOBAL_UNIT_ASSIGNED_CONTEXT'
         IN TYPEOF(REPRESENTATION[i].CONTEXT_OF_ITEMS));
      END_IF;
    END_REPEAT;
  END_REPEAT;
  WHERE
    WR1: has_guac;
END_RULE;
(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **representation** that contains either **value_representation_item**s or **geometric_representation_item**s has a **representation_context** that is a **global_unit_assigned_context**.

5.2.4.184 representation_item_for_transformation_to_parent

The **representation_item_for_transformation_to_parent** rule specifies that an instance of **property_definition** with class id 'local co ordinate system' is referenced by exactly one instance of **representation_map** via **mapped_representation**.

EXPRESS specification:

```
*)
RULE representation_item_for_transformation_to_parent
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
```

```
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    t3_set:  SET OF representation := [];
    t4_set:  SET OF representation_map := [];
    t5_set:  SET OF mapped_item := [];
    arg_list:  LIST OF STRING :=
['local coordinate system position in global coordinate system',
'local coordinate system position in parent local coordinate system',
'local coordinate system position in parent local coordinate system with
position reference'];
    violation1: LOGICAL := FALSE;
    violation2: LOGICAL := FALSE;

  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
            i.assigned_class.NAME = 'local co ordinate system');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation1;
    t2_set := bag_to_set(USEDIN(t1_set[i],

'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation1 := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
t2_inst.used_representation.name =
      'local axis representation')) = 1);
    t3_set := t3_set + t2_set[i].used_representation;
  END_REPEAT;

   REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation1;
    t4_set := bag_to_set(USEDIN(t3_set[i],
      'SHIP_ARRANGEMENT_SCHEMA.REPRESENTATION_MAP.MAPPED_REPRESENTATION'));
  END_REPEAT;

   REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation1;
    t5_set := bag_to_set(USEDIN(t4_set[i],
      'SHIP_ARRANGEMENT_SCHEMA.MAPPED_ITEM.MAPPING_SOURCE'));
      REPEAT j := 1 TO 3 WHILE NOT violation2;
      violation2 := NOT (SIZEOF(QUERY(t2_inst <* t5_set | t2_inst.name =
      ARG_LIST[j])) = 1);
      END_REPEAT;
  END_REPEAT;

  WHERE
    WR1: NOT violation1;
    WR2: NOT violation2;
END_RULE;
```

(*

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every **property_definition** that is referenced by an **applied_classification_assignment** with **name** = 'local co ordinate system', is referenced by exactly one **property_definition_-representation.definition**, where **property_definition_representation.name** = 'local coordinate system', and a **used_representation** attribute which is a **representation** with **name** = 'local axis representation'.

WR2: Every **mapped_item** with a **name** of 'local coordinate system position in global coordinate system', 'local coordinate system position in parent local coordinate system', or 'local coordinate system position in parent local coordinate system with position reference' has and attribute **mapping_source** that references a **representation_map** that has a **mapped_representation** attribute that references a **representation** with a **name** of = 'local axis representation'.

5.2.4.185 representation_items_optional_for_bulk_cargo

The **representation_items_optional_for_bulk_cargo** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'bulk cargo parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_bulk_cargo
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['flash point',
 'required carriage temperature', 'permeability', 'stowage factor',
 'user def cargo'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'bulk cargo
parameters'
   *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.name = 'bulk cargo parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
```

```
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'bulk cargo parameters', the values 'flash point', 'required carriage temperature', 'permeability', 'stowage factor', and 'user def cargo' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.186 representation_items_optional_for_capacity_properties

The **representation_items_optional_for_capacity_properties** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'capacity properties'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_capacity_properties
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['user defined capacity context'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'capacity
properties'
   *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'capacity properties')
                    )
              ) > 0
           );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                 rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
```



```
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item**s referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'capacity properties', the value 'user defined capacity context' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.187 representation_items_optional_for_class_deck_load_requirement_definition

The **representation_items_optional_for_class_deck_load_requirement_definition** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'class deck load requirement definition parameters'.

EXPRESS specification:

```
*)
RULE
representation_items_optional_for_class_deck_load_requirement_definition
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['grab weight', 'stowage height',
 'stowage rate', 'vehicle load'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'class deck
load requirement definition parameters'
   *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
'class deck load requirement definition parameters')
                 )
              ) > 0
           );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
```

```
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'class deck load requirement definition parameters', the value 'grab weight', 'stowage height', 'stowage rate', or 'vehicle load' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.188 representation_items_optional_for_class_notation

The **representation_items_optional_for_class_notation** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'class notation'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_class_notation
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['ice class notation','service factor',
'approval required for heavy cargo'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'class notation')
                )
             ) > 0
          );

  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;
```



```
  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item**s referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'class notation', the value 'ice class notation', 'service factor', or 'approval required for heavy cargo' shall not occur more than once as value of the **representation_item** attribute **name**.

5.2.4.189 representation_items_optional_for_compartment_access_authorization

The **representation_items_optional_for_compartment_access_authorization** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment access authorization'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_compartment_access_authorization
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['user defined value'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment access authorization'
   *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment access authorization')
                  )
               ) > 0
            );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
```

```
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item**s referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'compartment access authorization', the value 'user defined value' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.190 representation_items_optional_for_compartment_design_requirement

The **representation_items_optional_for_compartment_design_requirement** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment design requirement parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_compartment_design_requirement
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['user defined value'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment design requirement parameters'
   *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
 'compartment design requirement parameters')
                  )
               ) > 0
            );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
```



```
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'compartment design requirement parameters', the value 'user defined value' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.191 representation_items_optional_for_compartment_function

The **representation_items_optional_for_compartment_function** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment function parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_compartment_function
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['user def function'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment function parameters'
   *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.name = 'compartment function parameters')
                 )
              ) > 0
           );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
```

```
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'compartment function parameters', the value 'user def function' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.192 representation_items_optional_for_compartment_insulation

The **representation_items_optional_for_compartment_insulation** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment insulation'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_compartment_insulation
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['user defined value'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment insulation'
   *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment insulation')
                 )
              ) > 0
           );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
```



```
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'compartment insulation', the value 'user defined value' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.193 representation_items_optional_for_compartment_noise_category

The **representation_items_optional_for_compartment_noise_category** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment noise category'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_compartment_noise_category
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['user defined value'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment noise category'
   *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment noise category')
                  )
               ) > 0
            );

  (* iterate over all representations found above. Stop, if for one of
```

```
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'compartment noise category', the value 'user defined value' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.194 representation_items_optional_for_compartment_safety_class

The **representation_items_optional_for_compartment_safety_class** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment safety class'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_compartment_safety_class
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['user defined value'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment safety class'
   *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment safety class')
                   )
             ) > 0
```

```
          );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'compartment safety class', the value 'user defined value' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.195 representation_items_optional_for_compartment_security

The **representation_items_optional_for_compartment_security** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment security classification'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_compartment_security
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['user defined value'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment security classification'
   *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment security classification')
                  )
```

```
            ) > 0
          );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'compartment security classification', the value 'user defined value' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.196 representation_items_optional_for_compartment_tightness

The **representation_items_optional_for_compartment_tightness** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'compartment tightness'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_compartment_tightness
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['user defined value'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
'compartment tightness'
   *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'compartment tightness')
```



```
                    )
                ) > 0
            );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'compartment tightness', the value 'user defined value' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.197 representation_items_optional_for_corrosion_control_coating

The **representation_items_optional_for_corrosion_control_coating** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'corrosion control coating parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_corrosion_control_coating
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['user defined type'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'corrosion
control coating parameters'
   *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
```

```
 (temp_prop_def_rep.name = 'corrosion control coating parameters')
                    )
              ) > 0
           );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'corrosion control coating parameters', the value 'user defined type' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.198 representation_items_optional_for_damage_case

The **representation_items_optional_for_damage_case** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'damage case parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_damage_case
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['user defined'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'damage
case parameters'
   *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
```



```
 (temp_prop_def_rep.name = 'damage case parameters')
                    )
               ) > 0
          );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'damage case parameters', the value 'user defined' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.199 representation_items_optional_for_deck_zone_function

The **representation_items_optional_for_deck_zone_function** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'deck zone function parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_deck_zone_function
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['user def function'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'deck zone
function parameters'
   *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
```

```
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'deck zone function parameters')
                  )
              ) > 0
          );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item**s referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'deck zone function parameters', the value 'user def function' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.200 representation_items_optional_for_detailed_cargo_material_properties

The **representation_items_optional_for_detailed_cargo_material_properties** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'detailed cargo material properties parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_detailed_cargo_material_properties
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['density',  'expansion coefficient',
'specific heat capacity', 'thermal conductivity', 'viscosity'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'detailed
cargo material properties parameters'
   *)
  reps := QUERY(
```



```
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
(temp_prop_def_rep.name = 'detailed cargo material properties parameters')
                 )
              ) > 0
           );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'detailed cargo material properties parameters', the value 'density', 'expansion coefficient', 'specific heat capacity', 'thermal conductivity', and 'viscosity' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.201 representation_items_optional_for_dry_cargo

The **representation_items_optional_for_dry_cargo** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'dry cargo parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_dry_cargo
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['flash point',
'required carriage temperature', 'user def cargo', 'permeability',
'stowage factor'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'dry cargo
parameters'
```

```
   *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'dry cargo parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'dry cargo parameters', the values 'flash point', 'required carriage temperature', 'permeability', 'stowage factor', and 'user def cargo' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.202 representation_items_optional_for_gaseous_cargo

The **representation_items_optional_for_gaseous_cargo** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'gaseous cargo parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_gaseous_cargo
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['flash point',
'required carriage pressure', 'required carriage temperature',
'user def cargo'];
    found: LOGICAL := FALSE;
  END_LOCAL;
```

```
  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'gaseous
cargo parameters'
   *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'gaseous cargo parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'gaseous cargo parameters', the values 'flash point', 'required carriage pressure', 'required carriage temperature', and 'user def cargo' shall not occur more than once as values of the **representation_-item** attribute **name**.

5.2.4.203 representation_items_optional_for_general_cargo_material_properties

The **representation_items_optional_for_general_cargo_material_properties** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'general cargo material properties parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_general_cargo_material_properties
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['density'];
```

```
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'general
cargo material properties parameters'
   *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
(temp_prop_def_rep.name = 'general cargo material properties parameters')
                )
             ) > 0
          );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
               rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'general cargo material properties parameters', the value 'density' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.204 representation_items_optional_for_liquid_cargo

The **representation_items_optional_for_liquid_cargo** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'liquid cargo parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_liquid_cargo
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['flash point',
```



```
'required carriage pressure', 'required carriage temperature',
'user def cargo'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'liquid
cargo parameters'
   *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'liquid cargo parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'liquid cargo parameters', the values 'flash point', 'required carriage pressure', 'required carriage temperature', and 'user def cargo' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.205 representation_items_optional_for_loading_condition_operating_definition

The **representation_items_optional_for_loading_condition_operating_definition** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'loading condition operating definition parameters'.

EXPRESS specification:

```
*)
RULE
representation_items_optional_for_loading_condition_operating_definition
```

```
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['place of loading'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'loading
condition operating definition parameters'
   *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
 'loading condition operating definition parameters')
                 )
              ) > 0
           );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'loading condition operating definition parameters', the value 'place of loading' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.206 representation_items_optional_for_owner_designation

The **representation_items_optional_for_owner_designation** rule specifies the **items** attribute of a **representation** to have for each entry in the list, zero or one **representation_item** which has a **name** attribute with the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** which has a **name** attribute value of 'owner designation'.

EXPRESS specification:

```
*)
```



```
RULE representation_items_optional_for_owner_designation
   FOR (representation);
LOCAL
   reps: BAG OF REPRESENTATION := [];
   arg_list: LIST OF STRING := ['owner approval'];
   found: LOGICAL := FALSE;
END_LOCAL;

(* Find all instances of representation which are used
by a property_definition_representation with name equal to 'owner
designation'
*)

reps := QUERY(temp_rep <* representation |
  SIZEOF (QUERY(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
        'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
        'USED_REPRESENTATION'))|
 (temp_prop_def_rep.name = 'owner designation')))>0 );

(* iterate over all representations found above. Stop, if for one of
them the names of its representation_items are duplicated.
*)

REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
 REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
 found := (SIZEOF(QUERY(rep_item <* reps[i].items |
          rep_item.name = arg_list[j])) > 1);
 END_REPEAT;
END_REPEAT;
WHERE

wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **used_representation** attribute of a **property_definition_representation** with **name** = 'owner designation', the value of 'owner approval' is not to occur more than once as the value of the **representation_item** attribute **name**.

5.2.4.207 representation_items_optional_for_principal_characteristics

The **representation_items_optional_for_principal_characteristics** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'principal characteristics'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_principal_characteristics
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
```

```
    arg_list: LIST OF STRING := ['block coefficient', 'design draught',
'design deadweight','min draught at fp','max draught at fp',
'min draught at ap','max draught at ap'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'principal
characteristics'
   *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'principal characteristics')
                )
             ) > 0
          );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
               rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with name of 'principal characteristics', the value of 'block coefficient', 'design draught', 'design deadweight', 'min draught at fp', 'max draught at fp', 'min draught at ap', or 'max draught at ap' shall not occur more than once as value of the **representation_item** attribute **name**.

5.2.4.208 representation_items_optional_for_space_connection_relationship

The **representation_items_optional_for_space_connection_relationship** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'space connection relationship parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_space_connection_relationship
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['connecting system'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'space
connection relationship parameters'
   *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'space connection relationship parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'space connection relationship parameters', the value 'connecting system' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.209 representation_items_optional_for_tank_geometric_parameters

The **representation_items_optional_for_tank_geometric_parameters** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'tank geometric parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_tank_geometric_parameters
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['breadth wash', 'length wash'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'tank
geometric parameters'
   *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'tank geometric parameters')
                 )
              ) > 0
           );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                 rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'tank geometric parameters', the values 'breadth wash' and 'length wash' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.210 representation_items_optional_for_tank_piping_design_properties

The **representation_items_optional_for_tank_piping_design_properties** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'tank piping design properties'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_tank_piping_design_properties
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['airpipe height', 'filling height',
'relief valve pressure setting', 'sounding pipe height'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'tank
piping design properties'
   *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'tank piping design properties')
                 )
              ) > 0
           );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'tank piping design properties', the values 'airpipe height', 'filling height', 'relief valve pressure setting', and 'sounding pipe height' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.211 representation_items_optional_for_unit_cargo

The **representation_items_optional_for_unit_cargo** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name**

attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'unit cargo parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_unit_cargo
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['flash point', 'permeability',
'stowage factor', 'required carriage temperature', 'stack limit',
'user def cargo', 'volume'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'unit cargo
parameters'
   *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'unit cargo parameters')
                   )
             ) > 0
          );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;


(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'unit cargo parameters', the values 'flash point', 'permeability', 'stowage factor', 'required carriage temperature', 'stack limit', 'user def cargo', or 'volume' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.212 representation_items_optional_for_vehicle_load_description

The **representation_items_optional_for_vehicle_load_description** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'vehicle load description'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_vehicle_load_description
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['max tyre pressure', 'print area'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'vehicle
load description'
   *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'vehicle load description')
                 )
              ) > 0
           );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'vehicle load description', the value 'max tyre pressure' or 'print area' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.213 representation_items_optional_for_zone_function

The **representation_items_optional_for_zone_function** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'zone function parameters'.

EXPRESS specification:

```
*)
RULE representation_items_optional_for_zone_function
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['user def function'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to 'zone
function parameters'
   *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'zone function parameters')
                )
             ) > 0
          );

  (* iterate over all representations found above. Stop, if for one of
     them the names of its representation_items are duplicated.
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: For all **representation_item**s referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** of 'zone function parameters', the value 'user def function' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.214 representation_local_coordinate_system_with_position_reference

The **representation_local_coordinate_system_with_position_reference** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** whose **name** attribute has a value 'local coordinate system with position reference'.

EXPRESS specification:

```
*)
RULE representation_local_coordinate_system_with_position_reference
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['local axes and origin'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
      'local coordinate system with position reference')
                   )
              ) > 0
          );

  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **property_definition_representation** with **name** of ‘local coordinate system with position reference’, shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of 'local axes and origin'.

5.2.4.215 representation_restricted_weight_and_centre_of_gravity

The **representation_restricted_weight_and_centre_of_gravity** rule specifies that the **items** attribute of a **representation** to have for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list if the **representation** is the **used_representation**

in a **property_definition_representation** whose **name** attribute has a value 'weight and centre of gravity'.

EXPRESS specification:

```
*)
RULE representation_restricted_weight_and_centre_of_gravity
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['mass ','centre of gravity'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'weight and
centre of gravity' *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'weight and centre of gravity')
                 )
              ) > 0
           );

  (* iterate over all representations found above; stop, if one of
     them has not exactly one rep_item with for each name of the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: every **property_definition_representation** with **name** of 'weight and centre of gravity' shall use a **representation** that has exactly one item in its set of **items** that shall be a type of **representation_item** with a **name** of either 'mass', or 'centre of gravity'.

5.2.4.216 revision_has_mandatory_attribute_description

The **revision_has_mandatory_attribute_description** rule specifies that for an instance of **group** with class id 'revision' the optional attribute **description** is instantiated.

EXPRESS specification:

```
*)
RULE revision_has_mandatory_attribute_description
FOR (group);
LOCAL
    t1_set:  SET OF group := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

(* get all instances of group being classified as 'revision' *)
t1_set := QUERY(i <* group | VALUE_IN(WHICH_CLASS(i), 'revision'));

(* from all instances found above:
   find those for which attribute description is not instantiated
*)

violate := (SIZEOF(QUERY(k <* t1_set |
NOT EXISTS (k.description))) > 0);

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

group: the set of all instances of **group** entities.

Formal propositions:

WR1: The optional attribute **description** shall exist for every instance of **group** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'revision'.

5.2.4.217 revision_with_context_referenced_for_context_of_revision

The **revision_with_context_referenced_for_context_of_revision** rule specifies that each instance of type **group** with class 'revision with context' shall be referenced by exactly one assignment of type **applied_group_assignment** with **role** 'context of revision' via attribute **assigned_group**.

EXPRESS specification:

```
*)
RULE revision_with_context_referenced_for_context_of_revision
FOR(applied_group_assignment, group);
  LOCAL
    t1_set:  SET OF group := [];
    a_set:  SET OF applied_group_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

(* get all instances of group with class 'revision with context' *)
t1_set := QUERY(a <* group |
 VALUE_IN(WHICH_CLASS(a), 'revision with context'));

  (* for all instances found above *)
REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
(* get the instances of applied_group_assignment that have a role 'context
of revision' *)
   a_set := QUERY(b <* applied_group_assignment |
             (b.assigned_group = t1_set[i]) AND
```

```
            (b.role.name = 'context of revision'));

(* there shall be no such instances *)
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_group_assignment: the set of all instances of **applied_group_assignment** entities.

group: the set of all instances of **group** entities

Formal propositions:

WR1: Every instance of **group** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'revision with context' shall be referenced from exactly one instance of type **applied_group_assignment** whose **role** equals 'context of revision' through attribute **assigned_group**.

5.2.4.218 shape_representation_subtype_exclusiveness

Ensures that an instance of **shape_representation** is only a **non_manifold_surface_shape_-representation**.

EXPRESS specification:

```
*)
RULE shape_representation_subtype_exclusiveness
FOR (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (sr <* shape_representation |
    NOT (SIZEOF (TYPEOF (sr) *
    ['SHIP_ARRANGEMENT_SCHEMA.NON_MANIFOLD_SURFACE_SHAPE_REPRESENTATION'])
    <= 2))) = 0;
END_RULE;
(*
```

Argument definitions:

shape_representation: the set of all instances of **shape_representation**.

Formal propositions:

WR1: Each instance of **shape_representation** is a **non_manifold_surface_shape_representation**.

5.2.4.219 ship_designation_has_one_specified_names

The **ship_designation_has_one_specified_names** rule specifies that an instance of **product_definition** with class id 'ship designation' is referenced by exactly one instance of **applied_-identification_assignment** via **items** whose attribute **role** is an **identification_role** with attribute **name** equal to either 'imo number' or 'pennant hull number'.

EXPRESS specification:

```
*)
RULE ship_designation_has_one_specified_names
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_identification_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'ship designation'
*)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'ship designation');

  (* get all instances of T1 that have class id 'ship designation' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of product_definition in t1_set:
     get the applied_identification_assignment instances that are
referencing a product_definition instance via items, filter out those
applied_identification_assignment instances whose attribute role.name has
the value ‘imo number’; check if their number equals 1.
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | (t2_inst.role.name =
'imo number')OR (t2_inst.role.name = 'pennant hull number') )) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **product_definition** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'ship designation' is referenced by exactly one instance of **applied_identification_assignment** through attribute **items** whose attribute **role** is an **identification_role** with attribute **name** equal to either 'imo number' or 'pennant hull number'.

5.2.4.220 spacing_position_compound_representation_has_name

The **spacing_position_compound_representation_has_name** rule specifies the **item_element** attribute of a **compound_representation_item** with the class id 'spacing position' to have in the list of **representation_item** for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list.

EXPRESS specification:

```
*)
RULE spacing_position_compound_representation_has_name
FOR (applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF Compound_representation_item := [];
    t2_set:  SET Of representation_item := [];
    arg_list:  LIST OF STRING := ['position number', 'position'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME = 'spacing position');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set |
           items.name = arg_list[j])) <> 1);
 END_REPEAT;
  END_REPEAT;
  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **compound_representation_item** that has an **applied_classification_-assignment** whose attribute **assigned_class** is a **group** with attribute **name** equal 'spacing position' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall for each value of 'position number' or 'position' collect exactly one instance of **representation_item** whose **name** attribute equals that value.

5.2.4.221 spacing_position_with_offset_compound_representation_has_class

The **spacing_position_with_offset_compound_representation_has_class** rule specifies the **item_-element** attribute of a **compound_representation_item** with the class id 'spacing position with offset' to have in the list of **representation_item** exactly one **compound_representation_item** with the class id 'spacing position'.

EXPRESS specification:

```
*)
RULE spacing_position_with_offset_compound_representation_has_class
```



```
FOR (applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t3_set:  SET Of representation_item := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.assigned_class.NAME = 'spacing position with offset');
REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;
  c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.assigned_class.NAME = 'spacing position');

  REPEAT i := 1 TO HIINDEX(c_a_set2);
    REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
      t2_set := t2_set + c_a_set2[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
      t3_set := t3_set + t1_set[i].item_element;
    END_REPEAT;
    violation := (SIZEOF(t3_set * t2_set) <> 1);
    t3_set:= [];
  END_REPEAT;
  WHERE
    WR1: NOT violation;
END_RULE;
```

(*

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **compound_representation_item** that has an **applied_classification_-assignment** whose attribute **assigned_class** is a **group** with attribute **name** equal 'spacing position with offset' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall collect exactly one instance of **compound_representation_item** that has an **applied_-classification_assignment** whose attribute **assigned_class** equals 'spacing position'.

5.2.4.222 spacing_position_with_offset_compound_representation_has_name

The **spacing_position_with_offset_compound_representation_has_name** rule specifies the **item_element** attribute of a **compound_representation_item** with the class id 'spacing position with offset' to have in the list of **representation_item** for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list.

EXPRESS specification:

```
*)
RULE spacing_position_with_offset_compound_representation_has_name
FOR (applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF Compound_representation_item := [];
    t2_set:  SET Of representation_item := [];
    arg_list:  LIST OF STRING := ['offset'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME =
                  'spacing position with offset');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set |
        items.name = arg_list[j])) <> 1);
 END_REPEAT;
  END_REPEAT;
  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **compound_representation_item** that has an **applied_classification_-assignment** whose attribute **assigned_class** is a **group** with attribute **name** equal 'spacing position with offset' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall contain exactly one instance of **representation_item** whose **name** attribute equals 'offset'.

5.2.4.223 stability_properties_for_floating_position_has_class

The **stability_properties_for_floating_position_has_class** rule specifies the **item_element** attribute of a **compound_representation_item** with the class id 'stability properties for one floating position' to have in the **list_representation_item** one or more **compound_representation_item** with the class id 'stability property'.

EXPRESS specification:

```
*)
RULE stability_properties_for_floating_position_has_class
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
```



```
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t3_set:  SET OF REPRESENTATION_ITEM := [];
    l_rep_item : list_representation_item;
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'stability
properties for one floating position' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.Assigned_class.name = 'stability properties for one floating position');

  (* get all instances of compound_representation_item that have class id
'stability properties for one floating position' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

(* get all classification_assignment instances with id 'stability property'
*)
  c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
i.Assigned_class.name = 'stability property');

  (* get all instances of compound_representation_item that have class id
'stability property' *)
  REPEAT i := 1 TO HIINDEX(c_a_set2);
    REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
      t2_set := t2_set + c_a_set2[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* iterate over all compound_representation_item found in the first list;
then iterate over all item_element for each compound_representation_item,
check that the intersection of these item_elements and and the second list
of compound_representation_item is greater than or equal to 1.
   *)
  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
          l_rep_item := t1_set[i].item_element;
      t3_set := t3_set + l_rep_item[j];
    END_REPEAT;
    violation := (SIZEOF(t3_set * t2_set) < 1);
    t3_set:= [];
  END_REPEAT;

  WHERE
    wr1: NOT violation;

END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **compound_representation_item** that has an **applied_classification_assignment** whose attribute **assigned_class.name** equals 'stability properties for one floating position' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall collect one or more instances of **compound_representation_item** that has an **applied_classification_assignment** whose attribute **assigned_class** equals 'stability property'.

5.2.4.224 stability_properties_for_floating_position_has_name

The **stability_properties_for_floating_position_has_name** rule specifies the **item_element** attribute of a **compound_representation_item** with the class id 'stability properties for one floating position' to have in the **list_representation_item** for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list.

EXPRESS specification:

```
*)
RULE stability_properties_for_floating_position_has_name
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF Compound_representation_item := [];
    t2_set:  SET OF representation_item := [];
    arg_list:  LIST OF STRING := ['centre of gravity above keel',
'definition of starting floating position'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'stability
properties for one floating position' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.Assigned_class.name =
 'stability properties for one floating position');

  (* get all instances of compound_representation_item that have class id
'stability properties for one floating position' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* iterate over all compound_representation_item found above; stop, if
one of
     them has not exactly one rep_item for each name in the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
  violation := (SIZEOF(QUERY(items <* t2_set | items.name = arg_list[j]))
<> 1);
 END_REPEAT;
  END_REPEAT;
```



```
  WHERE
    wr1: NOT violation;

END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **compound_representation_item** that has an **applied_classification_-assignment** whose attribute **assigned_class.name** equals 'stability properties for one floating position' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall for each value out of 'centre of gravity above keel' or 'definition of starting floating position' collect exactly one instance of **representation_item** whose **name** attribute equals that value.

5.2.4.225 stability_property_has_name

The **stability_property_has_name** rule specifies the **item_element** attribute of a **compound_-representation_item** with the class id 'stability property' to have in the **list_representation_item** for each entry in the list exactly one **representation_item** whose **name** attribute has the value given in the list.

EXPRESS specification:

```
*)
RULE stability_property_has_name
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF Compound_representation_item := [];
    t2_set:  SET OF representation_item := [];
    arg_list:  LIST OF STRING := ['angle of heel', 'righting arm',
 'centre of buoyancy'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'stability
property' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.Assigned_class.name = 'stability property');

  (* get all instances of compound_representation_item that have class id
'stability property' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* iterate over all compound_representation_item found above; stop, if
one of them has not exactly one rep_item for each name in the arg_list
   *)
  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
   violation := (SIZEOF(QUERY(items <* t2_set |
```

```
 items.name = arg_list[j])) <> 1);
 END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;

END_RULE;
```

(*

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **compound_representation_item** that has an **applied_classification_-assignment** whose attribute **assigned_class.name** equals 'stability property' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall for each value out of 'angle of heel', 'righting arm', or 'centre of buoyancy' collect exactly one instance of **representation_item** whose **name** attribute equals that value.

5.2.4.226 tonnage_definition_has_properties

The **tonnage_definition_has_properties** rule specifies that a **product_definition** with a class id 'tonnage definition' is referenced by one **property_definition_representation** with the **name** 'tonnage definition' via a **property_definition.**

EXPRESS specification:

```
*)
RULE tonnage_definition_has_properties
FOR (property_definition_representation,
applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  LIST OF product_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    t3_set:  LIST OF property_definition := [];
    t4_set:  LIST OF product_definition := [];

    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'tonnage
definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.name = 'tonnage definition');

  (* get all instances of product_definition that have class id 'tonnage
definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;
```



```
  (* get all property_definition_representation instances with name
'tonnage definition' *)
  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
                  i.NAME = 'tonnage definition');

(* get all property_definition instances which are the .definition of the
property_definition_representation *)
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
  END_REPEAT;

(* get all product_definition instances which are the .definition of the
property_definition *)
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;

(* compare both lists with product_definition instances which have to be
identical *)
  violation := t1_set <> t4_set;

  WHERE
    wr1: NOT violation;


END_RULE;

(*
```

Argument definitions:

property_definition_representation: the set of all instances of **property_definition_-representation** entities.

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every instance of **product_definition** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'tonnage definition' shall be referenced by exactly one instance of **property_definition** through attribute **definition** that in turn is referenced by an instance of **property_definition_representation** through attribute **definition** whose attribute **name** equals 'tonnage definition'.

5.2.4.227 unique_approvals_in_approval_history

The **unique_approvals_in_approval_history** rule specifies that all instances of **approval** in a **group** with class approval_history must be unique.

EXPRESS specification:

```
*)
RULE unique_approvals_in_approval_history
FOR (GROUP, APPLIED_GROUP_ASSIGNMENT);
LOCAL
    t1_set:  SET OF GROUP := [];
    t2_set:  SET OF APPLIED_GROUP_ASSIGNMENT := [];
    t3_set:  SET OF approval :=[];
    violate: LOGICAL := FALSE;
```

```
END_LOCAL;
  t1_set := QUERY(i <* GROUP | VALUE_IN(WHICH_CLASS(i),
    'approval history'));
REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
   t2_set := QUERY(a <* APPLIED_GROUP_ASSIGNMENT |
     a.ASSIGNED_GROUP = t1_set[i]);
   t3_set := QUERY(b <* t2_set[1].items |
     'SHIP_ARRANGEMENT_SCHEMA.APPROVAL' IN TYPEOF(b));
    violate := NOT (VALUE_UNIQUE(t3_set));
END_REPEAT;
WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_group_assignment: the set of all instances of **applied_group_assignment** entities.

group: the set of all instances of **group** entities.

Formal propositions:

WR1: All instances of **approval** that are items of an instance of **group** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'approval history' must be unique.

5.2.4.228 user_def_cargo_description_required_for_cargo_type

The **user_def_cargo_description_required_for_cargo_type** rule specifies a **representation** that has a **descriptive_representation_item** with **name** 'cargo type' and **description** 'user defined' to have another **descriptive_representation_item** with **name** 'user def cargo'.

EXPRESS specification:

```
*)
RULE user_def_cargo_description_required_for_cargo_type
FOR (REPRESENTATION);

  LOCAL
    violation: LOGICAL := FALSE;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
    violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |

('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
IN TYPEOF(r)) AND
(r.NAME = 'cargo type') AND
(r\DESCRIPTIVE_REPRESENTATION_ITEM.DESCRIPTION = 'user defined'))) > 0)
AND (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
(r.NAME = 'user def cargo'))) = 0);
  END_REPEAT;

  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **representation** that has a **descriptive_representation_item** with **name** 'cargo type' and **description** 'user defined' in its set of **items** shall have another **descriptive_representation_item** with **name** 'user def cargo' in its set of **items**.

5.2.4.229 user_def_cargo_description_required_for_type_of

The **user_def_cargo_description_required_required_for_type_of** rule specifies a **representation** that has a **descriptive_representation_item** with **name** 'type of' and **description** 'user defined' to have another **descriptive_representation_item** with **name** 'user def cargo'.

EXPRESS specification:

```
*)
RULE user_def_cargo_description_required_for_type_of
FOR (REPRESENTATION);

  LOCAL
    violation: LOGICAL := FALSE;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
    violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |

('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
 IN TYPEOF(r)) AND
 (r.NAME = 'type of') AND
 (r\DESCRIPTIVE_REPRESENTATION_ITEM.DESCRIPTION = 'user defined'))) > 0)
 AND (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
 (r.NAME = 'user def cargo'))) = 0);
  END_REPEAT;

  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **representation** that has a **descriptive_representation_item** with **name** 'type of' and **description** 'user defined' in its set of **items** shall have another **descriptive_representation_item** with **name** 'user def cargo' in its set of **items**.

5.2.4.230 user_def_function_description_required

The **user_def_function_description_required** rule specifies a **representation** that has a **descriptive_representation_item** with **name** 'used for' and **description** 'user defined' to have another **descriptive_representation_item** with **name** 'user def function'.

EXPRESS specification:

```
*)
RULE user_def_function_description_required
FOR (REPRESENTATION);

  LOCAL
    violation: LOGICAL := FALSE;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
    violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |

('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
                          IN TYPEOF(r)) AND
                         (r.NAME = 'used for') AND
                         (r\DESCRIPTIVE_REPRESENTATION_ITEM.DESCRIPTION =
'user defined'))) > 0)
                 AND
                 (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
                         (r.NAME = 'user def function'))) = 0);
  END_REPEAT;

  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **representation** that has a **descriptive_representation_item** with **name** ‘used for’ and **description** ‘user defined’ in its set of **items** shall have another **descriptive_representation_item** with **name** ‘user def function’ in its set of **items**.

5.2.4.231 user_defined_capacity_context_description_required_for_capacity_context

The **user_defined_capacity_context_description_required_for_capacity_context** rule specifies a **representation** that has a **descriptive_representation_item** with **name** 'capacity context' and **description** 'user defined' to have another **descriptive_representation_item** with **name** 'user defined capacity context'.

EXPRESS specification:

```
*)
RULE
user_defined_capacity_context_description_required_for_capacity_context
FOR (REPRESENTATION);

  LOCAL
    violation: LOGICAL := FALSE;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
    violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |

('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
                          IN TYPEOF(r)) AND
                         (r.NAME = 'capacity context') AND
```



```
                        (r\DESCRIPTIVE_REPRESENTATION_ITEM.DESCRIPTION =
'user defined'))) > 0)
                AND
                (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
                        (r.NAME = 'user defined capacity context'))) = 0);
  END_REPEAT;

  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **representation** that has a **descriptive_representation_item** with **name** ‘capacity context’ and **description** ‘user defined’ in its set of **items** shall have another **descriptive_-representation_item** with **name** ‘user defined capacity context’ in its set of **items**.

5.2.4.232 user_defined_description_required_for_damage_cause

The **user_defined_description_required_for_damage_cause** rule specifies a **representation** that has a **descriptive_representation_item** with **name** 'damage cause' and **description** 'user defined' to have another **descriptive_representation_item** with **name** 'user defined'.

EXPRESS specification:

```
*)
RULE user_defined_description_required_for_damage_cause
FOR (REPRESENTATION);

  LOCAL

    violation: LOGICAL := FALSE;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
    violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |

('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
                         IN TYPEOF(r)) AND
                        (r.NAME = 'damage cause') AND
                        (r\DESCRIPTIVE_REPRESENTATION_ITEM.DESCRIPTION =
'user defined'))) > 0)
                AND
                (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
                        (r.NAME = 'user defined'))) = 0);
  END_REPEAT;

  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **representation** that has a **descriptive_representation_item** with **name** 'damage cause' and **description** 'user defined' in its set of **items** shall have another **descriptive_representation_item** with **name** 'user defined' in its set of **items**.

5.2.4.233 user_defined_type_description_required_for_type_of

The **user_defined_type_description_required_for_type_of** rule specifies a **representation** that has a **descriptive_representation_item** with **name** 'type of' and **description** 'user defined' to have another **descriptive_representation_item** with **name** 'user defined type'.

EXPRESS specification:

```
*)
RULE user_defined_type_description_required_for_type_of
FOR (REPRESENTATION);

  LOCAL
    violation: LOGICAL := FALSE;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
    violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |

('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
                          IN TYPEOF(r)) AND
                         (r.NAME = 'type of') AND
                         (r\DESCRIPTIVE_REPRESENTATION_ITEM.DESCRIPTION =
'user defined'))) > 0)
                 AND
                 (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
                         (r.NAME = 'user defined type'))) = 0);
  END_REPEAT;

  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **representation** that has a **descriptive_representation_item** with **name** ‘type of’ and **description** ‘user defined’ in its set of **items** shall have another **descriptive_representation_item** with **name** ‘user defined type’ in its set of **items**.

5.2.4.234 user_defined_value_description_required_for_authorization_classification

The **user_defined_value_description_required_for_authorization_classification** rule specifies a **representation** that has a **descriptive_representation_item** with **name** 'authorization classification' and **description** 'user defined' to have another **descriptive_representation_item** with **name** 'user defined value'.

EXPRESS specification:


```
*)
RULE
user_defined_value_description_required_for_authorization_classification
FOR (REPRESENTATION);

  LOCAL
    violation: LOGICAL := FALSE;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
    violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |

('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
                          IN TYPEOF(r)) AND
                         (r.NAME = 'authorization classification') AND
                         (r\DESCRIPTIVE_REPRESENTATION_ITEM.DESCRIPTION =
'user defined'))) > 0)
                 AND
                 (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
                         (r.NAME = 'user defined value'))) = 0);
  END_REPEAT;

  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **representation** that has a **descriptive_representation_item** with **name** ‘authorization classification’ and **description** ‘user defined’ in its set of **items** shall have another **descriptive_-representation_item** with **name** ‘user defined value’ in its set of **items**.

5.2.4.235 user_defined_value_description_required_for_compartment_insulation

The **user_defined_value_description_required** rule specifies a **representation** that has a **descriptive_representation_item** with **name** 'compartment insulation' and **description** 'user defined' to have another **descriptive_representation_item** with **name** 'user defined value'.

EXPRESS specification:

```
*)
RULE user_defined_value_description_required_for_compartment_insulation
FOR (REPRESENTATION);

  LOCAL
    rep_set:   SET OF representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
    violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |

('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
                          IN TYPEOF(r)) AND
                         (r.NAME = 'compartment insulation') AND
                         (r\DESCRIPTIVE_REPRESENTATION_ITEM.DESCRIPTION =
'user defined'))) > 0)
```

```
                AND
                (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
                        (r.NAME = 'user defined value'))) = 0);
  END_REPEAT;

  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **representation** that has a **descriptive_representation_item** with **name** ‘compartment insulation’ and **description** ‘user defined’ in its set of **items** shall have another **descriptive_-representation_item** with **name** ‘user defined value’ in its set of **items**.

5.2.4.236 user_defined_value_description_required_for_compartment_noise_category

The **user_defined_value_description_required** rule specifies a **representation** that has a **descriptive_representation_item** with **name** 'compartment noise category' and **description** 'user defined' to have another **descriptive_representation_item** with **name** 'user defined value'.

EXPRESS specification:

```
*)
RULE user_defined_value_description_required_for_compartment_noise_category
FOR (REPRESENTATION);

  LOCAL
    rep_set:   SET OF representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
    violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |

('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
                          IN TYPEOF(r)) AND
                         (r.NAME = 'compartment noise category') AND
                         (r\DESCRIPTIVE_REPRESENTATION_ITEM.DESCRIPTION =
'user defined'))) > 0)
                AND
                (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
                        (r.NAME = 'user defined value'))) = 0);
  END_REPEAT;

  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **representation** that has a **descriptive_representation_item** with **name** ‘compartment noise category’ and **description** ‘user defined’ in its set of **items** shall have another **descriptive_-representation_item** with **name** ‘user defined value’ in its set of **items**.

5.2.4.237 user_defined_value_description_required_for_compartment_safety_class

The **user_defined_value_description_required** rule specifies a **representation** that has a **descriptive_representation_item** with **name** 'compartment safety class' and **description** 'user defined' to have another **descriptive_representation_item** with **name** 'user defined value'.

EXPRESS specification:

```
*)
RULE user_defined_value_description_required_for_compartment_safety_class
FOR (REPRESENTATION);

  LOCAL
    rep_set:   SET OF representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
    violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |

('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
                          IN TYPEOF(r)) AND
                         (r.NAME = 'compartment safety class') AND
                         (r\DESCRIPTIVE_REPRESENTATION_ITEM.DESCRIPTION =
'user defined'))) > 0)
                 AND
                 (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
                         (r.NAME = 'user defined value'))) = 0);
  END_REPEAT;

  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **representation** that has a **descriptive_representation_item** with **name** ‘compartment safety class’ and **description** ‘user defined’ in its set of **items** shall have another **descriptive_-representation_item** with **name** ‘user defined value’ in its set of **items**.

5.2.4.238 user_defined_value_description_required_for_compartment_security

The **user_defined_value_description_required_for_compartment_security** rule specifies a **representation** that has a **descriptive_representation_item** with **name** 'compartment security classification' and **description** 'user defined' to have another **descriptive_representation_item** with **name** 'user defined value'.

EXPRESS specification:

```
*)
RULE user_defined_value_description_required_for_compartment_security
FOR (REPRESENTATION);

  LOCAL
    rep_set:   SET OF representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
    violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |

('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
                          IN TYPEOF(r)) AND
                         (r.NAME = 'compartment security classification')
AND (r\DESCRIPTIVE_REPRESENTATION_ITEM.DESCRIPTION = 'user defined'))) > 0)
AND (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
                         (r.NAME = 'user defined value'))) = 0);
  END_REPEAT;

  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **representation** that has a **descriptive_representation_item** with **name** ‘compartment security classification’ and **description** ‘user defined’ in its set of **items** shall have another **descriptive_representation_item** with **name** ‘user defined value’ in its set of **items**.

5.2.4.239 user_defined_value_description_required_for_compartment_tightness

The **user_defined_value_description_required** rule specifies a **representation** that has a **descriptive_representation_item** with **name** 'compartment tightness' and **description** 'user defined' to have another **descriptive_representation_item** with **name** 'user defined value'.

EXPRESS specification:

```
*)
RULE user_defined_value_description_required_for_compartment_tightness
FOR (REPRESENTATION);

  LOCAL
    rep_set:   SET OF representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
    violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |

('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
                          IN TYPEOF(r)) AND
                         (r.NAME = 'compartment tightness') AND
                         (r\DESCRIPTIVE_REPRESENTATION_ITEM.DESCRIPTION =
'user defined'))) > 0)
```



```
                AND
                (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
                        (r.NAME = 'user defined value'))) = 0);
  END_REPEAT;

  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **representation** that has a **descriptive_representation_item** with **name** ‘compartment tightness’ and **description** ‘user defined’ in its set of **items** shall have another **descriptive_-representation_item** with **name** ‘user defined value’ in its set of **items**.

5.2.4.240 user_defined_value_description_required_for_requirement_type

The **user_defined_value_description_required** rule specifies a **representation** that has a **descriptive_representation_item** with **name** 'requirement type' and **description** 'user defined' to have another **descriptive_representation_item** with **name** 'user defined value'.

EXPRESS specification:

```
*)
RULE user_defined_value_description_required_for_requirement_type
FOR (REPRESENTATION);

  LOCAL
    rep_set:   SET OF representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(REPRESENTATION) WHILE NOT violation;
    violation := (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |

('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
                         IN TYPEOF(r)) AND
                        (r.NAME = 'requirement type') AND
                        (r\DESCRIPTIVE_REPRESENTATION_ITEM.DESCRIPTION =
'user defined'))) > 0)
                AND
                (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
                        (r.NAME = 'user defined value'))) = 0);
  END_REPEAT;

  WHERE
    WR1: NOT violation;
END_RULE;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities.

Formal propositions:

WR1: Every **representation** that has a **descriptive_representation_item** with **name** 'requirement type' and **description** 'user defined' in its set of **items** shall have another **descriptive_-representation_item** with **name** 'user defined value' in its set of **items**.

5.2.4.241 version_creation_has_mandatory_attribute_description

The **version_creation_has_mandatory_attribute_description** rule specifies that for an instance of **action** with class id 'version creation' the optional attribute **description** is instantiated.

EXPRESS specification:

```
*)
RULE version_creation_has_mandatory_attribute_description
FOR (action);
LOCAL
    t1_set:  SET OF action := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

(* get all instances of action being classified as 'version creation' *)
t1_set := QUERY(i <* action |
 VALUE_IN(WHICH_CLASS(i), 'version creation'));

(* from all instances found above:
   find those for which attribute description is not instantiated
*)

violate := (SIZEOF(QUERY(k <* t1_set |
 NOT EXISTS (k.description))) > 0);

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

action: the set of all instances of **action** entities.

Formal propositions:

WR1: The optional attribute **description** shall exist for every instance of **action** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'version creation'.

5.2.4.242 version_deletion_has_mandatory_attribute_description

The **version_deletion_has_mandatory_attribute_description** rule specifies that for an instance of **action** with class id 'version deletion' the optional attribute **description** is instantiated.

EXPRESS specification:

```
*)
RULE version_deletion_has_mandatory_attribute_description
```

```
FOR (action);
LOCAL
    t1_set:  SET OF action := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

(* get all instances of action being classified as 'version deletion' *)
t1_set := QUERY(i <* action |
 VALUE_IN(WHICH_CLASS(i), 'version deletion'));

(* from all instances found above:
   find those for which attribute description is not instantiated
*)

violate := (SIZEOF(QUERY(k <* t1_set |
 NOT EXISTS (k.description))) > 0);

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

action: the set of all instances of **action** entities.

Formal propositions:

WR1: The optional attribute **description** shall exist for every instance of **action** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'version deletion'.

5.2.4.243 version_history_has_exactly_one_assigned_group

The **version_history_has_exactly_one_assigned_group** rule specifies that each instance of type **group** with class 'version history' shall be referenced by exactly one assignment of type **applied_-group_assignment** via attribute **assigned_group**.

EXPRESS specification:

```
*)
RULE version_history_has_exactly_one_assigned_group
FOR(applied_group_assignment, group);
  LOCAL
    t1_set:  SET OF group := [];
    set_1, set_2: SET OF applied_group_assignment := [];
    set_3: SET OF group_item := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

(* get all instances of group with class 'version history' *)
t1_set := QUERY(a <* group | VALUE_IN(WHICH_CLASS(a), 'version history'));

  (* for all instances found above *)
REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
(* get the instances of applied_group_assignment that have a role 'current
version' *)
   set_1 := QUERY(b <* applied_group_assignment |
           (b.assigned_group = t1_set[i]) AND
          (b.role.name = 'current version'));
```

```
(* get the instances of applied_group_assignment that have a role 'members'
*)
   set_2 := QUERY(c <* applied_group_assignment |
             (c.assigned_group = t1_set[i]) AND
             (c.role.name = 'members'));

(* there shall be only one assignment in each of both cases *)
   violate := ((SIZEOF(set_1) <> 1) OR (SIZEOF(set_2) <> 1));

   IF NOT violate THEN
(* find all instances with class versionable object which are shared by
      set_1[1].items and in set_2[1].items *)
      set_3 := set_1[1].items * set_2[1].items;

(* there must be at exactly one instance with class 'versionable object' *)
      violate := (SIZEOF(set_3) <> 1) OR
                  NOT (VALUE_IN(WHICH_CLASS(set_3[1]),
             'versionable object'));
   END_IF;
END_REPEAT;

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_group_assignment: the set of all instances of **applied_group_assignment** entities.

group: the set of all instances of **group** entities.

Formal propositions:

WR1: Every instance of **group** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'version history' shall be referenced from exactly one instance of type **applied_group_assignment** through attribute **assigned_group**.

5.2.4.244 version_history_is_referenced_by_at_least_one_versions

The **version_history_is_referenced_by_at_least_one_versions** rule specifies that each instance of type **group** with class 'version history' shall be referenced by at least one assignment of type **applied_group_assignment** with **role.name** 'versions' via attribute **assigned_group**

EXPRESS specification:

```
*)
RULE version_history_is_referenced_by_at_least_one_versions
FOR(applied_group_assignment, group);
  LOCAL
    t1_set:  SET OF group := [];
    a_set:  SET OF applied_group_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

t1_set := QUERY(a <* group | VALUE_IN(WHICH_CLASS(a), 'version history'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
```

```
(* get the instances of applied_group_assignment that have a role
'versions' *)
   a_set := QUERY(b <* applied_group_assignment |
           (b.assigned_group = t1_set[i]) AND
           (b.role.name = 'versions'));

(* there shall be at least one such instance *)
    violate := SIZEOF(a_set) < 1;
  END_REPEAT;

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_group_assignment: the set of all instances of **applied_group_assignment** entities.

group: the set of all instances of **group** entities

Formal propositions:

WR1: Every instance of **group** that has an **applied_classification_assignment** whose **assigned_class.name** equals 'version history' shall be referenced by one or more instances of type **applied_group_assignment** whose **role.name** equals 'versions' through attribute **assigned_group**.

5.2.4.245 version_history_referenced_by_exactly_one_current_version

The **version_history_referenced_by_exactly_one_current_version** rule specifies that each instance of type **group** with class 'version history' shall be referenced by exactly one assignment of type **applied_group_assignment** with **role** 'current version' via attribute **assigned_group**.

EXPRESS specification:

```
*)
RULE version_history_referenced_by_exactly_one_current_version
FOR(applied_group_assignment, group);
  LOCAL
    t1_set:  SET OF group := [];
    a_set:  SET OF applied_group_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

(* get all instances of group with class 'version history' *)
t1_set := QUERY(a <* group | VALUE_IN(WHICH_CLASS(a), 'version history'));

  (* for all instances found above *)
REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
(* get the instances of applied_group_assignment that have a role 'current
version' *)
   a_set := QUERY(b <* applied_group_assignment |
             (b.assigned_group = t1_set[i]) AND
             (b.role.name = 'current version'));

(* there shall be no such instances *)
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;

WHERE
```

```
    wr1: NOT violate;
END_RULE;
```

(*

Argument definitions:

applied_group_assignment: the set of all instances of **applied_group_assignment** entities.

group: the set of all instances of **group** entities.

Formal propositions:

WR1: Every instance of **group** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'version history' shall be referenced from exactly one instance of type **applied_group_assignment** whose **role** equals 'current version' through attribute **assigned_group**.

5.2.4.246 version_history_referenced_by_multiple_roles

The **version_history_referenced_by_multiple_roles** rule specifies that each instance of type **group** with class 'version history' shall be referenced by one or more instances of type **applied_group_assignment** with **role.name** equal 'versions', 'current version', or 'relationships' via attribute assigned_group.

EXPRESS specification:

```
*)
RULE version_history_referenced_by_multiple_roles
FOR(applied_group_assignment, group);
  LOCAL
    t1_set:  SET OF group := [];
    a_set:  SET OF applied_group_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
t1_set := QUERY(a <* group | VALUE_IN(WHICH_CLASS(a), 'version history'));
REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
   a_set := QUERY(b <* applied_group_assignment |
    (b.assigned_group = t1_set[i]) AND NOT (b.role.name IN ['versions',
'current version', 'relationships']));
    violate := SIZEOF(a_set) < 1;
END_REPEAT;
WHERE
    wr1: NOT violate;
END_RULE;
```

(*

Argument definitions:

applied_group_assignment: the set of all instances of **applied_group_assignment** entities.

group: the set of all instances of **group** entities

Formal propositions:

WR1: Every instance of **group** that has an **applied_classification_assignment** whose **assigned_class** identifies an entity with attribute **name** equal 'version history' shall be only

referenced by instances of type **applied_group_assignment** whose **role.name** equals 'versions', 'current version', or 'relationships' through attribute **assigned_group**.

5.2.4.247 version_modification_has_mandatory_attribute_description

The **version_modification_has_mandatory_attribute_description** rule specifies that for an instance of **action** with class id 'version modification' the optional attribute **description** is instantiated.

EXPRESS specification:

```
*)
RULE version_modification_has_mandatory_attribute_description
FOR (action);
LOCAL
    t1_set:  SET OF action := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

(* get all instances of action being classified as 'version modification'
*)
t1_set := QUERY(i <* action |
 VALUE_IN(WHICH_CLASS(i), 'version modification'));

(* from all instances found above:
   find those for which attribute description is not instantiated
*)

violate := (SIZEOF(QUERY(k <* t1_set | NOT EXISTS (k.description))) > 0);

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

action: the set of all instances of **action** entities.

Formal propositions:

WR1: The optional attribute **description** shall exist for every instance of **action** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'version modification'.

5.2.4.248 version_relationship_associates_with_versionable_object

The **version_relationship_associates_with_versionable_object** rule specifies an instance of **identification_assignment_relationship** with class 'version relationship' only relates instances of type **applied_identification_assignment** whose attribute **items** points to instances of class 'versionable object'.

EXPRESS specification:

```
*)
RULE version_relationship_associates_with_versionable_object
FOR (applied_identification_assignment);
LOCAL
```

```
    violate: LOGICAL := FALSE;
END_LOCAL;
REPEAT i := 1 TO HIINDEX(applied_identification_assignment) BY 1 WHILE NOT
violate;
 IF ( (SIZEOF(USEDIN(applied_identification_assignment[i],
('SHIP_ARRANGEMENT_SCHEMA.IDENTIFICATION_ASSIGNMENT_RELATIONSHIP.' +
'RELATING_IDENTIFICATION_ASSIGNMENT'))) > 0) OR
      (SIZEOF(USEDIN(applied_identification_assignment[i],
      ('SHIP_ARRANGEMENT_SCHEMA.IDENTIFICATION_ASSIGNMENT_RELATIONSHIP.' +
'RELATED_IDENTIFICATION_ASSIGNMENT'))) > 0) ) THEN
  REPEAT j := 1 to HIINDEX(applied_identification_assignment[i].items) BY 1
WHILE NOT violate;
   violate := NOT
VALUE_IN(which_class(applied_identification_assignment[i].items[j]),
'versionable object');
  END_REPEAT;
 END_IF;

END_REPEAT;
WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_identification_assignment: the set of all instances of **applied_identification_assignment** entities.

Formal propositions:

WR1: Every instance of **identification_assignment_relationship** that has an **applied_classification_assignment** whose **assigned_class.name** equals 'version relationship' shall only reference instances of **applied_identification_assignment** whose **items** attribute value points to an instance that has an **applied_classification_assignment** whose **assigned_class.name** equals 'versionable object'.

5.2.4.249 version_relationship_has_mandatory_attribute_description

The **version_relationship_has_mandatory_attribute_description** rule specifies that for an instance of **identification_assignment_relationship** with class id 'version relationship' the optional attribute **description** is instantiated.

EXPRESS specification:

```
*)
RULE version_relationship_has_mandatory_attribute_description
FOR (identification_assignment_relationship);
LOCAL
    t1_set:  SET OF identification_assignment_relationship := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

(* get all instances of identification_assignment_relationship being
classified as 'version relationship' *)
t1_set := QUERY(i <* identification_assignment_relationship |
VALUE_IN(WHICH_CLASS(i), 'version relationship'));
```



```
(* from all instances found above:
   find those for which attribute description is not instantiated
*)

violate := (SIZEOF(QUERY(k <* t1_set | NOT EXISTS (k.description))) > 0);

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

identification_assignment_relationship: the set of all instances of **identification_assignment_-relationship** entities.

Formal propositions:

WR1: The optional attribute **description** shall exist for every instance of **identification_assignment_relationship** that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal to 'version relationship'.

5.2.4.250 version_relationship_has_unique_versions

The **version_relationship_has_unique_versions** rule specifies that for all instances of **identification_assignment_relationship** with class 'version relationship', the attributes **successor** and **predecessor** must reference different entities of class ‘versionable object’.

EXPRESS specification:

```
*)
RULE version_relationship_has_unique_versions
FOR (identification_assignment_relationship);
LOCAL
    t1_set:  SET OF identification_assignment_relationship := [];
    violate: LOGICAL := FALSE;
END_LOCAL;
   t1_set := QUERY(a <* identification_assignment_relationship |
   VALUE_IN(WHICH_CLASS(a), 'version relationship'));
REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
    violate :=
          ( t1_set[i].relating_identification_assignment.assigned_id =
            t1_set[i].related_identification_assignment.assigned_id );
END_REPEAT;
WHERE
    WR1: NOT violate;
END_RULE;

(*
```

Argument definitions:

identification_assignment_relationship: the set of all instances of **identification_assignment_-relationship** entities.

Formal propositions:

WR1: The **assigned_id** values for the **related_identification_assignment** and the **relating_-identification_assignment** attributes of every instance of **identification_assignment_relationship**

that has an **applied_classification_assignment** whose **assigned_class** is a **group** with attribute **name** equal 'version relationship' shall be distinct.

5.2.4.251 versionable_object_has_one_version_id

The **versionable_object_has_one_version_id** rule specifies that a type that is referenced by an **applied_identification_assignment** whose **role.name** is 'version identifier' has exactly one reference of this type.

EXPRESS specification:

```
*)
RULE versionable_object_has_one_version_id
FOR(APPLIED_IDENTIFICATION_ASSIGNMENT);
  LOCAL
    version_ids:         SET OF APPLIED_IDENTIFICATION_ASSIGNMENT := [];
    versionable_objects: BAG OF identification_item := [];
    duplicate:           LOGICAL := FALSE;
  END_LOCAL;

  version_ids := QUERY(i <* APPLIED_IDENTIFICATION_ASSIGNMENT |
                       i.ROLE.NAME = 'version identifier');

  REPEAT i := 1 TO HIINDEX(version_ids);
    versionable_objects := versionable_objects + version_ids[i].items;
  END_REPEAT;
   REPEAT i := 1 TO HIINDEX(versionable_objects) WHILE NOT duplicate;
    REPEAT j := i + 1 TO HIINDEX(versionable_objects) WHILE NOT duplicate;
      duplicate := versionable_objects[i] :=: versionable_objects[j];
    END_REPEAT;
  END_REPEAT;
  WHERE
    WR1: NOT duplicate;
END_RULE;

(*
```

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment** entities.

Formal propositions:

WR1: Every entity instance shall have zero or one **applied_identification_assignment** whose attribute **role.name** is 'version identifier'.

5.2.4.252 versioned_action_request_with_identification_assignment

The **versioned_action_request_with_identification_assignment** rule specifies a list of entities that require an identification. The identification is defined by the **applied_identification_assignment** entity.

EXPRESS specification:

```
*)
RULE versioned_action_request_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
```



```
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF versioned_action_request := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := ['change request'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
      t2_set := QUERY ( j <* t2_set |
              j.role.name = 'globally unambiguous identifier');
violation := NOT (SIZEOF(T2_SET) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;
```

(*

Argument definitions:

applied_classification_assignment: the set of all instances of **applied_classification_assignment**.

Formal propositions:

WR1: Every instance of **version_action_request** that is referenced by an **applied_classification_-assignment** whose **assigned_class** has a **name** attribute of value 'change request' shall require an **applied_identification_assignment** to define the instance identifier.

5.2.4.253 versions_is_referenced_by_at_least_one_version_history

The **versions_is_referenced_by_at_least_one_version_history** rule specifies that each instance of type **group** of class 'versions' shall be referenced by at least one assignment of type **applied_-group_assignment** with **role.name** equal 'version history' via attribute **assigned_group**

EXPRESS specification:

```
*)
RULE versions_is_referenced_by_at_least_one_version_history
FOR(applied_group_assignment, group);
  LOCAL
    t1_set:  SET OF group := [];
    a_set:  SET OF applied_group_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

t1_set := QUERY(a <* group |
 VALUE_IN(WHICH_CLASS(a), 'versions'));
```

```
REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;

   a_set := QUERY(b <* applied_group_assignment |
      (b.assigned_group = t1_set[i]) AND
      (b.role.name = 'version history'));

    violate := SIZEOF(a_set) < 1;
  END_REPEAT;

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

applied_group_assignment: the set of all instances of **applied_group_assignment** entities.

group: the set of all instances of **group** entities

Formal propositions:

WR1: Every instance of **group** that has an **applied_classification_assignment** whose **assigned_class** identifies an entity with attribute **name** equal 'versions' shall be referenced by one or more instances of type **applied_group_assignment** whose **role.name** equals 'version history' through attribute **assigned_group**.

5.2.5 Ship arrangement function definitions

5.2.5.1 which_class

The function **which_class** determines the **applied_classification_assignment**s pointing to an instance of an arbitrary type and returns the class ids in a string list.

EXPRESS specification:

```
*)
FUNCTION WHICH_CLASS(T: GENERIC): LIST OF STRING;
  LOCAL
     elements: BAG OF APPLIED_CLASSIFICATION_ASSIGNMENT;
     class_list: LIST OF STRING :=[];
  END_LOCAL;
  elements :=   USEDIN(T,

'SHIP_ARRANGEMENT_SCHEMA.APPLIED_CLASSIFICATION_ASSIGNMENT.ITEMS');
  REPEAT i:=1 TO HIINDEX(elements);
 IF (elements[i]\classification_assignment.role.name = 'class membership')
THEN
  class_list := class_list +
  elements[i]\classification_assignment.assigned_class\group.name;
 END_IF;
  END_REPEAT;
  RETURN(class_list);
END_FUNCTION;
(*

*)
```



```
END_SCHEMA;
(*
```

6 Conformance requirements

Conformance to this part of ISO 10303 includes satisfying the requirements stated in this part, the requirements of the implementation method(s) supported, and the relevent requirements of the normative references.

An implementation shall support at least one of the following implementation methods:

— ISO 10303-21;

— ISO 10303-22;

— ISO 10303-28.

Requirements with respect to implementation methods-specific requirements are specified in annex C.

The Protocol Implementation Conformance Statement (PICS) proforma lists the options or the combinations of options that may be included in the implementation. The PICS proforma is provided in annex D.

This part of ISO 10303 provides for a number of options that may be supported by an implementation. These options have been grouped into the following conformance classes:

— Class 1 is a conformance class to exchange early design data regarding ship arrangement;

— Class 2 is a conformance class to exchange detail design CAD geometry data with limited properties;

— Class 3 is a conformance class to exchange detail design data regarding ship arrangement;

— Class 4 is a conformance class to exchange operational data regarding ship arrangement;

— Class 5 is a conformance class to exchange analysis data regarding ship arrangement.

Support for a particular conformance class requires support of all the options specified in that class.

Comformance to a particular class requires that all AIM elements defined as part of that class be supported. Table 1 defines the UoFs included in each class. Table 2 defines the classes to which each AIM element belongs.

NOTE ISO 10303-32 describes the conformance assessment process.

Table 1 — UoFs in conformance classes

Unit of Functionality	Conformance class				
	Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 5**
arrangement_relationships	X		X	X	X
cargoes			X	X	X
coatings			X	X	
compartment_design_definitions		X	X	X	X
compartment_properties			X	X	X
compartment_requirements	X		X		
configuration_management	X	X	X	X	X
damaged_stability			X		X
definitions	X	X	X	X	X
external_references	X	X	X	X	X
items	X	X	X	X	X
loading_conditions			X	X	X
location_concepts		X	X	X	X
product_structures	X	X	X	X	X
ship_general_characteristics	X	X	X	X	X
ship_measures	X	X	X	X	X
spaces	X	X	X	X	X
surface_representations		X	X		X
tonnage	X		X	X	X
weights			X		X

Table 2 — Conformance class elements

AIM Elements	Conformance class				
	Class 1	Class 2	Class 3	Class 4	Class 5
action	X	X	X	X	X
action_assignment	X	X	X	X	X
action_method	X	X	X	X	X
action_request_assignment	X	X	X	X	X
action_request_solution	X	X	X	X	X
address	X	X	X	X	X
advanced_face		X	X		X
application_context	X	X	X	X	X
application_context_element	X	X	X	X	X
application_protocol_definition	X	X	X	X	X
applied_action_assignment	X	X	X	X	X
applied_action_request_assignment	X	X	X	X	X
applied_approval_assignment	X	X	X	X	X
applied_classification_assignment	X	X	X	X	X
applied_date_and_time_assignment	X	X	X	X	X
applied_document_reference	X	X	X	X	X
applied_effectivity_assignment	X	X	X	X	X
applied_external_identification_assignment	X	X	X	X	X
applied_group_assignment	X	X	X	X	X
applied_identification_assignment	X	X	X	X	X
applied_organization_assignment	X	X	X	X	X
applied_person_and_organization_assignment	X	X	X	X	X
applied_person_assignment	X	X	X	X	X
approval	X	X	X	X	X

Table 2 — Conformance class elements (continued)

approval_assignment	X	X	X	X	X
approval_date_time	X	X	X	X	X
approval_person_organization	X	X	X	X	X
approval_role	X	X	X	X	X
approval_status	X	X	X	X	X
axis1_placement		X	X		X
axis2_placement_2d		X	X		X
axis2_placement_3d		X	X		X
b_spline_curve		X	X		X
b_spline_curve_with_knots		X	X		X
b_spline_surface		X	X		X
b_spline_surface_with_knots		X	X		X
bezier_curve		X	X		X
bezier_surface		X	X		X
bounded_curve		X	X		X
bounded_pcurve		X	X		X
bounded_surface		X	X		X
bounded_surface_curve		X	X		X
calendar_date	X	X	X	X	X
cartesian_point		X	X		X
cartesian_transformation_operator		X	X		X
cartesian_transformation_operator_3d		X	X		X
characterized_object	X	X	X	X	X
circle		X	X		X
class	X	X	X	X	X

Table 2 — Conformance class elements (continued)

classification_assignment	X	X	X	X	X
classification_role	X	X	X	X	X
closed_shell		X	X		X
composite_curve		X	X		X
composite_curve_on_surface		X	X		X
composite_curve_segment		X	X		X
compound_representation_item	X	X	X	X	X
conic		X	X		X
conical_surface		X	X		X
connected_face_set		X	X		X
context_dependent_unit	X	X	X	X	X
coordinated_universal_time_offset	X	X	X	X	X
curve		X	X		X
curve_replica		X	X		X
cylindrical_surface		X	X		X
date	X	X	X	X	X
date_and_time	X	X	X	X	X
date_and_time_assignment	X	X	X	X	X
date_time_role	X	X	X	X	X
definitional_representation	X	X	X	X	X
degenerate_pcurve		X	X		X
degenerate_toroidal_surface		X	X		X
derived_unit	X	X	X	X	X
derived_unit_element	X	X	X	X	X
description_attribute	X	X	X	X	X
descriptive_representation_item	X	X	X	X	X

Table 2 — Conformance class elements (continued)

dimensional_exponents	X	X	X	X	X
direction		X	X		X
document	X	X	X	X	X
document_reference	X	X	X	X	X
document_representation_type	X	X	X	X	X
document_type	X	X	X	X	X
document_usage_constraint	X	X	X	X	X
edge		X	X		X
edge_curve		X	X		X
edge_loop		X	X		X
effectivity	X	X	X	X	X
effectivity_assignment	X	X	X	X	X
elementary_surface		X	X		X
ellipse		X	X		X
evaluated_degenerate_pcurve		X	X		X
executed_action	X	X	X	X	X
external_identification_assignment	X	X	X	X	X
external_source	X	X	X	X	X
external_source_relationship	X	X	X	X	X
face		X	X		X
face_based_surface_model		X	X		X
face_bound		X	X		X
face_outer_bound		X	X		X
face_surface		X	X		X
founded_item	X	X	X	X	X
functionally_defined_transformation	X	X	X	X	X

Table 2 — Conformance class elements (continued)

geometric_representation_context		X	X		X
geometric_representation_item		X	X		X
global_uncertainty_assigned_context	X	X	X	X	X
global_unit_assigned_context	X	X	X	X	X
group	X	X	X	X	X
group_assignment	X	X	X	X	X
group_relationship	X	X	X	X	X
hyperbola		X	X		X
id_attribute	X	X	X	X	X
identification_assignment	X	X	X	X	X
identification_assignment_relationship	X	X	X	X	X
identification_role	X	X	X	X	X
intersection_curve		X	X		X
item_defined_transformation	X	X	X	X	X
length_unit	X	X	X	X	X
line		X	X		X
local_time	X	X	X	X	X
loop		X	X		X
luminous_intensity_unit	X		X	X	X
mapped_item	X	X	X	X	X
mass_unit	X		X	X	X
measure_with_unit	X		X	X	X
name_attribute	X	X	X	X	X
named_unit	X	X	X	X	X
non_manifold_surface_shape_representation		X	X		X
object_role	X	X	X	X	X
offset_curve_3d		X	X		X

Table 2 — Conformance class elements (continued)

offset_surface		X	X		X
open_shell		X	X		X
ordinal_date	X	X	X	X	X
organization	X	X	X	X	X
organization_assignment	X	X	X	X	X
organization_role	X	X	X	X	X
organizational_address	X	X	X	X	X
organizational_project	X	X	X	X	X
oriented_closed_shell		X	X		X
oriented_edge		X	X		X
oriented_face		X	X		X
oriented_open_shell		X	X		X
oriented_path		X	X		X
oriented_surface		X	X		X
parabola		X	X		X
parametric_representation_context	X	X	X	X	X
path		X	X		X
pcurve		X	X		X
person	X	X	X	X	X
person_and_organization	X	X	X	X	X
person_and_organization_assignment	X	X	X	X	X
person_and_organization_role	X	X	X	X	X
person_assignment	X	X	X	X	X
person_role	X	X	X	X	X
personal_address	X	X	X	X	X
placement		X	X		X

Table 2 — Conformance class elements (continued)

plane		X	X		X
plane_angle_unit	X	X	X	X	X
point		X	X		X
point_on_curve		X	X		X
point_on_surface		X	X		X
poly_loop		X	X		X
polyline		X	X		X
product	X	X	X	X	X
product_category	X	X	X	X	X
product_context	X	X	X	X	X
product_definition	X	X	X	X	X
product_definition_context	X	X	X	X	X
product_definition_formation	X	X	X	X	X
product_definition_relationship	X	X	X	X	X
product_definition_shape	X	X	X	X	X
product_definition_with_associated_documents	X	X	X	X	X
product_related_product_category	X	X	X	X	X
property_definition	X	X	X	X	X
property_definition_relationship	X	X	X	X	X
property_definition_representation	X	X	X	X	X
quasi_uniform_curve		X	X		X
quasi_uniform_surface		X	X		X
ratio_unit	X	X	X	X	X
rational_b_spline_curve		X	X		X
rational_b_spline_surface		X	X		X
representation	X	X	X	X	X

Table 2 — Conformance class elements (continued)

representation_context	X	X	X	X	X
representation_item	X	X	X	X	X
representation_map	X	X	X	X	X
representation_relationship	X	X	X	X	X
role_association	X	X	X	X	X
seam_curve	X	X	X	X	X
serial_numbered_effectivity	X	X	X	X	X
shape_aspect	X	X	X	X	X
Shape_definition_representation	X	X	X	X	X
shape_representation		X	X		X
si_unit	X	X	X	X	X
spherical_surface		X	X		X
surface		X	X		X
surface_curve		X	X		X
surface_of_linear_extrusion		X	X		X
surface_of_revolution		X	X		X
surface_replica		X	X		X
swept_surface		X	X		X
thermodynamic_temperature_unit	X		X	X	X
time_unit	X		X	X	X
topological_representation_item	X	X	X	X	X
toroidal_surface		X	X		X
uncertainty_measure_with_unit	X	X	X	X	X
uniform_curve		X	X		X
uniform_surface		X	X		X
value_representation_item	X	X	X	X	X

Table 2 — Conformance class elements (concluded)

vector		X	X		X
versioned_action_request	X	X	X	X	X
vertex		X	X		X
vertex_loop		X	X		X
vertex_point		X	X		X
week_of_year_and_day_date	X	X	X	X	X

Annex A

(normative)

AIM EXPRESS expanded listing

The following EXPRESS is the expanded form of the short form schema given in 5.2. In the event of any discrepancy between the short form and this expanded listing, the expanded listing shall be used.

```
SCHEMA ship_arrangement_schema;

CONSTANT dummy_gri : geometric_representation_item :=
  representation_item('') ||  geometric_representation_item();
  dummy_tri : topological_representation_item :=
  representation_item('') || topological_representation_item();
END_CONSTANT;

TYPE action_item = SELECT
    (action_request_solution,
     document,
     executed_action,
     group,
     product,
     product_definition,
     product_definition_relationship,
     product_definition_shape,
     product_related_product_category,
     property_definition
     );
END_TYPE; -- action_item

TYPE action_request_item = SELECT
  (action,
   executed_action);
END_TYPE; -- action_request_item

TYPE ahead_or_behind = ENUMERATION OF
  (exact,
   ahead,
   behind);
END_TYPE; -- ahead_or_behind

TYPE amount_of_substance_measure = REAL;
END_TYPE; -- amount_of_substance_measure

TYPE approval_item = SELECT(
   product_definition,
   product_definition_shape,
   product_related_product_category,
   property_definition
   );
END_TYPE; -- approval_item

TYPE area_measure = REAL;
END_TYPE; -- area_measure

TYPE axis2_placement = SELECT
  (axis2_placement_2d,
   axis2_placement_3d);
END_TYPE; -- axis2_placement
```

```
TYPE b_spline_curve_form = ENUMERATION OF
  (elliptic_arc,
   polyline_form,
   parabolic_arc,
   circular_arc,
   unspecified,
   hyperbolic_arc);
END_TYPE; -- b_spline_curve_form

TYPE b_spline_surface_form = ENUMERATION OF
  (surf_of_linear_extrusion,
   plane_surf,
   generalised_cone,
   toroidal_surf,
   conical_surf,
   spherical_surf,
   unspecified,
   ruled_surf,
   surf_of_revolution,
   cylindrical_surf,
   quadric_surf);
END_TYPE; -- b_spline_surface_form

TYPE characterized_definition = SELECT
  (characterized_object,
   characterized_product_definition,
   shape_definition);
END_TYPE; -- characterized_definition

TYPE characterized_product_definition = SELECT
  (product_definition,
   product_definition_relationship);
END_TYPE; -- characterized_product_definition

TYPE classification_item = SELECT
  (action,
   action_request_solution,
   applied_action_request_assignment,
   approval,
   compound_representation_item,
   document,
   executed_action,
   external_source,
   group,
   identification_assignment_relationship,
   product,
   product_definition,
   product_definition_relationship,
   product_definition_shape,
   product_related_product_category,
   property_definition,
   property_definition_representation,
   representation,
   shape_aspect,
   versioned_action_request);
END_TYPE; -- classification_item

TYPE compound_item_definition = SELECT
  (list_representation_item,
   set_representation_item);
END_TYPE; -- compound_item_definition

TYPE context_dependent_measure = REAL;
END_TYPE; -- context_dependent_measure
```

```
TYPE count_measure = NUMBER;
END_TYPE; -- count_measure

TYPE curve_on_surface = SELECT
  (pcurve,
   surface_curve,
   composite_curve_on_surface);
END_TYPE; -- curve_on_surface

TYPE date_and_time_item = SELECT
  (action,
   action_request_solution,
   executed_action,
   product_definition,
   property_definition,
   property_definition_representation,
   versioned_action_request);
END_TYPE; -- date_and_time_item

TYPE date_time_select = SELECT
  (date,
   local_time,
   date_and_time);
END_TYPE; -- date_time_select

TYPE day_in_month_number = INTEGER;
WHERE
  wr1: (1 <= SELF) AND (SELF <= 31);
END_TYPE; -- day_in_month_number

TYPE day_in_week_number = INTEGER;
WHERE
  wr1: (1 <= SELF) AND (SELF <= 7);
END_TYPE; -- day_in_week_number

TYPE day_in_year_number = INTEGER;
WHERE
  wr1: (1 <= SELF) AND (SELF <= 366);
END_TYPE; -- day_in_year_number

TYPE description_attribute_select = SELECT
  (action_request_solution,
   application_context,
   approval_role,
   date_time_role,
   effectivity,
   external_source,
   organization_role,
   person_and_organization_role,
   person_and_organization,
   person_role,
   property_definition_representation,
   representation);
END_TYPE; -- description_attribute_select

TYPE dimension_count = INTEGER;
WHERE
  wr1: SELF > 0;
END_TYPE; -- dimension_count

TYPE document_reference_item = SELECT
  (action,
   product,
   product_definition,
   property_definition);
```

```
END_TYPE; -- document_reference_item

TYPE electric_current_measure = REAL;
END_TYPE; -- electric_current_measure

TYPE effectivity_item = SELECT
   (product_definition,
   product_definition_shape,
   product_related_product_category,
   property_definition);
END_TYPE; -- effectivity_item

TYPE external_identification_item = SELECT
  (action,
   document,
   product,
   product_definition,
   property_definition,
   shape_aspect);
END_TYPE; -- external_identification_item

TYPE founded_item_select = SELECT
  (founded_item,
   representation_item);
END_TYPE; -- founded_item_select

TYPE group_item = SELECT
  (applied_external_identification_assignment,
   approval,
   document,
   group,
   identification_assignment_relationship,
   product,
   product_definition,
   product_definition_relationship,
   product_definition_shape,
   product_related_product_category,
   property_definition);
END_TYPE; -- group_item

TYPE hour_in_day = INTEGER;
WHERE
  wr1: (0 <= SELF) AND (SELF < 24);
END_TYPE; -- hour_in_day

TYPE id_attribute_select = SELECT
  (action,
   address,
   product_category,
   property_definition,
   shape_aspect,
   application_context,
   group,
   organizational_project,
   representation);
END_TYPE; -- id_attribute_select

TYPE identification_item = SELECT
  (action,
   action_request_solution,
   document,
   executed_action,
   group,
   product,
   product_definition,
```



```
   product_definition_relationship,
   product_definition_shape,
   product_related_product_category,
   property_definition,
   versioned_action_request);
END_TYPE; -- identification_item

TYPE identifier = STRING;
END_TYPE; -- identifier

TYPE knot_type = ENUMERATION OF
  (uniform_knots,
   quasi_uniform_knots,
   piecewise_bezier_knots,
   unspecified);
END_TYPE; -- knot_type

TYPE label = STRING;
END_TYPE; -- label

TYPE length_measure = REAL;
END_TYPE; -- length_measure

TYPE list_of_reversible_topology_item = LIST [0:?] OF
            reversible_topology_item;
END_TYPE; -- list_of_reversible_topology_item

TYPE list_representation_item = LIST [1:?] OF representation_item;
END_TYPE; -- list_representation_item

TYPE luminous_intensity_measure = REAL;
END_TYPE; -- luminous_intensity_measure

TYPE mass_measure = REAL;
END_TYPE; -- mass_measure

TYPE measure_value = SELECT
  (length_measure,
   mass_measure,
   time_measure,
   electric_current_measure,
   thermodynamic_temperature_measure,
   amount_of_substance_measure,
   luminous_intensity_measure,
   plane_angle_measure,
   solid_angle_measure,
   area_measure,
   volume_measure,
   ratio_measure,
   parameter_value,
   context_dependent_measure,
   positive_length_measure,
   positive_plane_angle_measure,
   count_measure);
END_TYPE; -- measure_value

TYPE minute_in_hour = INTEGER;
WHERE
  wr1: (0 <= SELF) AND (SELF <= 59);
END_TYPE; -- minute_in_hour

TYPE month_in_year_number = INTEGER;
WHERE
  wr1: (1 <= SELF) AND (SELF <= 12);
END_TYPE; -- month_in_year_number
```

```
TYPE name_attribute_select = SELECT
  (action_request_solution,
   address,
   derived_unit,
   effectivity,
   person_and_organization,
   product_definition,
   property_definition_representation);
END_TYPE; -- name_attribute_select

TYPE organization_item = SELECT
  (document,
   product_definition,
   property_definition);
END_TYPE; -- organization_item

TYPE parameter_value = REAL;
END_TYPE; -- parameter_value

TYPE pcurve_or_surface = SELECT
  (pcurve,
   surface);
END_TYPE; -- pcurve_or_surface

TYPE person_and_organization_item = SELECT
  (action,
   action_request_solution,
   document,
   executed_action,
   versioned_action_request);
END_TYPE; -- person_and_organization_item

TYPE person_item = SELECT
  (document);
END_TYPE; -- person_item

TYPE person_organization_select = SELECT
  (person,
   organization,
   person_and_organization);
END_TYPE; -- person_organization_select

TYPE plane_angle_measure = REAL;
END_TYPE; -- plane_angle_measure

TYPE positive_length_measure = length_measure;
WHERE
  wr1: SELF > 0;
END_TYPE; -- positive_length_measure

TYPE positive_plane_angle_measure = plane_angle_measure;
WHERE
  wr1: SELF > 0;
END_TYPE; -- positive_plane_angle_measure

TYPE preferred_surface_curve_representation = ENUMERATION OF
  (pcurve_s2,
   pcurve_s1,
   curve_3d);
END_TYPE; -- preferred_surface_curve_representation

TYPE product_or_formation_or_definition = SELECT
  (product,
   product_definition_formation,
```



```
   product_definition);
END_TYPE; -- product_or_formation_or_definition

TYPE ratio_measure = REAL;
END_TYPE; -- ratio_measure

TYPE represented_definition = SELECT
  (property_definition,
   property_definition_relationship,
   shape_aspect);
END_TYPE; -- represented_definition

TYPE reversible_topology = SELECT
  (reversible_topology_item,
   list_of_reversible_topology_item,
   set_of_reversible_topology_item);
END_TYPE; -- reversible_topology

TYPE reversible_topology_item = SELECT
  (edge,
   path,
   face,
   face_bound,
   closed_shell,
   open_shell);
END_TYPE; -- reversible_topology_item

TYPE role_select = SELECT
  (action_request_assignment,
   approval_assignment,
   approval_date_time,
   document_reference,
   effectivity_assignment,
   group_assignment);
END_TYPE; -- role_select

TYPE second_in_minute = REAL;
WHERE
  wr1: (0 <= SELF) AND (SELF <= 60);
END_TYPE; -- second_in_minute

TYPE set_of_reversible_topology_item = SET [0:?] OF
           reversible_topology_item;
END_TYPE; -- set_of_reversible_topology_item

TYPE set_representation_item = SET [1:?] OF representation_item;
END_TYPE; -- set_representation_item

TYPE shape_definition = SELECT
  (product_definition_shape,
   shape_aspect);
END_TYPE; -- shape_definition

TYPE shell = SELECT
      (open_shell,
      closed_shell);
END_TYPE;

TYPE si_prefix = ENUMERATION OF
  (exa,
   pico,
   mega,
   femto,
   atto,
   centi,
   nano,
```

```
   hecto,
   micro,
   tera,
   giga,
   milli,
   peta,
   deci,
   kilo,
   deca);
END_TYPE; -- si_prefix

TYPE si_unit_name = ENUMERATION OF
  (hertz,
   degree_celsius,
   siemens,
   sievert,
   lux,
   watt,
   ohm,
   second,
   becquerel,
   pascal,
   henry,
   tesla,
   volt,
   joule,
   kelvin,
   ampere,
   gram,
   steradian,
   mole,
   lumen,
   gray,
   candela,
   farad,
   radian,
   newton,
   metre,
   weber,
   coulomb);
END_TYPE; -- si_unit_name

TYPE solid_angle_measure = REAL;
END_TYPE; -- solid_angle_measure

TYPE source_item = SELECT
  (identifier);
END_TYPE; -- source_item

TYPE surface_model = SELECT
  (face_based_surface_model);
END_TYPE; -- surface_model

TYPE text = STRING;
END_TYPE; -- text

TYPE thermodynamic_temperature_measure = REAL;
END_TYPE; -- thermodynamic_temperature_measure

TYPE time_measure = REAL;
END_TYPE; -- time_measure

TYPE transformation = SELECT
  (item_defined_transformation,
```

```
   functionally_defined_transformation);
END_TYPE; -- transformation

TYPE transition_code = ENUMERATION OF
  (discontinuous,
   cont_same_gradient_same_curvature,
   cont_same_gradient,
   continuous);
END_TYPE; -- transition_code

TYPE unit = SELECT
  (named_unit,
   derived_unit);
END_TYPE; -- unit

TYPE vector_or_direction = SELECT
  (vector,
   direction);
END_TYPE; -- vector_or_direction

TYPE volume_measure = REAL;
END_TYPE; -- volume_measure

TYPE week_in_year_number = INTEGER;
WHERE
  wr1: (1 <= SELF) AND (SELF <= 53);
END_TYPE; -- week_in_year_number

TYPE year_number = INTEGER;
END_TYPE; -- year_number

ENTITY action;
    name          : label;
    description   : OPTIONAL text;
    chosen_method : action_method;
  DERIVE
    id : identifier := get_id_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
END_ENTITY; -- action

ENTITY  action_assignment
  ABSTRACT SUPERTYPE;
    assigned_action : action;
  DERIVE
    role : object_role := get_role(SELF);
  WHERE
      wr1: (SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
                'ROLE_ASSOCIATION.ITEM_WITH_ROLE')) <= 1);
END_ENTITY; -- action_assignment

ENTITY action_method;
    name        : label;
    description : OPTIONAL text;
    consequence : text;
    purpose     : text;
END_ENTITY; -- action_method

ENTITY action_request_assignment
  ABSTRACT SUPERTYPE;
    assigned_action_request : versioned_action_request;
  DERIVE
    role : object_role := get_role(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
```

```
            'ROLE_ASSOCIATION.ITEM_WITH_ROLE')) <= 1;
END_ENTITY; -- action_request_assignment

ENTITY action_request_solution;
    method  : action_method;
    request : versioned_action_request;
  DERIVE
    description : text := get_description_value(SELF);
    name        : label := get_name_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
            'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
    wr2: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
            'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
END_ENTITY; -- action_request_solution

ENTITY address;
    internal_location       : OPTIONAL label;
    street_number           : OPTIONAL label;
    street                  : OPTIONAL label;
    postal_box              : OPTIONAL label;
    town                    : OPTIONAL label;
    region                  : OPTIONAL label;
    postal_code             : OPTIONAL label;
    country                 : OPTIONAL label;
    facsimile_number        : OPTIONAL label;
    telephone_number        : OPTIONAL label;
    electronic_mail_address : OPTIONAL label;
    telex_number            : OPTIONAL label;
  DERIVE
    name : label := get_name_value(SELF);
    url  : identifier := get_id_value(SELF);
  WHERE
    wr1: ((((((((((EXISTS(internal_location) OR EXISTS(street_number))
              OR EXISTS(street)) OR EXISTS(postal_box)) OR EXISTS(town))
              OR EXISTS(region)) OR EXISTS(postal_code)) OR EXISTS(country))
              OR EXISTS(facsimile_number)) OR EXISTS(telephone_number)) OR
               EXISTS(electronic_mail_address)) OR EXISTS(telex_number);
END_ENTITY; -- address

ENTITY advanced_face
  SUBTYPE OF (face_surface);
  WHERE
    wr1 : SIZEOF(['SHIP_ARRANGEMENT_SCHEMA.ELEMENTARY_SURFACE',
              'SHIP_ARRANGEMENT_SCHEMA.B_SPLINE_SURFACE',
              'SHIP_ARRANGEMENT_SCHEMA.SWEPT_SURFACE'] * TYPEOF(
              face_geometry)) = 1;
    wr2 : SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
              'SHIP_ARRANGEMENT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) )
               | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
              edge_list | (NOT ('SHIP_ARRANGEMENT_SCHEMA.EDGE_CURVE' IN
              TYPEOF(oe\oriented_edge.edge_element))) )) = 0)) )) = 0;
    wr3 : SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
              'SHIP_ARRANGEMENT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) )
               | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
              edge_list | (NOT (SIZEOF(['SHIP_ARRANGEMENT_SCHEMA.LINE',
              'SHIP_ARRANGEMENT_SCHEMA.CONIC',
              'SHIP_ARRANGEMENT_SCHEMA.POLYLINE',
              'SHIP_ARRANGEMENT_SCHEMA.SURFACE_CURVE',
              'SHIP_ARRANGEMENT_SCHEMA.B_SPLINE_CURVE'] * TYPEOF(oe.
              edge_element\edge_curve.edge_geometry)) = 1)) )) = 0)) )) =
               0;
    wr4 : SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
              'SHIP_ARRANGEMENT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) )
```



```
                | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
              edge_list | (NOT ((('SHIP_ARRANGEMENT_SCHEMA.VERTEX_POINT'
              IN TYPEOF(oe\edge.edge_start)) AND (
              'SHIP_ARRANGEMENT_SCHEMA.CARTESIAN_POINT' IN TYPEOF(oe\
              edge.edge_start\vertex_point.vertex_geometry))) AND ((
              'SHIP_ARRANGEMENT_SCHEMA.VERTEX_POINT' IN TYPEOF(oe\edge.
              edge_end)) AND ('SHIP_ARRANGEMENT_SCHEMA.CARTESIAN_POINT'
              IN TYPEOF(oe\edge.edge_end\vertex_point.vertex_geometry)))))
))
              = 0)) )) = 0;
    wr5 : SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
              'SHIP_ARRANGEMENT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) )
               | ('SHIP_ARRANGEMENT_SCHEMA.ORIENTED_PATH' IN TYPEOF(
              elp_fbnds.bound)) )) = 0;
    wr6 : (NOT ('SHIP_ARRANGEMENT_SCHEMA.SWEPT_SURFACE' IN TYPEOF(
              face_geometry))) OR (SIZEOF([
              'SHIP_ARRANGEMENT_SCHEMA.LINE',
              'SHIP_ARRANGEMENT_SCHEMA.CONIC',
              'SHIP_ARRANGEMENT_SCHEMA.POLYLINE',
              'SHIP_ARRANGEMENT_SCHEMA.B_SPLINE_CURVE'] * TYPEOF(
              face_geometry\swept_surface.swept_curve)) = 1);
    wr7 : SIZEOF(QUERY ( vlp_fbnds <* QUERY ( bnds <* bounds | (
              'SHIP_ARRANGEMENT_SCHEMA.VERTEX_LOOP' IN TYPEOF(bnds.bound))
)
               | (NOT (('SHIP_ARRANGEMENT_SCHEMA.VERTEX_POINT' IN
              TYPEOF(vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex))
              AND ('SHIP_ARRANGEMENT_SCHEMA.CARTESIAN_POINT' IN TYPEOF(
              vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex\
              vertex_point.vertex_geometry)))) )) = 0;
    wr8 : SIZEOF(QUERY ( bnd <* bounds | (NOT (SIZEOF([
              'SHIP_ARRANGEMENT_SCHEMA.EDGE_LOOP',
              'SHIP_ARRANGEMENT_SCHEMA.VERTEX_LOOP'] * TYPEOF(bnd.bound))
              = 1)) )) = 0;
    wr9 : SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
              'SHIP_ARRANGEMENT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) )
               | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
              edge_list | (('SHIP_ARRANGEMENT_SCHEMA.SURFACE_CURVE' IN
              TYPEOF(oe\oriented_edge.edge_element\edge_curve.
              edge_geometry)) AND (NOT (SIZEOF(QUERY ( sc_ag <* oe.
              edge_element\edge_curve.edge_geometry\surface_curve.
              associated_geometry | (NOT (
              'SHIP_ARRANGEMENT_SCHEMA.PCURVE' IN TYPEOF(sc_ag))) )) = 0)))
))
              = 0)) )) = 0;
    wr10: ((NOT ('SHIP_ARRANGEMENT_SCHEMA.SWEPT_SURFACE' IN TYPEOF(
              face_geometry))) OR ((NOT (
              'SHIP_ARRANGEMENT_SCHEMA.POLYLINE' IN TYPEOF(face_geometry
              \swept_surface.swept_curve))) OR (SIZEOF(face_geometry\
              swept_surface.swept_curve\polyline.points) >= 3))) AND (
              SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
              'SHIP_ARRANGEMENT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) )
               | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
              edge_list | (('SHIP_ARRANGEMENT_SCHEMA.POLYLINE' IN
              TYPEOF(oe\oriented_edge.edge_element\edge_curve.
              edge_geometry)) AND (NOT (SIZEOF(oe\oriented_edge.
              edge_element\edge_curve.edge_geometry\polyline.points) >=
3))) ))
              = 0)) )) = 0);
END_ENTITY; -- advanced_face

ENTITY application_context;
    application : label;
  DERIVE
    description : text := get_description_value(SELF);
    id          : identifier := get_id_value(SELF);
  INVERSE
```

```
    context_elements : SET [1:?] OF application_context_element FOR
                          frame_of_reference;
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
    wr2: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
END_ENTITY; -- application_context

ENTITY application_context_element
  SUPERTYPE OF (ONEOF (product_context,product_definition_context));
    name                : label;
    frame_of_reference : application_context;
END_ENTITY; -- application_context_element

ENTITY application_protocol_definition;
    status                                  : label;
    application_interpreted_model_schema_name : label;
    application_protocol_year                 : year_number;
    application                               : application_context;
END_ENTITY; -- application_protocol_definition

ENTITY applied_action_assignment
  SUBTYPE OF (action_assignment);
  items : SET [1:?] OF action_item;
END_ENTITY; -- applied_action_assignment

ENTITY applied_action_request_assignment
  SUBTYPE OF (action_request_assignment);
    items : SET [1:?] OF action_request_item;
END_ENTITY; -- applied_action_request_assignment

ENTITY applied_approval_assignment
  SUBTYPE OF (approval_assignment);
    items : SET [1:?] OF approval_item;
END_ENTITY; -- applied_approval_assignment

ENTITY applied_classification_assignment
  SUBTYPE OF (classification_assignment);
    items : SET [1:?] OF classification_item;
END_ENTITY; -- applied_classification_assignment

ENTITY applied_date_and_time_assignment
  SUBTYPE OF (date_and_time_assignment);
    items : SET [1:?] OF date_and_time_item;
END_ENTITY; -- applied_date_and_time_assignment

ENTITY applied_document_reference
  SUBTYPE OF (document_reference);
    items : SET [1:?] OF document_reference_item;
END_ENTITY; -- applied_document_reference

ENTITY applied_effectivity_assignment
SUBTYPE OF (effectivity_assignment);
items : SET [1:?] OF effectivity_item;
END_ENTITY; -- applied_effectivity_assignment

ENTITY applied_external_identification_assignment
  SUBTYPE OF (external_identification_assignment);
    items : SET [1:?] OF external_identification_item;
END_ENTITY; -- applied_external_identification_assignment

ENTITY applied_group_assignment
  SUBTYPE OF (group_assignment);
    items : SET [1:?] OF group_item;
```



```
END_ENTITY; -- applied_group_assignment

ENTITY applied_identification_assignment
  SUBTYPE OF (identification_assignment);
    items : SET [1:?] OF identification_item;
END_ENTITY; -- applied_identification_assignment

ENTITY applied_organization_assignment
  SUBTYPE OF (organization_assignment);
    items : SET [1:?] OF organization_item;
END_ENTITY; -- applied_organization_assignment

ENTITY applied_person_and_organization_assignment
  SUBTYPE OF (person_and_organization_assignment);
    items : SET [1:?] OF person_and_organization_item;
END_ENTITY; -- applied_person_and_organization_assignment

ENTITY applied_person_assignment
  SUBTYPE OF (person_assignment);
    items : SET [1:?] OF person_item;
END_ENTITY; -- applied_person_assignment

ENTITY approval;
    status : approval_status;
    level  : label;
END_ENTITY; -- approval

ENTITY approval_assignment
  ABSTRACT SUPERTYPE;
    assigned_approval : approval;
  DERIVE
    role : object_role := get_role(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'ROLE_ASSOCIATION.ITEM_WITH_ROLE')) <= 1;
END_ENTITY; -- approval_assignment

ENTITY approval_date_time;
    date_time      : date_time_select;
    dated_approval : approval;
  DERIVE
    role : object_role := get_role(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'ROLE_ASSOCIATION.ITEM_WITH_ROLE')) <= 1;
END_ENTITY; -- approval_date_time

ENTITY approval_person_organization;
    person_organization : person_organization_select;
    authorized_approval : approval;
    role                : approval_role;
END_ENTITY; -- approval_person_organization

ENTITY approval_role;
    role : label;
  DERIVE
    description : text := get_description_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY; -- approval_role

ENTITY approval_status;
    name : label;
END_ENTITY; -- approval_status
```

```
ENTITY axis1_placement
  SUBTYPE OF (placement);
    axis : OPTIONAL direction;
  DERIVE
    z : direction := NVL(normalise(axis),dummy_gri || direction([0,0,1]));
  WHERE
    wr1: SELF\geometric_representation_item.dim = 3;
END_ENTITY; -- axis1_placement

ENTITY axis2_placement_2d
  SUBTYPE OF (placement);
    ref_direction : OPTIONAL direction;
  DERIVE
    p : LIST [2:2] OF direction := build_2axes(ref_direction);
  WHERE
    wr1: SELF\geometric_representation_item.dim = 2;
END_ENTITY; -- axis2_placement_2d

ENTITY axis2_placement_3d
  SUBTYPE OF (placement);
    axis          : OPTIONAL direction;
    ref_direction : OPTIONAL direction;
  DERIVE
    p : LIST [3:3] OF direction := build_axes(axis,ref_direction);
  WHERE
    wr1: SELF\placement.location.dim = 3;
    wr2: (NOT EXISTS(axis)) OR (axis.dim = 3);
    wr3: (NOT EXISTS(ref_direction)) OR (ref_direction.dim = 3);
    wr4: ((NOT EXISTS(axis)) OR (NOT EXISTS(ref_direction))) OR (
              cross_product(axis,ref_direction).magnitude > 0);
END_ENTITY; -- axis2_placement_3d

ENTITY b_spline_curve
  SUPERTYPE OF (ONEOF (uniform_curve,b_spline_curve_with_knots,
      quasi_uniform_curve,bezier_curve) ANDOR rational_b_spline_curve)
  SUBTYPE OF (bounded_curve);
    degree              : INTEGER;
    control_points_list : LIST [2:?] OF cartesian_point;
    curve_form          : b_spline_curve_form;
    closed_curve        : LOGICAL;
    self_intersect      : LOGICAL;
  DERIVE
    upper_index_on_control_points : INTEGER := SIZEOF(
                                       control_points_list) - 1;
    control_points                : ARRAY [0:
                                       upper_index_on_control_points] OF
                                        cartesian_point := list_to_array(
                                       control_points_list,0,
                                       upper_index_on_control_points);
  WHERE
    wr1: ((('SHIP_ARRANGEMENT_SCHEMA.UNIFORM_CURVE' IN TYPEOF(SELF)) OR
              ('SHIP_ARRANGEMENT_SCHEMA.QUASI_UNIFORM_CURVE' IN TYPEOF(
             SELF))) OR ('SHIP_ARRANGEMENT_SCHEMA.BEZIER_CURVE' IN
             TYPEOF(SELF))) OR (
             'SHIP_ARRANGEMENT_SCHEMA.B_SPLINE_CURVE_WITH_KNOTS' IN
             TYPEOF(SELF));
END_ENTITY; -- b_spline_curve

ENTITY b_spline_curve_with_knots
  SUBTYPE OF (b_spline_curve);
    knot_multiplicities : LIST [2:?] OF INTEGER;
    knots               : LIST [2:?] OF parameter_value;
    knot_spec           : knot_type;
  DERIVE
```

```
    upper_index_on_knots : INTEGER := SIZEOF(knots);
  WHERE
    wr1: constraints_param_b_spline(degree,upper_index_on_knots,
             upper_index_on_control_points,knot_multiplicities,knots);
    wr2: SIZEOF(knot_multiplicities) = upper_index_on_knots;
END_ENTITY; -- b_spline_curve_with_knots

ENTITY b_spline_surface
  SUPERTYPE OF (ONEOF (b_spline_surface_with_knots,uniform_surface,
      quasi_uniform_surface,bezier_surface) ANDOR
      rational_b_spline_surface)
  SUBTYPE OF (bounded_surface);
    u_degree            : INTEGER;
    v_degree            : INTEGER;
    control_points_list : LIST [2:?] OF LIST [2:?] OF cartesian_point;
    surface_form        : b_spline_surface_form;
    u_closed            : LOGICAL;
    v_closed            : LOGICAL;
    self_intersect      : LOGICAL;
  DERIVE
    u_upper        : INTEGER := SIZEOF(control_points_list) - 1;
    v_upper        : INTEGER := SIZEOF(control_points_list[1]) - 1;
    control_points : ARRAY [0:u_upper] OF ARRAY [0:v_upper] OF
                         cartesian_point := make_array_of_array(
                        control_points_list,0,u_upper,0,v_upper);
  WHERE
    wr1: ((('SHIP_ARRANGEMENT_SCHEMA.UNIFORM_SURFACE' IN TYPEOF(SELF))
             OR ('SHIP_ARRANGEMENT_SCHEMA.QUASI_UNIFORM_SURFACE' IN
             TYPEOF(SELF))) OR ('SHIP_ARRANGEMENT_SCHEMA.BEZIER_SURFACE'
             IN TYPEOF(SELF))) OR (
             'SHIP_ARRANGEMENT_SCHEMA.B_SPLINE_SURFACE_WITH_KNOTS' IN
             TYPEOF(SELF));
END_ENTITY; -- b_spline_surface

ENTITY b_spline_surface_with_knots
  SUBTYPE OF (b_spline_surface);
    u_multiplicities : LIST [2:?] OF INTEGER;
    v_multiplicities : LIST [2:?] OF INTEGER;
    u_knots          : LIST [2:?] OF parameter_value;
    v_knots          : LIST [2:?] OF parameter_value;
    knot_spec        : knot_type;
  DERIVE
    knot_u_upper : INTEGER := SIZEOF(u_knots);
    knot_v_upper : INTEGER := SIZEOF(v_knots);
  WHERE
    wr1: constraints_param_b_spline(SELF\b_spline_surface.u_degree,
             knot_u_upper,SELF\b_spline_surface.u_upper,u_multiplicities,
             u_knots);
    wr2: constraints_param_b_spline(SELF\b_spline_surface.v_degree,
             knot_v_upper,SELF\b_spline_surface.v_upper,v_multiplicities,
             v_knots);
    wr3: SIZEOF(u_multiplicities) = knot_u_upper;
    wr4: SIZEOF(v_multiplicities) = knot_v_upper;
END_ENTITY; -- b_spline_surface_with_knots

ENTITY bezier_curve
  SUBTYPE OF (b_spline_curve);
END_ENTITY; -- bezier_curve

ENTITY bezier_surface
  SUBTYPE OF (b_spline_surface);
END_ENTITY; -- bezier_surface

ENTITY bounded_curve
  SUPERTYPE OF (ONEOF (polyline,b_spline_curve,bounded_pcurve,
      bounded_surface_curve,composite_curve))
```

```
  SUBTYPE OF (curve);
END_ENTITY; -- bounded_curve

ENTITY bounded_pcurve
  SUBTYPE OF (pcurve, bounded_curve);
  WHERE
    wr1: 'SHIP_ARRANGEMENT_SCHEMA.BOUNDED_CURVE' IN TYPEOF(SELF\pcurve.
            reference_to_curve.items[1]);
END_ENTITY; -- bounded_pcurve

ENTITY bounded_surface
  SUPERTYPE OF (b_spline_surface)
  SUBTYPE OF (surface);
END_ENTITY; -- bounded_surface

ENTITY bounded_surface_curve
  SUBTYPE OF (surface_curve, bounded_curve);
  WHERE
    wr1: 'SHIP_ARRANGEMENT_SCHEMA.BOUNDED_CURVE' IN TYPEOF(SELF\
            surface_curve.curve_3d);
END_ENTITY; -- bounded_surface_curve

ENTITY calendar_date
  SUBTYPE OF (date);
    day_component   : day_in_month_number;
    month_component : month_in_year_number;
  WHERE
    wr1: valid_calendar_date(SELF);
END_ENTITY; -- calendar_date

ENTITY cartesian_point
  SUBTYPE OF (point);
    coordinates : LIST [1:3] OF length_measure;
END_ENTITY; -- cartesian_point

ENTITY cartesian_transformation_operator
  SUPERTYPE OF (cartesian_transformation_operator_3d)
  SUBTYPE OF (geometric_representation_item,
      functionally_defined_transformation);
    axis1        : OPTIONAL direction;
    axis2        : OPTIONAL direction;
    local_origin : cartesian_point;
    scale        : OPTIONAL REAL;
  DERIVE
    scl : REAL := NVL(scale,1);
  WHERE
    wr1: scl > 0;
END_ENTITY; -- cartesian_transformation_operator

ENTITY cartesian_transformation_operator_3d
  SUBTYPE OF (cartesian_transformation_operator);
    axis3 : OPTIONAL direction;
  DERIVE
    u : LIST [3:3] OF direction := base_axis(3,SELF\
           cartesian_transformation_operator.axis1,SELF\
           cartesian_transformation_operator.axis2,axis3);
  WHERE
    wr1: SELF\geometric_representation_item.dim = 3;
END_ENTITY; -- cartesian_transformation_operator_3d

ENTITY characterized_object;
    name        : label;
    description : OPTIONAL text;
END_ENTITY; -- characterized_object
```

```
ENTITY circle
  SUBTYPE OF (conic);
    radius : positive_length_measure;
END_ENTITY; -- circle

ENTITY class
  SUBTYPE OF (group);
  WHERE
    wr1: SIZEOF(QUERY ( oa <* USEDIN(SELF,
              'SHIP_ARRANGEMENT_SCHEMA.GROUP_ASSIGNMENT.ASSIGNED_GROUP')
               | (NOT ('SHIP_ARRANGEMENT_SCHEMA.APPLIED_GROUP_ASSIGNMENT'
              IN TYPEOF(oa))) )) = 0;
END_ENTITY; -- class

ENTITY classification_assignment
  ABSTRACT SUPERTYPE;
    assigned_class : group;
    role           : classification_role;
END_ENTITY; -- classification_assignment

ENTITY classification_role;
    name        : label;
    description : OPTIONAL text;
END_ENTITY; -- classification_role

ENTITY closed_shell
  SUBTYPE OF (connected_face_set);
END_ENTITY; -- closed_shell

ENTITY composite_curve
  SUBTYPE OF (bounded_curve);
    segments       : LIST [1:?] OF composite_curve_segment;
    self_intersect : LOGICAL;
  DERIVE
    n_segments   : INTEGER := SIZEOF(segments);
    closed_curve : LOGICAL := segments[n_segments].transition <>
                        discontinuous;
  WHERE
    wr1: ((NOT closed_curve) AND (SIZEOF(QUERY ( temp <* segments | (
              temp.transition = discontinuous) )) = 1)) OR (closed_curve
              AND (SIZEOF(QUERY ( temp <* segments | (temp.transition =
              discontinuous) )) = 0));
END_ENTITY; -- composite_curve

ENTITY composite_curve_on_surface
  SUBTYPE OF (composite_curve);
  DERIVE
    basis_surface : SET [0:2] OF surface := get_basis_surface(SELF);
  WHERE
    wr1: SIZEOF(basis_surface) > 0;
    wr2: constraints_composite_curve_on_surface(SELF);
END_ENTITY; -- composite_curve_on_surface

ENTITY composite_curve_segment
  SUBTYPE OF (founded_item);
    transition   : transition_code;
    same_sense   : BOOLEAN;
    parent_curve : curve;
  INVERSE
    using_curves : BAG [1:?] OF composite_curve FOR segments;
  WHERE
    wr1: 'SHIP_ARRANGEMENT_SCHEMA.BOUNDED_CURVE' IN TYPEOF(parent_curve);
END_ENTITY; -- composite_curve_segment

ENTITY compound_representation_item
```

```
  SUBTYPE OF (representation_item);
    item_element : compound_item_definition;
END_ENTITY; -- compound_representation_item

ENTITY conic
  SUPERTYPE OF (ONEOF (circle,ellipse,hyperbola,parabola))
  SUBTYPE OF (curve);
    position : axis2_placement;
END_ENTITY; -- conic

ENTITY conical_surface
  SUBTYPE OF (elementary_surface);
    radius     : length_measure;
    semi_angle : plane_angle_measure;
  WHERE
    wr1: radius >= 0;
END_ENTITY; -- conical_surface

ENTITY connected_face_set
  SUPERTYPE OF (ONEOF (closed_shell,open_shell))
  SUBTYPE OF (topological_representation_item);
    cfs_faces : SET [1:?] OF face;
END_ENTITY; -- connected_face_set

ENTITY context_dependent_unit
  SUBTYPE OF (named_unit);
    name : label;
END_ENTITY; -- context_dependent_unit

ENTITY coordinated_universal_time_offset;
    hour_offset   : INTEGER;
    minute_offset : OPTIONAL INTEGER;
    sense         : ahead_or_behind;
  DERIVE
    actual_minute_offset : INTEGER := NVL(minute_offset,0);
  WHERE
    wr1: (0 <= hour_offset) AND (hour_offset < 24);
    wr2: (0 <= actual_minute_offset) AND (actual_minute_offset <= 59);
    wr3: NOT (((hour_offset <> 0) OR (actual_minute_offset <> 0)) AND (
           sense = exact));
END_ENTITY; -- coordinated_universal_time_offset

ENTITY curve
  SUPERTYPE OF (ONEOF (line,conic,pcurve,surface_curve,offset_curve_3d,
      curve_replica))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- curve

ENTITY curve_replica
  SUBTYPE OF (curve);
    parent_curve   : curve;
    transformation : cartesian_transformation_operator;
  WHERE
    wr1: transformation.dim = parent_curve.dim;
    wr2: acyclic_curve_replica(SELF,parent_curve);
END_ENTITY; -- curve_replica

ENTITY cylindrical_surface
  SUBTYPE OF (elementary_surface);
    radius : positive_length_measure;
END_ENTITY; -- cylindrical_surface

ENTITY date
  SUPERTYPE OF (ONEOF (calendar_date,ordinal_date,
```



```
      week_of_year_and_day_date));
    year_component : year_number;
END_ENTITY; -- date

ENTITY date_and_time;
    date_component : date;
    time_component : local_time;
END_ENTITY; -- date_and_time

ENTITY date_and_time_assignment
  ABSTRACT SUPERTYPE;
    assigned_date_and_time : date_and_time;
    role                   : date_time_role;
END_ENTITY; -- date_and_time_assignment

ENTITY date_time_role;
    name : label;
  DERIVE
    description : text := get_description_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY; -- date_time_role

ENTITY definitional_representation
  SUBTYPE OF (representation);
  WHERE
    wr1: 'SHIP_ARRANGEMENT_SCHEMA.PARAMETRIC_REPRESENTATION_CONTEXT' IN
              TYPEOF(SELF\representation.context_of_items);
END_ENTITY; -- definitional_representation

ENTITY degenerate_pcurve
  SUBTYPE OF (point);
    basis_surface      : surface;
    reference_to_curve : definitional_representation;
  WHERE
    wr1: SIZEOF(reference_to_curve\representation.items) = 1;
    wr2: 'SHIP_ARRANGEMENT_SCHEMA.CURVE' IN TYPEOF(reference_to_curve\
              representation.items[1]);
    wr3: reference_to_curve\representation.items[1]\
              geometric_representation_item.dim = 2;
END_ENTITY; -- degenerate_pcurve

ENTITY degenerate_toroidal_surface
  SUBTYPE OF (toroidal_surface);
    select_outer : BOOLEAN;
  WHERE
    wr1: major_radius < minor_radius;
END_ENTITY; -- degenerate_toroidal_surface

ENTITY derived_unit;
    elements : SET [1:?] OF derived_unit_element;
  DERIVE
    name : label := get_name_value(SELF);
  WHERE
    wr1: (SIZEOF(elements) > 1) OR ((SIZEOF(elements) = 1) AND (elements
              [1].exponent <> 1));
    wr2: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
END_ENTITY; -- derived_unit

ENTITY derived_unit_element;
    unit     : named_unit;
    exponent : REAL;
END_ENTITY; -- derived_unit_element
```

```
ENTITY description_attribute;
    attribute_value : text;
    described_item  : description_attribute_select;
END_ENTITY; -- description_attribute

ENTITY descriptive_representation_item
  SUBTYPE OF (representation_item);
    description : text;
END_ENTITY; -- descriptive_representation_item

ENTITY dimensional_exponents;
    length_exponent                   : REAL;
    mass_exponent                     : REAL;
    time_exponent                     : REAL;
    electric_current_exponent         : REAL;
    thermodynamic_temperature_exponent : REAL;
    amount_of_substance_exponent      : REAL;
    luminous_intensity_exponent       : REAL;
END_ENTITY; -- dimensional_exponents

ENTITY direction
  SUBTYPE OF (geometric_representation_item);
    direction_ratios : LIST [2:3] OF REAL;
  WHERE
    wr1: SIZEOF(QUERY ( tmp <* direction_ratios | (tmp <> 0) )) > 0;
END_ENTITY; -- direction

ENTITY document;
    id          : identifier;
    name        : label;
    description : OPTIONAL text;
    kind        : document_type;
  INVERSE
    representation_types : SET [0:?] OF document_representation_type FOR
                              represented_document;
END_ENTITY; -- document

ENTITY document_reference
  ABSTRACT SUPERTYPE;
    assigned_document : document;
    source            : label;
  DERIVE
    role : object_role := get_role(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'ROLE_ASSOCIATION.ITEM_WITH_ROLE')) <= 1;
END_ENTITY; -- document_reference

ENTITY document_representation_type;
    name                : label;
    represented_document : document;
END_ENTITY; -- document_representation_type

ENTITY document_type;
    product_data_type : label;
END_ENTITY; -- document_type

ENTITY document_usage_constraint;
    source                : document;
    subject_element       : label;
    subject_element_value : text;
END_ENTITY; -- document_usage_constraint

ENTITY edge
```

```
  SUPERTYPE OF (ONEOF (edge_curve,oriented_edge))
  SUBTYPE OF (topological_representation_item);
    edge_start : vertex;
    edge_end   : vertex;
END_ENTITY; -- edge

ENTITY edge_curve
  SUBTYPE OF (edge, geometric_representation_item);
    edge_geometry : curve;
    same_sense    : BOOLEAN;
END_ENTITY; -- edge_curve

ENTITY edge_loop
  SUBTYPE OF (loop, path);
  DERIVE
    ne : INTEGER := SIZEOF(SELF\path.edge_list);
  WHERE
    wr1: SELF\path.edge_list[1].edge_start :=: SELF\path.edge_list[ne].
              edge_end;
END_ENTITY; -- edge_loop

ENTITY effectivity
  SUPERTYPE OF (serial_numbered_effectivity);
    id : identifier;
  DERIVE
    name : label := get_name_value(SELF);
    description : text := get_description_value(SELF);
  WHERE
   wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
     'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
   wr2: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
     'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY; -- effectivity

ENTITY effectivity_assignment
  ABSTRACT SUPERTYPE;
    assigned_effectivity : effectivity;
  DERIVE
    role : object_role := get_role(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
     'ROLE_ASSOCIATION.ITEM_WITH_ROLE')) <= 1;
END_ENTITY; -- effectivity_assignment

ENTITY elementary_surface
  SUPERTYPE OF (ONEOF (plane,cylindrical_surface,conical_surface,
      spherical_surface,toroidal_surface))
  SUBTYPE OF (surface);
    position : axis2_placement_3d;
END_ENTITY; -- elementary_surface

ENTITY ellipse
  SUBTYPE OF (conic);
    semi_axis_1 : positive_length_measure;
    semi_axis_2 : positive_length_measure;
END_ENTITY; -- ellipse

ENTITY evaluated_degenerate_pcurve
  SUBTYPE OF (degenerate_pcurve);
    equivalent_point : cartesian_point;
END_ENTITY; -- evaluated_degenerate_pcurve

ENTITY executed_action
  SUBTYPE OF (action);
END_ENTITY; -- executed_action

ENTITY external_identification_assignment
```

```
  ABSTRACT SUPERTYPE
  SUBTYPE OF (identification_assignment);
    source : external_source;
END_ENTITY; -- external_identification_assignment

ENTITY external_source;
    source_id : source_item;
  DERIVE
    description : text := get_description_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY; -- external_source

ENTITY external_source_relationship;
    name            : label;
    description     : OPTIONAL text;
    relating_source : external_source;
    related_source  : external_source;
END_ENTITY; -- external_source_relationship

ENTITY face
  SUPERTYPE OF (ONEOF (face_surface,oriented_face))
  SUBTYPE OF (topological_representation_item);
    bounds : SET [1:?] OF face_bound;
  WHERE
    wr1: NOT mixed_loop_type_set(list_to_set(list_face_loops(SELF)));
    wr2: SIZEOF(QUERY ( temp <* bounds | (
              'SHIP_ARRANGEMENT_SCHEMA.FACE_OUTER_BOUND' IN TYPEOF(temp)) ))
              <= 1;
END_ENTITY; -- face

ENTITY face_based_surface_model
  SUBTYPE OF (geometric_representation_item);
    fbsm_faces : SET [1:?] OF connected_face_set;
END_ENTITY; -- face_based_surface_model

ENTITY face_bound
  SUBTYPE OF (topological_representation_item);
    bound       : loop;
    orientation : BOOLEAN;
END_ENTITY; -- face_bound

ENTITY face_outer_bound
  SUBTYPE OF (face_bound);
END_ENTITY; -- face_outer_bound

ENTITY face_surface
  SUBTYPE OF (face, geometric_representation_item);
    face_geometry : surface;
    same_sense    : BOOLEAN;
  WHERE
    wr1: NOT ('SHIP_ARRANGEMENT_SCHEMA.ORIENTED_SURFACE' IN TYPEOF(
              face_geometry));
END_ENTITY; -- face_surface

ENTITY founded_item;
END_ENTITY; -- founded_item

ENTITY functionally_defined_transformation;
    name        : label;
    description : OPTIONAL text;
END_ENTITY; -- functionally_defined_transformation
```

```
ENTITY geometric_representation_context
  SUBTYPE OF (representation_context);
    coordinate_space_dimension : dimension_count;
END_ENTITY; -- geometric_representation_context

ENTITY geometric_representation_item
  SUPERTYPE OF (ONEOF (point,direction,vector,placement,
      cartesian_transformation_operator,curve,surface,edge_curve,
      face_surface,poly_loop,vertex_point,face_based_surface_model))
  SUBTYPE OF (representation_item);
  DERIVE
    dim : dimension_count := dimension_of(SELF);
  WHERE
    wr1: SIZEOF(QUERY ( using_rep <* using_representations(SELF) | (NOT
              ('SHIP_ARRANGEMENT_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT'
              IN TYPEOF(using_rep.context_of_items))) )) = 0;
END_ENTITY; -- geometric_representation_item

ENTITY  global_uncertainty_assigned_context
  SUBTYPE OF (representation_context);
      uncertainty : SET [1:?] OF uncertainty_measure_with_unit;
END_ENTITY; -- global_uncertainty_assigned_context

ENTITY global_unit_assigned_context
  SUBTYPE OF (representation_context);
    units : SET [1:?] OF unit;
END_ENTITY; -- global_unit_assigned_context

ENTITY group;
    name        : label;
    description : OPTIONAL text;
  DERIVE
    id : identifier := get_id_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
END_ENTITY; -- group

ENTITY group_assignment
  ABSTRACT SUPERTYPE;
    assigned_group : group;
  DERIVE
    role : object_role := get_role(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'ROLE_ASSOCIATION.ITEM_WITH_ROLE')) <= 1;
END_ENTITY; -- group_assignment

ENTITY group_relationship;
    name           : label;
    description    : OPTIONAL text;
    relating_group : group;
    related_group  : group;
END_ENTITY; -- group_relationship

ENTITY hyperbola
  SUBTYPE OF (conic);
    semi_axis      : positive_length_measure;
    semi_imag_axis : positive_length_measure;
END_ENTITY; -- hyperbola

ENTITY id_attribute;
    attribute_value : identifier;
    identified_item : id_attribute_select;
END_ENTITY; -- id_attribute
```

```
ENTITY identification_assignment
  ABSTRACT SUPERTYPE;
    assigned_id : identifier;
    role        : identification_role;
END_ENTITY; -- identification_assignment

ENTITY identification_assignment_relationship;
    name                              : label;
    description                       : OPTIONAL text;
    relating_identification_assignment : identification_assignment;
    related_identification_assignment  : identification_assignment;
END_ENTITY; -- identification_assignment_relationship

ENTITY identification_role;
    name        : label;
    description : OPTIONAL text;
END_ENTITY; -- identification_role

ENTITY intersection_curve
  SUBTYPE OF (surface_curve);
  WHERE
    wr1: SIZEOF(SELF\surface_curve.associated_geometry) = 2;
    wr2: associated_surface(SELF\surface_curve.associated_geometry[1])
             <> associated_surface(SELF\surface_curve.associated_geometry
             [2]);
END_ENTITY; -- intersection_curve

ENTITY item_defined_transformation;
    name             : label;
    description      : OPTIONAL text;
    transform_item_1 : representation_item;
    transform_item_2 : representation_item;
END_ENTITY; -- item_defined_transformation

ENTITY length_unit
  SUBTYPE OF (named_unit);
  WHERE
    wr1: ((((((SELF\named_unit.dimensions.length_exponent = 1) AND (SELF
             \named_unit.dimensions.mass_exponent = 0)) AND (SELF\
             named_unit.dimensions.time_exponent = 0)) AND (SELF\
             named_unit.dimensions.electric_current_exponent = 0)) AND (
             SELF\named_unit.dimensions.
             thermodynamic_temperature_exponent = 0)) AND (SELF\
             named_unit.dimensions.amount_of_substance_exponent = 0)) AND
              (SELF\named_unit.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- length_unit

ENTITY line
  SUBTYPE OF (curve);
    pnt : cartesian_point;
    dir : vector;
  WHERE
    wr1: dir.dim = pnt.dim;
END_ENTITY; -- line

ENTITY local_time;
    hour_component   : hour_in_day;
    minute_component : OPTIONAL minute_in_hour;
    second_component : OPTIONAL second_in_minute;
    zone             : coordinated_universal_time_offset;
  WHERE
    wr1: valid_time(SELF);
END_ENTITY; -- local_time
```



```
ENTITY loop
  SUPERTYPE OF (ONEOF (vertex_loop,edge_loop,poly_loop))
  SUBTYPE OF (topological_representation_item);
END_ENTITY; -- loop

ENTITY luminous_intensity_unit
  SUBTYPE OF (named_unit);
  WHERE
    wr1: ((((((SELF\named_unit.dimensions.length_exponent = 0) AND (SELF
              \named_unit.dimensions.mass_exponent = 0)) AND (SELF\
              named_unit.dimensions.time_exponent = 0)) AND (SELF\
              named_unit.dimensions.electric_current_exponent = 0)) AND (
              SELF\named_unit.dimensions.
              thermodynamic_temperature_exponent = 0)) AND (SELF\
              named_unit.dimensions.amount_of_substance_exponent = 0)) AND
               (SELF\named_unit.dimensions.luminous_intensity_exponent = 1);
END_ENTITY; -- luminous_intensity_unit

ENTITY mapped_item
  SUBTYPE OF (representation_item);
    mapping_source : representation_map;
    mapping_target : representation_item;
  WHERE
    wr1: acyclic_mapped_representation(using_representations(SELF),[SELF]);
END_ENTITY; -- mapped_item

ENTITY mass_unit
  SUBTYPE OF (named_unit);
  WHERE
    wr1: ((((((SELF\named_unit.dimensions.length_exponent = 0) AND (SELF
              \named_unit.dimensions.mass_exponent = 1)) AND (SELF\
              named_unit.dimensions.time_exponent = 0)) AND (SELF\
              named_unit.dimensions.electric_current_exponent = 0)) AND (
              SELF\named_unit.dimensions.
              thermodynamic_temperature_exponent = 0)) AND (SELF\
              named_unit.dimensions.amount_of_substance_exponent = 0)) AND
               (SELF\named_unit.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- mass_unit

ENTITY measure_with_unit;
    value_component : measure_value;
    unit_component  : unit;
  WHERE
    wr1: valid_units(SELF);
END_ENTITY; -- measure_with_unit

ENTITY name_attribute;
    attribute_value : label;
    named_item      : name_attribute_select;
END_ENTITY; -- name_attribute

ENTITY named_unit
  SUPERTYPE OF (ONEOF (si_unit,context_dependent_unit) ANDOR ONEOF (
      length_unit,mass_unit,time_unit,thermodynamic_temperature_unit,
      luminous_intensity_unit,plane_angle_unit, ratio_unit));
    dimensions : dimensional_exponents;
END_ENTITY; -- named_unit

ENTITY non_manifold_surface_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    wr1 : SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
              'SHIP_ARRANGEMENT_SCHEMA.FACE_BASED_SURFACE_MODEL',
              'SHIP_ARRANGEMENT_SCHEMA.MAPPED_ITEM',
              'SHIP_ARRANGEMENT_SCHEMA.AXIS2_PLACEMENT_3D'] * TYPEOF(it))
             = 1)) )) = 0;
```

```
    wr2 : SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
              'SHIP_ARRANGEMENT_SCHEMA.FACE_BASED_SURFACE_MODEL',
              'SHIP_ARRANGEMENT_SCHEMA.MAPPED_ITEM'] * TYPEOF(it)) = 1) ))
              > 0;
    wr3 : SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
              'SHIP_ARRANGEMENT_SCHEMA.MAPPED_ITEM' IN TYPEOF(it)) ) | (
              NOT ((('SHIP_ARRANGEMENT_SCHEMA.' +
              'NON_MANIFOLD_SURFACE_SHAPE_REPRESENTATION') IN TYPEOF(mi\
              mapped_item.mapping_source.mapped_representation)) AND (
              SIZEOF(QUERY ( mr_it <* mi\mapped_item.mapping_source.
              mapped_representation.items | (
              'SHIP_ARRANGEMENT_SCHEMA.FACE_BASED_SURFACE_MODEL' IN
              TYPEOF(mr_it)) )) > 0))) )) = 0;
    wr4 : SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
              'SHIP_ARRANGEMENT_SCHEMA.FACE_BASED_SURFACE_MODEL' IN
              TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* fbsm\
              face_based_surface_model.fbsm_faces | (NOT (SIZEOF(
              QUERY ( fa <* cfs.cfs_faces | (NOT (SIZEOF([
              'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE',
              'SHIP_ARRANGEMENT_SCHEMA.ORIENTED_FACE'] * TYPEOF(fa)) = 1))
))
              = 0)) )) = 0)) )) = 0;
    wr5 : SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
              'SHIP_ARRANGEMENT_SCHEMA.FACE_BASED_SURFACE_MODEL' IN
              TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* fbsm\
              face_based_surface_model.fbsm_faces | (NOT (SIZEOF(
              QUERY ( f_sf <* QUERY ( fa <* cfs.cfs_faces | ((
              'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa)) OR (
              'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa\
              oriented_face.face_element))) ) | (NOT ((
              'SHIP_ARRANGEMENT_SCHEMA.ADVANCED_FACE' IN TYPEOF(f_sf))
              OR (SIZEOF(['SHIP_ARRANGEMENT_SCHEMA.B_SPLINE_SURFACE',
              'SHIP_ARRANGEMENT_SCHEMA.ELEMENTARY_SURFACE',
              'SHIP_ARRANGEMENT_SCHEMA.OFFSET_SURFACE',
              'SHIP_ARRANGEMENT_SCHEMA.SURFACE_REPLICA',
              'SHIP_ARRANGEMENT_SCHEMA.SWEPT_SURFACE'] * TYPEOF(f_sf\
              face_surface.face_geometry)) = 1))) )) = 0)) )) = 0)) )) =
              0;
    wr6 : SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
              'SHIP_ARRANGEMENT_SCHEMA.FACE_BASED_SURFACE_MODEL' IN
              TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* fbsm\
              face_based_surface_model.fbsm_faces | (NOT (SIZEOF(
              QUERY ( f_sf <* QUERY ( fa <* cfs.cfs_faces | ((
              'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa)) OR (
              'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa\
              oriented_face.face_element))) ) | (NOT ((
              'SHIP_ARRANGEMENT_SCHEMA.ADVANCED_FACE' IN TYPEOF(f_sf))
              OR nmsf_surface_check(f_sf\face_surface.face_geometry))) ))
              = 0)) )) = 0)) )) = 0;
    wr7 : SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
              'SHIP_ARRANGEMENT_SCHEMA.FACE_BASED_SURFACE_MODEL' IN
              TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* fbsm\
              face_based_surface_model.fbsm_faces | (NOT (SIZEOF(
              QUERY ( f_sf <* QUERY ( fa <* cfs.cfs_faces | ((
              'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa)) OR (
              'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa\
              oriented_face.face_element))) ) | (NOT ((
              'SHIP_ARRANGEMENT_SCHEMA.ADVANCED_FACE' IN TYPEOF(f_sf))
              OR (SIZEOF(QUERY ( bnds <* f_sf.bounds | (NOT (SIZEOF([
              'SHIP_ARRANGEMENT_SCHEMA.EDGE_LOOP',
              'SHIP_ARRANGEMENT_SCHEMA.VERTEX_LOOP'] * TYPEOF(bnds.bound))
              = 1)) )) = 0))) )) = 0)) )) = 0)) )) = 0;
    wr8 : SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
              'SHIP_ARRANGEMENT_SCHEMA.FACE_BASED_SURFACE_MODEL' IN
```

```
            TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* fbsm\
            face_based_surface_model.fbsm_faces | (NOT (SIZEOF(
            QUERY ( f_sf <* QUERY ( fa <* cfs.cfs_faces | ((
            'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa)) OR (
            'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa\
            oriented_face.face_element))) ) | (NOT ((
            'SHIP_ARRANGEMENT_SCHEMA.ADVANCED_FACE' IN TYPEOF(f_sf))
            OR (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* f_sf.bounds
             | ('SHIP_ARRANGEMENT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.
            bound)) ) | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds\path.
            edge_list | (NOT ('SHIP_ARRANGEMENT_SCHEMA.EDGE_CURVE' IN
            TYPEOF(oe.edge_element))) )) = 0)) )) = 0))) )) = 0)) )) =
            0)) )) = 0;
    wr9 : SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
            'SHIP_ARRANGEMENT_SCHEMA.FACE_BASED_SURFACE_MODEL' IN
            TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* fbsm\
            face_based_surface_model.fbsm_faces | (NOT (SIZEOF(
            QUERY ( f_sf <* QUERY ( fa <* cfs.cfs_faces | ((
            'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa)) OR (
            'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa\
            oriented_face.face_element))) ) | (NOT ((
            'SHIP_ARRANGEMENT_SCHEMA.ADVANCED_FACE' IN TYPEOF(f_sf))
            OR (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* f_sf.bounds
             | ('SHIP_ARRANGEMENT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.
            bound)) ) | (NOT (SIZEOF(QUERY ( oe_cv <* QUERY ( oe <*
            elp_fbnds\path.edge_list | (
            'SHIP_ARRANGEMENT_SCHEMA.EDGE_CURVE' IN TYPEOF(oe.
            edge_element)) ) | (NOT (SIZEOF([
            'SHIP_ARRANGEMENT_SCHEMA.B_SPLINE_CURVE',
            'SHIP_ARRANGEMENT_SCHEMA.CONIC',
            'SHIP_ARRANGEMENT_SCHEMA.CURVE_REPLICA',
            'SHIP_ARRANGEMENT_SCHEMA.LINE',
            'SHIP_ARRANGEMENT_SCHEMA.OFFSET_CURVE_3D',
            'SHIP_ARRANGEMENT_SCHEMA.PCURVE',
            'SHIP_ARRANGEMENT_SCHEMA.POLYLINE',
            'SHIP_ARRANGEMENT_SCHEMA.SURFACE_CURVE'] * TYPEOF(oe_cv.
            edge_element\edge_curve.edge_geometry)) = 1)) )) = 0)) )) =
             0))) )) = 0)) )) = 0)) )) = 0;
    wr10: SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
            'SHIP_ARRANGEMENT_SCHEMA.FACE_BASED_SURFACE_MODEL' IN
            TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* fbsm\
            face_based_surface_model.fbsm_faces | (NOT (SIZEOF(
            QUERY ( f_sf <* QUERY ( fa <* cfs.cfs_faces | ((
            'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa)) OR (
            'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa\
            oriented_face.face_element))) ) | (NOT ((
            'SHIP_ARRANGEMENT_SCHEMA.ADVANCED_FACE' IN TYPEOF(f_sf))
            OR (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* f_sf.bounds
             | ('SHIP_ARRANGEMENT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.
            bound)) ) | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds\path.
            edge_list | (NOT nmsf_curve_check(oe.edge_element\
            edge_curve.edge_geometry)) )) = 0)) )) = 0))) )) = 0)) )) =
             0)) )) = 0;
    wr11: SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
            'SHIP_ARRANGEMENT_SCHEMA.FACE_BASED_SURFACE_MODEL' IN
            TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* fbsm\
            face_based_surface_model.fbsm_faces | (NOT (SIZEOF(
            QUERY ( f_sf <* QUERY ( fa <* cfs.cfs_faces | ((
            'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa)) OR (
            'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa\
            oriented_face.face_element))) ) | (NOT ((
            'SHIP_ARRANGEMENT_SCHEMA.ADVANCED_FACE' IN TYPEOF(f_sf))
            OR (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* f_sf.bounds
             | ('SHIP_ARRANGEMENT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.
            bound)) ) | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds\path.
            edge_list | (NOT (('SHIP_ARRANGEMENT_SCHEMA.VERTEX_POINT'
```

```
            IN TYPEOF(oe.edge_element.edge_start)) AND (
            'SHIP_ARRANGEMENT_SCHEMA.VERTEX_POINT' IN TYPEOF(oe.
            edge_element.edge_end)))) )) = 0)) )) = 0))) )) = 0)) )) =
            0)) )) = 0;
    wr12: SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
            'SHIP_ARRANGEMENT_SCHEMA.FACE_BASED_SURFACE_MODEL' IN
            TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* fbsm\
            face_based_surface_model.fbsm_faces | (NOT (SIZEOF(
            QUERY ( f_sf <* QUERY ( fa <* cfs.cfs_faces | ((
            'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa)) OR (
            'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa\
            oriented_face.face_element))) ) | (NOT ((
            'SHIP_ARRANGEMENT_SCHEMA.ADVANCED_FACE' IN TYPEOF(f_sf))
            OR (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* f_sf.bounds
             | ('SHIP_ARRANGEMENT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.
            bound)) ) | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds\path.
            edge_list | (NOT ((SIZEOF([
            'SHIP_ARRANGEMENT_SCHEMA.CARTESIAN_POINT',
            'SHIP_ARRANGEMENT_SCHEMA.DEGENERATE_PCURVE',
            'SHIP_ARRANGEMENT_SCHEMA.POINT_ON_CURVE',
            'SHIP_ARRANGEMENT_SCHEMA.POINT_ON_SURFACE'] * TYPEOF(oe.
            edge_element.edge_start\vertex_point.vertex_geometry)) = 1)
            AND (SIZEOF(['SHIP_ARRANGEMENT_SCHEMA.CARTESIAN_POINT',
            'SHIP_ARRANGEMENT_SCHEMA.DEGENERATE_PCURVE',
            'SHIP_ARRANGEMENT_SCHEMA.POINT_ON_CURVE',
            'SHIP_ARRANGEMENT_SCHEMA.POINT_ON_SURFACE'] * TYPEOF(oe.
            edge_element.edge_end\vertex_point.vertex_geometry)) = 1)))
))
            = 0)) )) = 0))) )) = 0)) )) = 0)) )) = 0;
    wr13: SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
            'SHIP_ARRANGEMENT_SCHEMA.FACE_BASED_SURFACE_MODEL' IN
            TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* fbsm\
            face_based_surface_model.fbsm_faces | (NOT (SIZEOF(
            QUERY ( f_sf <* QUERY ( fa <* cfs.cfs_faces | ((
            'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa)) OR (
            'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa\
            oriented_face.face_element))) ) | (NOT ((
            'SHIP_ARRANGEMENT_SCHEMA.ADVANCED_FACE' IN TYPEOF(f_sf))
            OR (SIZEOF(QUERY ( vlp_fbnds <* QUERY ( bnds <* f_sf.bounds
             | ('SHIP_ARRANGEMENT_SCHEMA.VERTEX_LOOP' IN TYPEOF(bnds.
            bound)) ) | (NOT ('SHIP_ARRANGEMENT_SCHEMA.VERTEX_POINT'
            IN TYPEOF(vlp_fbnds\vertex_loop.loop_vertex))) )) = 0))) ))
            = 0)) )) = 0)) )) = 0;
    wr14: SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
            'SHIP_ARRANGEMENT_SCHEMA.FACE_BASED_SURFACE_MODEL' IN
            TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* fbsm\
            face_based_surface_model.fbsm_faces | (NOT (SIZEOF(
            QUERY ( f_sf <* QUERY ( fa <* cfs.cfs_faces | ((
            'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa)) OR (
            'SHIP_ARRANGEMENT_SCHEMA.FACE_SURFACE' IN TYPEOF(fa\
            oriented_face.face_element))) ) | (NOT ((
            'SHIP_ARRANGEMENT_SCHEMA.ADVANCED_FACE' IN TYPEOF(f_sf))
            OR (SIZEOF(QUERY ( vlp_fbnds <* QUERY ( bnds <* f_sf.bounds
             | ('SHIP_ARRANGEMENT_SCHEMA.VERTEX_LOOP' IN TYPEOF(bnds.
            bound)) ) | (NOT (SIZEOF([
            'SHIP_ARRANGEMENT_SCHEMA.CARTESIAN_POINT',
            'SHIP_ARRANGEMENT_SCHEMA.DEGENERATE_PCURVE',
            'SHIP_ARRANGEMENT_SCHEMA.POINT_ON_CURVE',
            'SHIP_ARRANGEMENT_SCHEMA.POINT_ON_SURFACE'] * TYPEOF(
            vlp_fbnds\vertex_loop.loop_vertex\vertex_point.
            vertex_geometry)) = 1)) )) = 0))) )) = 0)) )) = 0)) )) = 0;
END_ENTITY; -- non_manifold_surface_shape_representation

ENTITY object_role;
```

```
    name        : label;
    description : OPTIONAL text;
END_ENTITY; -- object_role

ENTITY offset_curve_3d
  SUBTYPE OF (curve);
    basis_curve    : curve;
    distance       : length_measure;
    self_intersect : LOGICAL;
    ref_direction  : direction;
  WHERE
    wr1: (basis_curve.dim = 3) AND (ref_direction.dim = 3);
END_ENTITY; -- offset_curve_3d

ENTITY offset_surface
  SUBTYPE OF (surface);
    basis_surface  : surface;
    distance       : length_measure;
    self_intersect : LOGICAL;
END_ENTITY; -- offset_surface

ENTITY open_shell
  SUBTYPE OF (connected_face_set);
END_ENTITY; -- open_shell

ENTITY ordinal_date
  SUBTYPE OF (date);
    day_component : day_in_year_number;
  WHERE
    wr1: ((NOT leap_year(SELF.year_component)) AND ((1 <= day_component)
              AND (day_component <= 365))) OR (leap_year(SELF.
              year_component) AND ((1 <= day_component) AND (day_component
              <= 366)));
END_ENTITY; -- ordinal_date

ENTITY organization;
    id          : OPTIONAL identifier;
    name        : label;
    description : OPTIONAL text;
END_ENTITY; -- organization

ENTITY organization_assignment
  ABSTRACT SUPERTYPE;
    assigned_organization : organization;
    role                  : organization_role;
END_ENTITY; -- organization_assignment

ENTITY organization_role;
    name : label;
  DERIVE
    description : text := get_description_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY; -- organization_role

ENTITY organizational_address
  SUBTYPE OF (address);
    organizations : SET [1:?] OF organization;
    description   : OPTIONAL text;
END_ENTITY; -- organizational_address

ENTITY organizational_project;
    name                     : label;
    description              : OPTIONAL text;
    responsible_organizations : SET [1:?] OF organization;
```

```
  DERIVE
    id : identifier := get_id_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
END_ENTITY; -- organizational_project

ENTITY oriented_closed_shell
  SUBTYPE OF (closed_shell);
    closed_shell_element : closed_shell;
    orientation          : BOOLEAN;
  DERIVE
    SELF\connected_face_set.cfs_faces : SET [1:?] OF face :=
                                          conditional_reverse(SELF.
                                          orientation,SELF.
                                          closed_shell_element.cfs_faces);
  WHERE
    wr1: NOT ('SHIP_ARRANGEMENT_SCHEMA.ORIENTED_CLOSED_SHELL' IN
              TYPEOF(SELF.closed_shell_element));
END_ENTITY; -- oriented_closed_shell

ENTITY oriented_edge
  SUBTYPE OF (edge);
    edge_element : edge;
    orientation  : BOOLEAN;
  DERIVE
    SELF\edge.edge_start : vertex := boolean_choose(SELF.orientation,
                               SELF.edge_element.edge_start,SELF.
                               edge_element.edge_end);
    SELF\edge.edge_end   : vertex := boolean_choose(SELF.orientation,
                               SELF.edge_element.edge_end,SELF.
                               edge_element.edge_start);
  WHERE
    wr1: NOT ('SHIP_ARRANGEMENT_SCHEMA.ORIENTED_EDGE' IN TYPEOF(SELF.
              edge_element));
END_ENTITY; -- oriented_edge

ENTITY oriented_face
  SUBTYPE OF (face);
    face_element : face;
    orientation  : BOOLEAN;
  DERIVE
    SELF\face.bounds : SET [1:?] OF face_bound := conditional_reverse(
                           SELF.orientation,SELF.face_element.bounds);
  WHERE
    wr1: NOT ('SHIP_ARRANGEMENT_SCHEMA.ORIENTED_FACE' IN TYPEOF(SELF.
              face_element));
END_ENTITY; -- oriented_face

ENTITY oriented_open_shell
  SUBTYPE OF (open_shell);
    open_shell_element : open_shell;
    orientation        : BOOLEAN;
  DERIVE
    SELF\connected_face_set.cfs_faces : SET [1:?] OF face :=
                                          conditional_reverse(SELF.
                                          orientation,SELF.
                                          open_shell_element.cfs_faces);
  WHERE
    wr1: NOT ('SHIP_ARRANGEMENT_SCHEMA.ORIENTED_OPEN_SHELL' IN TYPEOF(
              SELF.open_shell_element));
END_ENTITY; -- oriented_open_shell

ENTITY oriented_path
```



```
  SUBTYPE OF (path);
    path_element : path;
    orientation  : BOOLEAN;
  DERIVE
    SELF\path.edge_list : LIST [1:?] OF UNIQUE oriented_edge :=
                            conditional_reverse(SELF.orientation,SELF.
                            path_element.edge_list);
  WHERE
    wr1: NOT ('SHIP_ARRANGEMENT_SCHEMA.ORIENTED_PATH' IN TYPEOF(SELF.
             path_element));
END_ENTITY; -- oriented_path

ENTITY oriented_surface
  SUBTYPE OF (surface);
    orientation : BOOLEAN;
END_ENTITY; -- oriented_surface

ENTITY parabola
  SUBTYPE OF (conic);
    focal_dist : length_measure;
  WHERE
    wr1: focal_dist <> 0;
END_ENTITY; -- parabola

ENTITY parametric_representation_context
  SUBTYPE OF (representation_context);
END_ENTITY; -- parametric_representation_context

ENTITY path
  SUPERTYPE OF (ONEOF (edge_loop,oriented_path))
  SUBTYPE OF (topological_representation_item);
    edge_list : LIST [1:?] OF UNIQUE oriented_edge;
  WHERE
    wr1: path_head_to_tail(SELF);
END_ENTITY; -- path

ENTITY pcurve
  SUBTYPE OF (curve);
    basis_surface      : surface;
    reference_to_curve : definitional_representation;
  WHERE
    wr1: SIZEOF(reference_to_curve\representation.items) = 1;
    wr2: 'SHIP_ARRANGEMENT_SCHEMA.CURVE' IN TYPEOF(reference_to_curve\
             representation.items[1]);
    wr3: reference_to_curve\representation.items[1]\
             geometric_representation_item.dim = 2;
END_ENTITY; -- pcurve

ENTITY person;
    id            : identifier;
    last_name     : OPTIONAL label;
    first_name    : OPTIONAL label;
    middle_names  : OPTIONAL LIST [1:?] OF label;
    prefix_titles : OPTIONAL LIST [1:?] OF label;
    suffix_titles : OPTIONAL LIST [1:?] OF label;
  WHERE
    wr1: EXISTS(last_name) OR EXISTS(first_name);
END_ENTITY; -- person

ENTITY person_and_organization;
    the_person       : person;
    the_organization : organization;
  DERIVE
    name        : label := get_name_value(SELF);
    description : text := get_description_value(SELF);
  WHERE
```

```
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
             'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
    wr2: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
             'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY; -- person_and_organization

ENTITY person_and_organization_assignment
  ABSTRACT SUPERTYPE;
    assigned_person_and_organization : person_and_organization;
    role                             : person_and_organization_role;
END_ENTITY; -- person_and_organization_assignment

ENTITY person_and_organization_role;
    name : label;
  DERIVE
    description : text := get_description_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
             'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY; -- person_and_organization_role

ENTITY person_assignment
  ABSTRACT SUPERTYPE;
    assigned_person : person;
    role            : person_role;
END_ENTITY; -- person_assignment

ENTITY person_role;
    name : label;
  DERIVE
    description : text := get_description_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
             'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY; -- person_role

ENTITY personal_address
  SUBTYPE OF (address);
    people      : SET [1:?] OF person;
    description : OPTIONAL text;
END_ENTITY; -- personal_address

ENTITY placement
  SUPERTYPE OF (ONEOF (axis1_placement,axis2_placement_2d,
      axis2_placement_3d))
  SUBTYPE OF (geometric_representation_item);
    location : cartesian_point;
END_ENTITY; -- placement

ENTITY plane
  SUBTYPE OF (elementary_surface);
END_ENTITY; -- plane

ENTITY plane_angle_unit
  SUBTYPE OF (named_unit);
  WHERE
    wr1: ((((((SELF\named_unit.dimensions.length_exponent = 0) AND (SELF
             \named_unit.dimensions.mass_exponent = 0)) AND (SELF\
             named_unit.dimensions.time_exponent = 0)) AND (SELF\
             named_unit.dimensions.electric_current_exponent = 0)) AND (
             SELF\named_unit.dimensions.
             thermodynamic_temperature_exponent = 0)) AND (SELF\
             named_unit.dimensions.amount_of_substance_exponent = 0)) AND
              (SELF\named_unit.dimensions.luminous_intensity_exponent = 0);
```

```
END_ENTITY; -- plane_angle_unit

ENTITY point
  SUPERTYPE OF (ONEOF (cartesian_point,point_on_curve,point_on_surface,
      degenerate_pcurve))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- point

ENTITY point_on_curve
  SUBTYPE OF (point);
    basis_curve     : curve;
    point_parameter : parameter_value;
END_ENTITY; -- point_on_curve

ENTITY point_on_surface
  SUBTYPE OF (point);
    basis_surface     : surface;
    point_parameter_u : parameter_value;
    point_parameter_v : parameter_value;
END_ENTITY; -- point_on_surface

ENTITY poly_loop
  SUBTYPE OF (loop, geometric_representation_item);
    polygon : LIST [3:?] OF UNIQUE cartesian_point;
END_ENTITY; -- poly_loop

ENTITY polyline
  SUBTYPE OF (bounded_curve);
    points : LIST [2:?] OF cartesian_point;
END_ENTITY; -- polyline

ENTITY product;
    id                : identifier;
    name              : label;
    description       : OPTIONAL text;
    frame_of_reference : SET [1:?] OF product_context;
END_ENTITY; -- product

ENTITY product_category;
    name        : label;
    description : OPTIONAL text;
  DERIVE
    id : identifier := get_id_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
            'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
END_ENTITY; -- product_category

ENTITY product_context
  SUBTYPE OF (application_context_element);
    discipline_type : label;
END_ENTITY; -- product_context

ENTITY product_definition;
    id                : identifier;
    description       : OPTIONAL text;
    formation         : product_definition_formation;
    frame_of_reference : product_definition_context;
  DERIVE
    name : label := get_name_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
            'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
END_ENTITY; -- product_definition

ENTITY product_definition_context
```

```
  SUBTYPE OF (application_context_element);
    life_cycle_stage : label;
END_ENTITY; -- product_definition_context

ENTITY product_definition_formation;
    id          : identifier;
    description : OPTIONAL text;
    of_product  : product;
  UNIQUE
    ur1 : id, of_product;
END_ENTITY; -- product_definition_formation

ENTITY product_definition_relationship;
    id                          : identifier;
    name                        : label;
    description                 : OPTIONAL text;
    relating_product_definition : product_definition;
    related_product_definition  : product_definition;
END_ENTITY; -- product_definition_relationship

ENTITY product_definition_shape
  SUBTYPE OF (property_definition);
  UNIQUE
    ur1 : definition;
  WHERE
    wr1: SIZEOF([
              'SHIP_ARRANGEMENT_SCHEMA.CHARACTERIZED_PRODUCT_DEFINITION',
              'SHIP_ARRANGEMENT_SCHEMA.CHARACTERIZED_OBJECT'] * TYPEOF(
             SELF\property_definition.definition)) > 0;
END_ENTITY; -- product_definition_shape

ENTITY product_definition_with_associated_documents
  SUBTYPE OF (product_definition);
    documentation_ids : SET [1:?] OF document;
END_ENTITY; -- product_definition_with_associated_documents

ENTITY product_related_product_category
  SUBTYPE OF (product_category);
    products : SET [1:?] OF product;
END_ENTITY; -- product_related_product_category

ENTITY property_definition;
    name        : label;
    description : OPTIONAL text;
    definition  : characterized_definition;
  DERIVE
    id : identifier := get_id_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
END_ENTITY; -- property_definition

ENTITY property_definition_relationship;
    name                          : label;
    description                   : text;
    relating_property_definition : property_definition;
    related_property_definition  : property_definition;
END_ENTITY; -- property_definition_relationship

ENTITY property_definition_representation;
    definition          : represented_definition;
    used_representation : representation;
  DERIVE
    description : text := get_description_value(SELF);
```



```
    name        : label := get_name_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
    wr2: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1;
END_ENTITY; -- property_definition_representation

ENTITY quasi_uniform_curve
  SUBTYPE OF (b_spline_curve);
END_ENTITY; -- quasi_uniform_curve

ENTITY quasi_uniform_surface
  SUBTYPE OF (b_spline_surface);
END_ENTITY; -- quasi_uniform_surface

ENTITY ratio_unit
  SUBTYPE OF (named_unit);
  WHERE
    wr1: ((((((SELF\named_unit.dimensions.length_exponent = 0) AND (SELF
              \named_unit.dimensions.mass_exponent = 0)) AND (SELF\
              named_unit.dimensions.time_exponent = 0)) AND (SELF\
              named_unit.dimensions.electric_current_exponent = 0)) AND (
              SELF\named_unit.dimensions.
              thermodynamic_temperature_exponent = 0)) AND (SELF\
              named_unit.dimensions.amount_of_substance_exponent = 0)) AND
               (SELF\named_unit.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- ratio_unit

ENTITY rational_b_spline_curve
  SUBTYPE OF (b_spline_curve);
    weights_data : LIST [2:?] OF REAL;
  DERIVE
    weights : ARRAY [0:upper_index_on_control_points] OF REAL :=
                 list_to_array(weights_data,0,
                 upper_index_on_control_points);
  WHERE
    wr1: SIZEOF(weights_data) = SIZEOF(SELF\b_spline_curve.
              control_points_list);
    wr2: curve_weights_positive(SELF);
END_ENTITY; -- rational_b_spline_curve

ENTITY rational_b_spline_surface
  SUBTYPE OF (b_spline_surface);
    weights_data : LIST [2:?] OF LIST [2:?] OF REAL;
  DERIVE
    weights : ARRAY [0:u_upper] OF ARRAY [0:v_upper] OF REAL :=
                 make_array_of_array(weights_data,0,u_upper,0,v_upper);
  WHERE
    wr1: (SIZEOF(weights_data) = SIZEOF(SELF\b_spline_surface.
              control_points_list)) AND (SIZEOF(weights_data[1]) = SIZEOF(
              SELF\b_spline_surface.control_points_list[1]));
    wr2: surface_weights_positive(SELF);
END_ENTITY; -- rational_b_spline_surface

ENTITY representation;
    name             : label;
    items            : SET [1:?] OF representation_item;
    context_of_items : representation_context;
  DERIVE
    id          : identifier := get_id_value(SELF);
    description : text := get_description_value(SELF);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
              'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
    wr2: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
```

```
            'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1;
END_ENTITY; -- representation

ENTITY representation_context;
    context_identifier : identifier;
    context_type       : text;
  INVERSE
    representations_in_context : SET [1:?] OF representation FOR
                                       context_of_items;
END_ENTITY; -- representation_context

ENTITY representation_item;
    name : label;
  WHERE
    wr1: SIZEOF(using_representations(SELF)) > 0;
END_ENTITY; -- representation_item

ENTITY representation_map;
    mapping_origin        : representation_item;
    mapped_representation : representation;
  INVERSE
    map_usage : SET [1:?] OF mapped_item FOR mapping_source;
  WHERE
    wr1: item_in_context(SELF.mapping_origin,SELF.mapped_representation.
            context_of_items);
END_ENTITY; -- representation_map

ENTITY representation_relationship;
    name        : label;
    description : OPTIONAL text;
    rep_1       : representation;
    rep_2       : representation;
END_ENTITY; -- representation_relationship

ENTITY role_association;
    role           : object_role;
    item_with_role : role_select;
END_ENTITY; -- role_association
ENTITY seam_curve
  SUBTYPE OF (surface_curve);
  WHERE
    wr1: SIZEOF(SELF\surface_curve.associated_geometry) = 2;
    wr2: associated_surface(SELF\surface_curve.associated_geometry[1]) =

associated_surface(SELF\surface_curve.associated_geometry[2]);
    wr3: 'SHIP_ARRANGEMENT_SCHEMA.PCURVE' IN TYPEOF(SELF\surface_curve.
             associated_geometry[1]);
    wr4: 'SHIP_ARRANGEMENT_SCHEMA.PCURVE' IN TYPEOF(SELF\surface_curve.
            associated_geometry[2]);
END_ENTITY; -- seam_curve

ENTITY serial_numbered_effectivity
  SUBTYPE OF (effectivity);
    effectivity_start_id : identifier;
    effectivity_end_id : OPTIONAL identifier;
END_ENTITY; -- serial_numbered_effectivity

ENTITY shape_aspect;
    name                 : label;
    description          : OPTIONAL text;
    of_shape             : product_definition_shape;
    product_definitional : LOGICAL;
  DERIVE
    id : identifier := get_id_value(SELF);
  WHERE
```



```
    wr1: SIZEOF(USEDIN(SELF,'SHIP_ARRANGEMENT_SCHEMA.' +
             'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1;
END_ENTITY; -- shape_aspect

ENTITY shape_definition_representation
    SUBTYPE OF (property_definition_representation);
    WHERE
      wr1: (('SHIP_ARRANGEMENT_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN
                TYPEOF(SELF.definition)) OR (
                'SHIP_ARRANGEMENT_SCHEMA.SHAPE_DEFINITION' IN
   TYPEOF(SELF.definition.definition)));
      wr2: ('SHIP_ARRANGEMENT_SCHEMA.SHAPE_REPRESENTATION' IN
   TYPEOF(SELF.used_representation));
END_ENTITY; -- shape_definition_representation

ENTITY shape_representation
  SUBTYPE OF (representation);
END_ENTITY; -- shape_representation

ENTITY si_unit
  SUBTYPE OF (named_unit);
    prefix : OPTIONAL si_prefix;
    name   : si_unit_name;
  DERIVE
    SELF\named_unit.dimensions : dimensional_exponents :=
                                    dimensions_for_si_unit(name);
END_ENTITY; -- si_unit

ENTITY spherical_surface
  SUBTYPE OF (elementary_surface);
    radius : positive_length_measure;
END_ENTITY; -- spherical_surface

ENTITY surface
  SUPERTYPE OF (ONEOF (elementary_surface,swept_surface,bounded_surface,
      offset_surface,surface_replica))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- surface

ENTITY surface_curve
  SUPERTYPE OF (ONEOF (intersection_curve,seam_curve) ANDOR
      bounded_surface_curve)
  SUBTYPE OF (curve);
    curve_3d              : curve;
    associated_geometry   : LIST [1:2] OF pcurve_or_surface;
    master_representation : preferred_surface_curve_representation;
  DERIVE
    basis_surface : SET [1:2] OF surface := get_basis_surface(SELF);
  WHERE
    wr1: curve_3d.dim = 3;
    wr2: ('SHIP_ARRANGEMENT_SCHEMA.PCURVE' IN TYPEOF(
              associated_geometry[1])) OR (master_representation <>
              pcurve_s1);
    wr3: ('SHIP_ARRANGEMENT_SCHEMA.PCURVE' IN TYPEOF(
              associated_geometry[2])) OR (master_representation <>
              pcurve_s2);
    wr4: NOT ('SHIP_ARRANGEMENT_SCHEMA.PCURVE' IN TYPEOF(curve_3d));
END_ENTITY; -- surface_curve

ENTITY surface_of_linear_extrusion
  SUBTYPE OF (swept_surface);
    extrusion_axis : vector;
END_ENTITY; -- surface_of_linear_extrusion

ENTITY surface_of_revolution
  SUBTYPE OF (swept_surface);
```

```
    axis_position : axis1_placement;
  DERIVE
    axis_line : line := (dummy_gri || curve()) || line(axis_position.
                  location,dummy_gri || vector(axis_position.z,1));
END_ENTITY; -- surface_of_revolution

ENTITY surface_replica
  SUBTYPE OF (surface);
    parent_surface : surface;
    transformation : cartesian_transformation_operator_3d;
  WHERE
    wr1: acyclic_surface_replica(SELF,parent_surface);
END_ENTITY; -- surface_replica

ENTITY swept_surface
  SUPERTYPE OF (ONEOF (surface_of_linear_extrusion,surface_of_revolution))
  SUBTYPE OF (surface);
    swept_curve : curve;
END_ENTITY; -- swept_surface

ENTITY thermodynamic_temperature_unit
  SUBTYPE OF (named_unit);
  WHERE
    wr1: ((((((SELF\named_unit.dimensions.length_exponent = 0) AND (SELF
              \named_unit.dimensions.mass_exponent = 0)) AND (SELF\
              named_unit.dimensions.time_exponent = 0)) AND (SELF\
              named_unit.dimensions.electric_current_exponent = 0)) AND (
              SELF\named_unit.dimensions.
              thermodynamic_temperature_exponent = 1)) AND (SELF\
              named_unit.dimensions.amount_of_substance_exponent = 0)) AND
               (SELF\named_unit.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- thermodynamic_temperature_unit

ENTITY time_unit
  SUBTYPE OF (named_unit);
  WHERE
    wr1: ((((((SELF\named_unit.dimensions.length_exponent = 0) AND (SELF
              \named_unit.dimensions.mass_exponent = 0)) AND (SELF\
              named_unit.dimensions.time_exponent = 1)) AND (SELF\
              named_unit.dimensions.electric_current_exponent = 0)) AND (
              SELF\named_unit.dimensions.
              thermodynamic_temperature_exponent = 0)) AND (SELF\
              named_unit.dimensions.amount_of_substance_exponent = 0)) AND
               (SELF\named_unit.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- time_unit

ENTITY topological_representation_item
  SUPERTYPE OF (ONEOF (vertex,edge,face_bound,face,connected_face_set,
      loop ANDOR path))
  SUBTYPE OF (representation_item);
END_ENTITY; -- topological_representation_item

ENTITY toroidal_surface
  SUBTYPE OF (elementary_surface);
    major_radius : positive_length_measure;
    minor_radius : positive_length_measure;
END_ENTITY; -- toroidal_surface

ENTITY uncertainty_measure_with_unit
  SUBTYPE OF (measure_with_unit);
    name        : label;
    description : OPTIONAL text;
  WHERE
    wr1: valid_measure_value(SELF\measure_with_unit.value_component);
```



```
END_ENTITY; -- uncertainty_measure_with_unit

ENTITY uniform_curve
  SUBTYPE OF (b_spline_curve);
END_ENTITY; -- uniform_curve

ENTITY uniform_surface
  SUBTYPE OF (b_spline_surface);
END_ENTITY; -- uniform_surface

ENTITY value_representation_item
  SUBTYPE OF (representation_item);
    value_component : measure_value;
  WHERE
    wr1: SIZEOF(QUERY ( rep <* using_representations(SELF) | (NOT (
              'SHIP_ARRANGEMENT_SCHEMA.GLOBAL_UNIT_ASSIGNED_CONTEXT' IN
             TYPEOF(rep.context_of_items))) )) = 0;
END_ENTITY; -- value_representation_item

ENTITY vector
  SUBTYPE OF (geometric_representation_item);
    orientation : direction;
    magnitude   : length_measure;
  WHERE
    wr1: magnitude >= 0;
END_ENTITY; -- vector

ENTITY versioned_action_request;
    id          : identifier;
    version     : label;
    purpose     : text;
    description : OPTIONAL text;
END_ENTITY; -- versioned_action_request

ENTITY vertex
  SUBTYPE OF (topological_representation_item);
END_ENTITY; -- vertex

ENTITY vertex_loop
  SUBTYPE OF (loop);
    loop_vertex : vertex;
END_ENTITY; -- vertex_loop

ENTITY vertex_point
  SUBTYPE OF (vertex, geometric_representation_item);
    vertex_geometry : point;
END_ENTITY; -- vertex_point

ENTITY week_of_year_and_day_date
  SUBTYPE OF (date);
    week_component : week_in_year_number;
    day_component  : OPTIONAL day_in_week_number;
END_ENTITY; -- week_of_year_and_day_date


RULE action_request_solution_connected_to_action FOR
(action_request_solution, action );
   LOCAL
      t1_set : SET OF action_request_solution := [];
      t2_set : SET OF action := [];
      set_3 : SET OF action_method := [];
      violate : LOGICAL := FALSE;
   END_LOCAL;
      t1_set := QUERY (a <* action_request_solution|
VALUE_IN(WHICH_CLASS(a), 'change plan'));
      t2_set := QUERY (b <* action| VALUE_IN(WHICH_CLASS(b), 'change'));
```

```
      REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
         set_3 := [];
         REPEAT j := 1 TO HIINDEX(t2_set);
            set_3 := set_3 + [ t2_set[j].chosen_method ];
         END_REPEAT;
         violate := VALUE_IN(set_3, t1_set[i].method);
      END_REPEAT;
   WHERE
      WR1:
         NOT violate;
END_RULE;

RULE action_request_solution_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF action_request_solution := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := ['change plan'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
     t2_set := QUERY ( j <* t2_set | j.role.name = 'globally unambiguous
identifier');

      violation := NOT (SIZEOF(t2_set) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE action_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF action := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := ['change', 'versionable object change
event', 'check'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
```

```
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
  'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
     t2_set := QUERY ( j <* t2_set | j.role.name = 'globally unambiguous
identifier');
      violation := NOT (SIZEOF(t2_set) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE alternative_version_relationship_has_mandatory_description FOR
(identification_assignment_relationship );
   LOCAL
      t1_set : SET OF identification_assignment_relationship := [];
      violate : LOGICAL := FALSE;
   END_LOCAL;
      t1_set := QUERY (i <* identification_assignment_relationship|
VALUE_IN(WHICH_CLASS(i), 'alternative version relationship'));
      violate := SIZEOF(QUERY (k <* t1_set | NOT EXISTS(k.description))) >
0;
   WHERE
      WR1:
         NOT violate;
END_RULE;

RULE alternative_version_relationship_has_unique_versions
FOR (identification_assignment_relationship );
   LOCAL
      t1_set : SET OF identification_assignment_relationship := [];
      violate : LOGICAL := FALSE;
   END_LOCAL;
      t1_set := QUERY (a <* identification_assignment_relationship|
VALUE_IN(WHICH_CLASS(a), 'alternative version relationship'));
      REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
         violate :=
t1_set[i].relating_identification_assignment.assigned_id =
t1_set[i].related_identification_assignment.assigned_id;
      END_REPEAT;
   WHERE
      WR1:
         NOT violate;
END_RULE;

RULE alternative_version_relationship_versionable_object FOR
(applied_identification_assignment,
     identification_assignment_relationship);
LOCAL
 violate: LOGICAL := FALSE;
END_LOCAL;

REPEAT i := 1 TO HIINDEX(applied_identification_assignment) BY 1 WHILE NOT
violate;

 IF ( (SIZEOF(USEDIN(applied_identification_assignment[i],

('SHIP_ARRANGEMENT_SCHEMA.IDENTIFICATION_ASSIGNMENT_RELATIONSHIP.'
             +'RELATING_IDENTIFICATION_ASSIGNMENT'))) > 0) OR
      (SIZEOF(USEDIN(applied_identification_assignment[i],
```

```
('SHIP_ARRANGEMENT_SCHEMA.IDENTIFICATION_ASSIGNMENT_RELATIONSHIP.'
             +'RELATED_IDENTIFICATION_ASSIGNMENT'))) > 0) ) THEN
  REPEAT j := 1 to HIINDEX(applied_identification_assignment[i].items) BY 1
WHILE NOT violate;
 violate := NOT VALUE_IN(which_class(
applied_identification_assignment [i].items[j]),'versionable object');
  END_REPEAT;
 END_IF;

END_REPEAT;

WHERE
wr1: NOT violate;
END_RULE;

RULE applied_approval_assignment_has_exactly_one_elements FOR (object_role,
applied_approval_assignment );
   WHERE
      WR1:
         SIZEOF(QUERY (ass_inst <* applied_approval_assignment| NOT
((ass_inst.role.NAME = 'subject') AND (SIZEOF(ass_inst.items) = 1)))) = 0;
END_RULE;

RULE applied_group_assignment_has_at_least_one_elements FOR (object_role,
applied_group_assignment );
   WHERE
      WR1:
         SIZEOF(QUERY (ass_inst <* applied_group_assignment| NOT
(((ass_inst.role.NAME = 'approvals') AND (SIZEOF(ass_inst.items) >= 1)) AND
(SIZEOF(QUERY (item <* ass_inst.items | NOT
('SHIP_ARRANGEMENT_SCHEMA.APPROVAL' IN TYPEOF(item)))) = 0)))) = 0;
END_RULE;

RULE approval_event_with_approval_date_time
FOR(approval);
LOCAL
    t1_set:  SET OF approval := [];
    t2_set:  SET OF approval_date_time := [];
    violate: LOGICAL := FALSE;

  END_LOCAL;

t1_set := QUERY(i <* approval |
 VALUE_IN(WHICH_CLASS(i), 'approval event'));

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
    t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPROVAL_DATE_TIME.' +
'DATED_APPROVAL'));
    violate := NOT (SIZEOF(t2_set) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violate;
END_RULE;

RULE approval_event_with_approval_person_organization FOR(approval);
LOCAL
    t1_set:  SET OF approval := [];
    t2_set:  SET OF approval_person_organization := [];
    violate: LOGICAL := FALSE;

  END_LOCAL;
```



```
t1_set := QUERY(i <* approval |
 VALUE_IN(WHICH_CLASS(i), 'approval event'));

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
    t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPROVAL_PERSON_ORGANIZATION.' +
'AUTHORIZED_APPROVAL'));

    violate := NOT (SIZEOF(t2_set) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violate;
END_RULE;

RULE approval_history_approves_same_definition FOR
(applied_group_assignment, applied_approval_assignment);
LOCAL
    t2_set:  SET OF APPLIED_GROUP_ASSIGNMENT := [];
    t3_set:  SET OF APPROVAL :=[];
    t4_set:  SET OF group_item :=[];
    t5_set:  SET OF APPLIED_APPROVAL_ASSIGNMENT := [];
    violate: LOGICAL := FALSE;
END_LOCAL;

t2_set := QUERY(a <* APPLIED_GROUP_ASSIGNMENT |
                VALUE_IN(WHICH_CLASS(a.ASSIGNED_GROUP),
          'approval history'));

t3_set := QUERY(b <* t2_set[1].items |
 'SHIP_ARRANGEMENT_SCHEMA.APPROVAL' IN TYPEOF(b));

t4_set := QUERY(b <* t2_set[1].items |
 VALUE_IN(WHICH_CLASS(b), 'definition'));

violate := NOT(SIZEOF(t4_set) = 1);

REPEAT i := 1 TO HIINDEX(t3_set) WHILE NOT violate;
    t5_set := QUERY(a <* APPLIED_APPROVAL_ASSIGNMENT |
(a.ASSIGNED_APPROVAL = t3_set[i]) AND
(NOT (VALUE_IN(a.ITEMS, t4_set[1]))));

    violate := (SIZEOF(t5_set) > 0);
  END_REPEAT;

WHERE
    wr1: NOT violate;
    wr2: (SIZEOF(t4_set) = 1);
END_RULE;

RULE approval_history_has_at_least_one_member FOR (GROUP,
APPLIED_GROUP_ASSIGNMENT);
LOCAL
    t1_set:  SET OF GROUP := [];
    t2_set:  SET OF APPLIED_GROUP_ASSIGNMENT := [];
    violate: LOGICAL := FALSE;
END_LOCAL;

t1_set := QUERY(i <* group |
 VALUE_IN(WHICH_CLASS(i), 'approval history'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
    t2_set := QUERY(a <* APPLIED_GROUP_ASSIGNMENT |
      a.ASSIGNED_GROUP = t1_set[i]);
```

```
    violate := NOT(SIZEOF(t2_set) = 1);
  END_REPEAT;

WHERE
    wr1: NOT violate;
END_RULE;

RULE approvals_references_approval_history
FOR(applied_group_assignment, group);
  LOCAL
    t1_set:  SET OF group := [];
    a_set:  SET OF applied_group_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

t1_set := QUERY(a <* group |
 VALUE_IN(WHICH_CLASS(a), 'approval history'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_group_assignment |
   NOT ((b.assigned_group = t1_set[i]) AND (b.role.name = 'approvals')));

    violate := SIZEOF(a_set) > 0;
  END_REPEAT;

WHERE
    wr1: NOT violate;
END_RULE;

RULE author_for_change_plan
FOR(applied_person_and_organization_assignment, action_request_solution);
  LOCAL
    t1_set:  SET OF action_request_solution := [];
    a_set:  SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action_request_solution |
             VALUE_IN(WHICH_CLASS(a), 'change plan'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'author')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE author_for_change_realization
FOR(applied_person_and_organization_assignment,executed_action);
  LOCAL
    t1_set:  SET OF executed_action := [];
    a_set:  SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* executed_action |
             VALUE_IN(WHICH_CLASS(a), 'change realization'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'author')));
    violate := SIZEOF(a_set) <> 1;
```



```
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE author_for_change_request
FOR(applied_person_and_organization_assignment, versioned_action_request);
  LOCAL
    t1_set:  SET OF versioned_action_request := [];
    a_set:  SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* versioned_action_request |
             VALUE_IN(WHICH_CLASS(a), 'change request'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'author')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE caused_by_for_check FOR(applied_person_and_organization_assignment,
action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action | VALUE_IN(WHICH_CLASS(a), 'check'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused by')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE caused_by_for_envisaged_version_creation
FOR(applied_person_and_organization_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action |
             VALUE_IN(WHICH_CLASS(a), 'envisaged version creation'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused by')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE caused_by_for_version_creation
FOR(applied_person_and_organization_assignment, action);
```

```
  LOCAL
    t1_set:  SET OF action := [];
    a_set:   SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action |
             VALUE_IN(WHICH_CLASS(a), 'version creation'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused by')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE caused_by_for_version_deletion
FOR(applied_person_and_organization_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action |
             VALUE_IN(WHICH_CLASS(a), 'version deletion'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused by')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE caused_by_for_version_modification
FOR(applied_person_and_organization_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:   SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action |
             VALUE_IN(WHICH_CLASS(a), 'version modification'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused by')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE caused_when_for_check
FOR(applied_date_and_time_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
```



```
  END_LOCAL;
   t1_set := QUERY(a <* action | VALUE_IN(WHICH_CLASS(a), 'check'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused when')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE caused_when_for_envisaged_version_creation
FOR(applied_date_and_time_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action | VALUE_IN(WHICH_CLASS(a),
             'envisaged version creation'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused when')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE caused_when_for_version_creation
FOR(applied_date_and_time_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action |
             VALUE_IN(WHICH_CLASS(a), 'version creation'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused when')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE caused_when_for_version_deletion
FOR(applied_date_and_time_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action |
             VALUE_IN(WHICH_CLASS(a), 'version deletion'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
```

```
              (b.role.name = 'caused when')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE caused_when_for_version_modification
FOR(applied_date_and_time_assignment, action);
  LOCAL
    t1_set:  SET OF action := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action |
             VALUE_IN(WHICH_CLASS(a), 'version modification'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'caused when')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE centre_location_compound_representation_has_specified_name FOR
(APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF Compound_representation_item := [];
    t2_set:  SET OF representation_item := [];
    arg_list:  LIST OF STRING := ['longitudinal location', 'transversal
location', 'vertical location'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'centre location');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
   violation := (SIZEOF(QUERY(items <* t2_set | items.name = arg_list[j]))
   <> 1);
 END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;

END_RULE;

RULE change_impact_with_versionable_object_change_event
FOR(applied_action_request_assignment);
  LOCAL
    t1_set:  SET OF applied_action_request_assignment := [];
```



```
    a_set:   SET OF action := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

   t1_set := QUERY(b <* applied_action_request_assignment |
                  (b.role.name= 'change impact'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
          a_set := QUERY(b <* t1_set[i].items |
          ('SHIP_ARRANGEMENT_SCHEMA.ACTION' IN TYPEOF(b)) AND
             VALUE_IN(WHICH_CLASS(b), 'versionable object change event'));
         violate := SIZEOF(a_set) = 0;
      END_REPEAT;

WHERE
    WR1: NOT violate;
END_RULE;

RULE change_plan_has_mandatory_attribute_description
FOR (action_request_solution);
LOCAL
    t1_set:  SET OF action_request_solution := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
  t1_set := QUERY(i <* action_request_solution |
  VALUE_IN(WHICH_CLASS(i), 'change plan'));
  violate := (SIZEOF(QUERY(k <* t1_set |
  NOT EXISTS (k.description))) > 0);
WHERE
    WR1: NOT violate;
END_RULE;

RULE class_and_statutory_designation_has_properties
FOR (property_definition_representation,
applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_list: LIST OF product_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    t3_list: LIST OF property_definition := [];
    t4_list: LIST OF product_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME =
                  'class and statutory designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_list := t1_list + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
                  i.NAME = 'class and statutory designation');
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_list := t3_list + t2_set[i].definition;
  END_REPEAT;
  REPEAT i := 1 TO HIINDEX(t3_list);
    t4_list := t4_list + t3_list[i].definition;
  END_REPEAT;
  violation := t1_list <> t4_list;
  WHERE
    WR1: NOT violation;
```

```
END_RULE;

RULE class_notation_with_named_representation_items
FOR (representation);
LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['class notations hull',
                                  'class notations machinery'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
             'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
             'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'class notation')
                    )
              ) > 0
           );

  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) < 1);
    END_REPEAT;
  END_REPEAT;
  WHERE
    WR1: NOT violation;
END_RULE;

RULE class_parameters_has_properties
FOR (property_definition_representation,
applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set: LIST OF product_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    t3_set: LIST OF property_definition := [];
    t4_set: LIST OF product_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME =
                  'class parameters');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
                  i.NAME = 'class parameters');
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
  END_REPEAT;
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;
  violation := t1_set <> t4_set;
```



```
  WHERE
    WR1: NOT violation;

END_RULE;

RULE compatible_dimension FOR (cartesian_point, direction,
representation_context, geometric_representation_context );
   WHERE
      WR1:
         SIZEOF(QUERY (x <* cartesian_point| (SIZEOF(QUERY (y <*
geometric_representation_context| item_in_context(x, y) AND
(HIINDEX(x.coordinates) <> y.coordinate_space_dimension))) > 0))) = 0;
      WR2:
         SIZEOF(QUERY (x <* direction| (SIZEOF(QUERY (y <*
geometric_representation_context| item_in_context(x, y) AND
(HIINDEX(x.direction_ratios) <> y.coordinate_space_dimension))) > 0))) = 0;
END_RULE;

RULE date_time_for_change_plan
FOR(applied_date_and_time_assignment,  action_request_solution);
  LOCAL
    t1_set:  SET OF action_request_solution := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* action_request_solution |
             VALUE_IN(WHICH_CLASS(a), 'change plan'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
               (VALUE_IN(b.items, t1_set[i]) AND
               (b.role.name = 'date time')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE date_time_for_change_realization
FOR(applied_date_and_time_assignment, executed_action);
  LOCAL
    t1_set:  SET OF executed_action := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* executed_action |
             VALUE_IN(WHICH_CLASS(a), 'change realization'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
               (VALUE_IN(b.items, t1_set[i]) AND
               (b.role.name = 'date time')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE date_time_for_change_request
FOR(applied_date_and_time_assignment,  versioned_action_request);
  LOCAL
    t1_set:  SET OF versioned_action_request := [];
    a_set:  SET OF applied_date_and_time_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* versioned_action_request |
```

```
            VALUE_IN(WHICH_CLASS(a), 'change request'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_date_and_time_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'date time')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE document_has_at_least_one_references
FOR(document);
  LOCAL
    t1_set:  SET OF document := [];
    t2_set:  SET OF document_representation_type := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
  t1_set := QUERY(i <* document | VALUE_IN(WHICH_CLASS(i),
                                    'document'));

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
    t2_set := bag_to_set(USEDIN(t1_set[i],
      'SHIP_ARRANGEMENT_SCHEMA.DOCUMENT_REPRESENTATION_TYPE.'+
      'REPRESENTED_DOCUMENT'));
    violate := SIZEOF(t2_set) < 1;
END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE document_has_exactly_one_author
FOR(document);
  LOCAL
    bag_1: BAG OF applied_person_assignment := [];
    bag_2: BAG OF applied_person_and_organization_assignment := [];
    bag_3: BAG OF applied_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
REPEAT i := 1 TO SIZEOF(document) WHILE (NOT violate);
  bag_1 := USEDIN(document[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
         'APPLIED_PERSON_ASSIGNMENT.ITEMS');
     bag_1 := QUERY( assign <* bag_1 | assign.role.name = ‘author’);
   bag_2 := USEDIN(document[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
         'APPLIED_PERSON_AND_ORGANIZATION_ASSIGNMENT.ITEMS');
  bag_2 := QUERY( assign <* bag_2 | assign.role.name = ‘author’);
  bag_3 := USEDIN(document[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
         'APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS');

  bag_3 := QUERY( assign <* bag_3 | assign.role.name = ‘author’);
    violate := NOT ((SIZEOF( bag_1 ) + SIZEOF( bag_2 )+ SIZEOF( bag_3 ))=
1);
END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE document_reference_with_address_has_at_least_one_references
FOR(document);
  LOCAL
    t1_set:  SET OF document := [];
    t2_set:  SET OF applied_external_identification_assignment := [];
    violate: LOGICAL := FALSE;
```



```
  END_LOCAL;
t1_set := QUERY(i <* document | VALUE_IN(WHICH_CLASS(i),
                                'document reference with address'));

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
    t2_set := bag_to_set(USEDIN(t1_set[i],

'SHIP_ARRANGEMENT_SCHEMA.APPLIED_EXTERNAL_IDENTIFICATION_ASSIGNMENT.ITEMS')
);
    violate := SIZEOF(t2_set) < 1;
END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE envisaged_version_creation_has_mandatory_attribute_description
FOR (action);
LOCAL
    t1_set:  SET OF action := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(i <* action | VALUE_IN(WHICH_CLASS(i),
   'envisaged version creation'));
   violate := (SIZEOF(QUERY(k <* t1_set |
   NOT EXISTS (k.description))) > 0);
WHERE
    WR1: NOT violate;
END_RULE;

RULE executed_action_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF executed_action := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := [ 'change realization'];
    violation: LOGICAL := FALSE;
  END_LOCAL;


    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
   t2_set := QUERY ( j <* t2_set | j.role.name = 'globally unambiguous
identifier');
       violation := NOT (SIZEOF(T2_SET) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE external_instance_reference_has_same_identifier
FOR (applied_external_identification_assignment);
```

```
LOCAL
 violation    : LOGICAL := FALSE;
 extref_set   : SET OF applied_external_identification_assignment := [];
 aia_set      : SET OF applied_identification_assignment := [];
END_LOCAL;

 extref_set := QUERY ( i <* applied_external_identification_assignment |
(i.role.name = 'external instance reference') );

 REPEAT i := 1 TO HIINDEX(extref_set) BY 1 WHILE NOT violation;
   aia_set := USEDIN(extref_set[i].items[1],
       'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS');
   violation := NOT (aia_set[1].assigned_id = extref_set[i].assigned_id);
 END_REPEAT;
 WHERE
   wr1: NOT violation;
 END_RULE;

RULE floating_position_compound_representation_with_name
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF Compound_representation_item := [];
    t2_set:  SET OF representation_item := [];
    arg_list:  LIST OF STRING := ['moulded form displacement',
  'draught at amidships', 'length of waterline', 'breadth of waterline',
  'angle of trim', 'angle of heel'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.Assigned_class.name = 'floating position');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
  violation := (SIZEOF(QUERY(items <* t2_set |
     items.name = arg_list[j])) <> 1);
 END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;

END_RULE;

RULE freeboard_characteristics_has_properties
FOR (property_definition_representation,
applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set: LIST OF product_definition := [];
    t2_set: SET OF property_definition_representation := [];
    t3_set: LIST OF property_definition := [];
    t4_set: LIST OF product_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;
```



```
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =
                   'freeboard characteristics');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
                   i.NAME = 'freeboard characteristics');
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
  END_REPEAT;
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;
  violation := t1_set <> t4_set;
  WHERE
    WR1: NOT violation;

END_RULE;

RULE global_axis_placement_has_properties
FOR (property_definition_representation,
group, applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set: LIST OF product_definition := [];
    t2_set: SET OF property_definition_representation := [];
    t3_set: LIST OF property_definition := [];
    t4_set: LIST OF product_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =
                   'global axis placement');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
                   i.NAME = 'global axis placement');
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
  END_REPEAT;
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;
  violation := t1_set <> t4_set;
  WHERE
    WR1: NOT violation;

END_RULE;

RULE global_id_is_unique
FOR (APPLIED_IDENTIFICATION_ASSIGNMENT);
  LOCAL
    set_1:  SET OF APPLIED_IDENTIFICATION_ASSIGNMENT:= [];
    bag_2:     BAG OF STRING := [];
    violation: LOGICAL := FALSE;
```

```
  END_LOCAL;



  set_1 := QUERY(i <* APPLIED_IDENTIFICATION_ASSIGNMENT |
          (i.role.name = 'globally unambiguous identifier'));

REPEAT i := 1 TO HIINDEX(set_1);
    bag_2 := bag_2 + [set_1[i].assigned_id];

END_REPEAT;
violation := SIZEOF (QUERY(i <* set_1 | (SIZEOF(i.items) = 1))) <>
SIZEOF(set_1);

WHERE
    WR1: VALUE_UNIQUE(bag_2);
    WR2: NOT violation;
END_RULE;

RULE identification_role_optional_attribute_description_required FOR
(identification_role );
   WHERE
      WR1:
         SIZEOF(QUERY (i <* identification_role|
 (i.NAME = 'external reference') AND NOT EXISTS(i.description))) = 0;
END_RULE;

RULE initiator_for_change_request
FOR(applied_person_and_organization_assignment,
    versioned_action_request);
  LOCAL
    t1_set:  SET OF versioned_action_request := [];
    a_set:  SET OF applied_person_and_organization_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
   t1_set := QUERY(a <* versioned_action_request |
             VALUE_IN(WHICH_CLASS(a), 'change request'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
  a_set := QUERY(b <* applied_person_and_organization_assignment |
              (VALUE_IN(b.items, t1_set[i]) AND
              (b.role.name = 'initiator')));
    violate := SIZEOF(a_set) <> 1;
  END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE lightship_definition_has_properties
FOR (property_definition_representation,
group, applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set: LIST OF product_definition := [];
    t2_set: SET OF property_definition_representation := [];
    t3_set: LIST OF property_definition := [];
    t4_set: LIST OF product_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME =
                  'lightship definition');
```

```
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
                  i.NAME = 'lightship definition parameters');
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
  END_REPEAT;
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;
  violation := t1_set <> t4_set;
  WHERE
    WR1: NOT violation;

END_RULE;

RULE mandatory_entity_type_for_external_instance_reference
FOR(external_source, external_source_relationship);
  LOCAL
    bag_1: BAG OF external_source := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
bag_1 := QUERY(a <* external_source |
            a.description = 'schema name');

REPEAT i := 1 TO SIZEOF(bag_1) WHILE (NOT violate);
violate := (SIZEOF( QUERY(a <* external_source_relationship |
            (a.relating_source :=: bag_1[i]) AND
            (a.related_source.description = 'entity type'))) = 0 );
END_REPEAT;
  WHERE
    WR1: NOT violate;
END_RULE;

RULE members_is_referenced_by_at_least_one_revision
FOR(applied_group_assignment, group);
  LOCAL
    t1_set:  SET OF group := [];
    a_set:  SET OF applied_group_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

t1_set := QUERY(a <* group | VALUE_IN(WHICH_CLASS(a), 'revision'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
   a_set := QUERY(b <* applied_group_assignment |
    (b.assigned_group :=: t1_set[i]) AND (b.role.name = 'members'));

    violate := SIZEOF(a_set) < 1;
  END_REPEAT;

WHERE
    wr1: NOT violate;
END_RULE;

RULE no_approvals_except_in_approval_history
FOR (approval);
LOCAL
    t1_set:  SET OF approval := [];
    t2_set:  SET OF APPLIED_GROUP_ASSIGNMENT := [];
    violate: LOGICAL := FALSE;
END_LOCAL;
```

```
   t1_set := QUERY(a <* approval |
      VALUE_IN(WHICH_CLASS(a), 'approval event'));
REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
   t2_set := bag_to_set(USEDIN(t1_set[i],
   'SHIP_ARRANGEMENT_SCHEMA.APPLIED_GROUP_ASSIGNMENT.ITEMS'));
    violate := (SIZEOF(t2_set) = 0);
   REPEAT k := 1 TO HIINDEX(t2_set) WHILE NOT violate;
      violate := NOT (VALUE_IN(WHICH_CLASS(t2_set[k].ASSIGNED_GROUP),
      'approval history'));
   END_REPEAT;
END_REPEAT;
WHERE
    WR1: NOT violate;
END_RULE;

RULE principal_characteristics_has_properties
FOR (property_definition_representation,
applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set: LIST OF product_definition := [];
    t2_set: SET OF property_definition_representation := [];
    t3_set: LIST OF property_definition := [];
    t4_set: LIST OF product_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
          i.assigned_class.NAME = 'principal characteristics');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
                  i.NAME = 'principal characteristics');
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
  END_REPEAT;
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;
  violation := t1_set <> t4_set;
  WHERE
    WR1: NOT violation;

END_RULE;

RULE product_definition_for_call_sign
FOR(applied_classification_assignment);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_identification_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME = 'ship designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
```



```
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
        'call sign')) = 1);
  END_REPEAT;
  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_definition_for_certifying_organization
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
             i.assigned_class.name = 'coating certification');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
   t2_inst.role.name = 'certifying organization')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_definition_for_class_notation
FOR(applied_classification_assignment);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF property_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =
                   'class and statutory designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
      'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
```

```
      'class notation' IN WHICH_CLASS(t2_inst))) = 1);
END_REPEAT;
  WHERE
    WR1: NOT violation;
END_RULE;

RULE product_definition_for_expiry_date
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_date_and_time_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.name = 'coating certification');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.role.name = 'expiry date')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_definition_for_flag_state
FOR(applied_classification_assignment);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_identification_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME = 'ship designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
    'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
    'flag state')) = 1);
  END_REPEAT;
  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_definition_for_loadline
```



```
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF property_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'freeboard characteristics');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
   'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
   'loadline' IN WHICH_CLASS(t2_inst))) = 1);
END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_definition_for_managing_company
FOR(applied_classification_assignment);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =
                    'owner designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
    'APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
    'managing company')) = 1);
  END_REPEAT;
  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_definition_for_ordering_company
FOR(applied_classification_assignment);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;
```

```
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME = 'owner designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
    'APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
    'ordering company')) = 1);
  END_REPEAT;
  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_definition_for_owning_company
FOR(applied_classification_assignment);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =
                   'owner designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
     'SHIP_ARRANGEMENT_SCHEMA.APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
    'owning company')) = 1);
  END_REPEAT;
  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_definition_for_port_of_registration
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_identification_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME = 'ship designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
```

```
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
    'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
     'port of registration')) = 1);
  END_REPEAT;
  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_definition_for_regulation
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF property_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME =
                  'class and statutory designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
     'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | 'regulation'
    IN WHICH_CLASS(t2_inst))) = 1);
END_REPEAT;
  WHERE
    WR1: NOT violation;
END_RULE;

RULE product_definition_for_shipyard
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME =
                  'shipyard designation');
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
    'SHIP_ARRANGEMENT_SCHEMA.APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
    'shipyard')) = 1);
  END_REPEAT;
  WHERE
```

```
    wr1: NOT violation;
END_RULE;

RULE product_definition_relationship_references_are_distinct
FOR (product_definition_relationship);
  LOCAL
    cyclic_relationship: LOGICAL := FALSE;
  END_LOCAL;

  REPEAT i := 1 TO HIINDEX(product_definition_relationship)
                                                WHILE NOT
cyclic_relationship;
    cyclic_relationship:=

product_definition_relationship[i].related_product_definition
                    :=:
product_definition_relationship[i].relating_product_definition;
  END_REPEAT;

  WHERE
    wr1: NOT cyclic_relationship;
END_RULE;

RULE product_definition_relationship_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition_relationship := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := ['space arrangement relationship'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
   t2_set := QUERY ( j <* t2_set |
  j.role.name = 'globally unambiguous identifier');
     violation := NOT (SIZEOF(T2_SET) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_definition_shape_for_deck_zone_design
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition_shape := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
```



```
  END_LOCAL;

  (* get all classification_assignment instances with id 'deck zone design
definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'deck zone design definition');

  (* get all instances of T1 that have class id 'deck zone design
definition' *)
  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  (* for all instances of product_definition_shape in t1_set:
     get the property_definition_representation instances that are
referencing a product_definition_shape instance via definition, filter out
those property_definition_representation instances whose attribute name has
the value ‘deck zone design parameters’; check if their number equals 1
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
   t2_inst.name = 'deck zone design parameters')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_definition_shape_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
 c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
 t1_set: SET OF product_definition_shape := [];
 t2_set: SET OF applied_identification_assignment := [];
 arg_list: LIST OF STRING := ['design definition'];
 violation: LOGICAL := FALSE;
END_LOCAL;

REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
        i.assigned_class.NAME = arg_LIST[j]);
END_REPEAT;

REPEAT i := 1 TO HIINDEX(c_a_set);
 REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
  t1_set := t1_set + c_a_set[i].items[j];
 END_REPEAT;
END_REPEAT;

REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
 t2_set := bag_to_set(USEDIN(t1_set[i],
    'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
 t2_set := QUERY ( j <* t2_set |
          j.role.name = 'globally unambiguous identifier');
 violation := NOT (SIZEOF(T2_SET) = 1);
END_REPEAT;
WHERE
wr1: NOT violation;
END_RULE;

RULE product_definition_with_date_freeboard_assigned
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
```

```
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_date_and_time_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
               i.assigned_class.name = 'freeboard characteristics');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
   t2_inst.role.name = 'date freeboard assigned')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_definition_with_freeboard_assigned_by
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_organization_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
              i.assigned_class.name = 'freeboard characteristics');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
    'SHIP_ARRANGEMENT_SCHEMA.APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
      t2_inst.role.name = 'freeboard assigned by')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_definition_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := [ 'definition',
                                   'definable object'];
```

```
    violation: LOGICAL := FALSE;
  END_LOCAL;

    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
      t2_set := QUERY ( j <* t2_set |
          j.role.name = 'globally unambiguous identifier');
violation := NOT (SIZEOF(T2_SET) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_related_product_category_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_related_product_category := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := [ 'shiptype'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
      t2_set := QUERY ( j <* t2_set |
              j.role.name = 'globally unambiguous identifier');
violation := NOT (SIZEOF(T2_SET) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE product_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := ['ship'];
```

```
    violation: LOGICAL := FALSE;
  END_LOCAL;

    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
      t2_set := QUERY ( j <* t2_set |
                j.role.name = 'globally unambiguous identifier');
violation := NOT (SIZEOF(T2_SET) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE property_definition_for_class_bulk_load_requirement_definition
FOR (applied_classification_assignment );
   LOCAL
      c_a_set : SET OF applied_classification_assignment := [];
      t1_set : SET OF property_definition := [];
      t2_set : SET OF property_definition_representation := [];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      c_a_set := QUERY (i <* applied_classification_assignment|
i.assigned_class.NAME = 'class bulk load requirement definition');
      REPEAT i := 1 TO HIINDEX(c_a_set);
         REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
            t1_set := t1_set + c_a_set[i].items[j];
         END_REPEAT;
      END_REPEAT;
      REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
       t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
         violation := NOT (SIZEOF(QUERY (t2_inst <* t2_set |
(t2_inst.name = 'class bulk load requirement definition parameters'))) =
1);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE property_definition_for_class_compartment_requirement_definition
FOR (applied_classification_assignment );
   LOCAL
      c_a_set : SET OF applied_classification_assignment := [];
      t1_set : SET OF property_definition := [];
      t2_set : SET OF property_definition_representation := [];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      c_a_set := QUERY (i <* applied_classification_assignment|
i.assigned_class.NAME = 'class compartment requirement definition');
```



```
      REPEAT i := 1 TO HIINDEX(c_a_set);
         REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
            t1_set := t1_set + c_a_set[i].items[j];
         END_REPEAT;
      END_REPEAT;
      REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
         violation := NOT (SIZEOF(QUERY (t2_inst <* t2_set |
(t2_inst.name = 'class compartment requirement definition parameters'))) =
1);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE property_definition_for_class_deck_load_requirement_definition
FOR (applied_classification_assignment );
   LOCAL
      c_a_set : SET OF applied_classification_assignment := [];
      t1_set : SET OF property_definition := [];
      t2_set : SET OF property_definition_representation := [];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      c_a_set := QUERY (i <* applied_classification_assignment|
i.assigned_class.NAME = 'class deck load requirement definition');
      REPEAT i := 1 TO HIINDEX(c_a_set);
         REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
            t1_set := t1_set + c_a_set[i].items[j];
         END_REPEAT;
      END_REPEAT;
      REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
       t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
         violation := NOT (SIZEOF(QUERY (t2_inst <* t2_set |
(t2_inst.name = 'class deck load requirement definition parameters'))) =
1);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE property_definition_for_class_notation
FOR (applied_classification_assignment );
   LOCAL
      c_a_set : SET OF applied_classification_assignment := [];
      t1_set : SET OF property_definition := [];
      t2_set : SET OF property_definition_representation := [];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      c_a_set := QUERY (i <* applied_classification_assignment|
i.assigned_class.NAME = 'class notation');
      REPEAT i := 1 TO HIINDEX(c_a_set);
         REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
            t1_set := t1_set + c_a_set[i].items[j];
         END_REPEAT;
      END_REPEAT;
      REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
        t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
         violation := NOT (SIZEOF(QUERY (t2_inst <* t2_set |
 (t2_inst.name = 'class notation'))) = 1);
      END_REPEAT;
   WHERE
```

```
      WR1:
         NOT violation;
END_RULE;

RULE property_definition_for_class_society
FOR (applied_classification_assignment );
   LOCAL
      c_a_set : SET OF applied_classification_assignment := [];
      t1_set : SET OF property_definition := [];
      t2_set : SET OF applied_organization_assignment := [];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      c_a_set := QUERY (i <* applied_classification_assignment|
i.assigned_class.NAME = 'class notation');
      REPEAT i := 1 TO HIINDEX(c_a_set);
         REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
            t1_set := t1_set + c_a_set[i].items[j];
         END_REPEAT;
      END_REPEAT;
      REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
        t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_ORGANIZATION_ASSIGNMENT.ITEMS'));
         violation := NOT (SIZEOF(QUERY (t2_inst <* t2_set |
(t2_inst.role.name = 'class society'))) = 1);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE property_definition_for_class_tank_requirement_definition
FOR (applied_classification_assignment );
   LOCAL
      c_a_set : SET OF applied_classification_assignment := [];
      t1_set : SET OF property_definition := [];
      t2_set : SET OF property_definition_representation := [];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      c_a_set := QUERY (i <* applied_classification_assignment|
i.assigned_class.NAME = 'class tank requirement definition');
      REPEAT i := 1 TO HIINDEX(c_a_set);
         REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
            t1_set := t1_set + c_a_set[i].items[j];
         END_REPEAT;
      END_REPEAT;
      REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
        t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
         violation := NOT (SIZEOF(QUERY (t2_inst <* t2_set |
 (t2_inst.name = 'class tank requirement definition parameters'))) = 1);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE property_definition_for_compartment_design_requirement
FOR (applied_classification_assignment);
   LOCAL
      c_a_set : SET OF applied_classification_assignment := [];
      t1_set : SET OF property_definition := [];
      t2_set : SET OF property_definition_representation := [];
      violation : LOGICAL := FALSE;
   END_LOCAL;
```



```
      c_a_set := QUERY (i <* applied_classification_assignment|
i.assigned_class.NAME = 'compartment design requirement');
      REPEAT i := 1 TO HIINDEX(c_a_set);
         REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
            t1_set := t1_set + c_a_set[i].items[j];
         END_REPEAT;
      END_REPEAT;
      REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
        t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
         violation := NOT (SIZEOF(QUERY (t2_inst <* t2_set |
 (t2_inst.name = 'compartment design requirement parameters'))) = 1);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE property_definition_for_compartment_function
FOR (applied_classification_assignment );
   LOCAL
      c_a_set : SET OF applied_classification_assignment := [];
      t1_set : SET OF property_definition := [];
      t2_set : SET OF property_definition_representation := [];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      c_a_set := QUERY (i <* applied_classification_assignment|
i.assigned_class.NAME = 'compartment functional definition');
      REPEAT i := 1 TO HIINDEX(c_a_set);
         REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
            t1_set := t1_set + c_a_set[i].items[j];
         END_REPEAT;
      END_REPEAT;
      REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
        t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
         violation := NOT (SIZEOF(QUERY (t2_inst <* t2_set |
(t2_inst.name = 'compartment function parameters'))) = 1);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE property_definition_for_compensated_gross_tonnage
FOR (applied_classification_assignment );
   LOCAL
      c_a_set : SET OF applied_classification_assignment := [];
      t1_set : SET OF property_definition := [];
      t2_set : SET OF property_definition_representation := [];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      c_a_set := QUERY (i <* applied_classification_assignment|
i.assigned_class.NAME = 'tonnage definition');
      REPEAT i := 1 TO HIINDEX(c_a_set);
         REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
            t1_set := t1_set + c_a_set[i].items[j];
         END_REPEAT;
      END_REPEAT;
      REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
       t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
         violation := NOT (SIZEOF(QUERY (t2_inst <* t2_set |
('compensated gross tonnage' IN WHICH_CLASS(t2_inst)))) = 1);
      END_REPEAT;
   WHERE
```

```
      WR1:
         NOT violation;
END_RULE;

RULE property_definition_for_damage_stability_definition_requires_reference
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF representation := [];
    t3_set:  SET OF property_definition_representation := [];
    t4_set:  SET OF property_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'damage stability definition');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'stability table');

  REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
      t2_set := t2_set + c2_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + bag_to_set(USEDIN(t2_set[i],
           'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'));
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;

  violation := t1_set <> t4_set;

  WHERE
    wr1: NOT violation;

END_RULE;

RULE property_definition_for_date_of_loading
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF applied_date_and_time_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
         i.ASSIGNED_CLASS.NAME = 'loading condition operating definition');

  REPEAT i := 1 TO HIINDEX(c_a_set);
```



```
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.role.name = 'date of loading')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE property_definition_for_deck_zone_function
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
         i.assigned_class.name = 'deck zone functional definition');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.name = 'deck zone function parameters')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE property_definition_for_gross_tonnage
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
              i.assigned_class.name = 'tonnage definition');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
```

```
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 'gross tonnage' IN WHICH_CLASS(t2_inst))) = 1);
END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE property_definition_for_local_coordinate_system
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'local co ordinate system');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
   t2_inst.name = 'local coordinate system')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE property_definition_for_local_coordinate_system_with_position
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME =
                  'local co ordinate system with position reference');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],

'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name =
      'local coordinate system with position reference')) = 1);
  END_REPEAT;
  WHERE
```

```
    WR1: NOT violation;
END_RULE;

RULE property_definition_for_net_tonnage
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'tonnage definition');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
   'net tonnage' IN WHICH_CLASS(t2_inst))) = 1);
END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE property_definition_for_stability_definition_requires_reference
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF representation := [];
    t3_set:  SET OF property_definition_representation := [];
    t4_set:  SET OF property_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'stability definition');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'stability table');

  REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
      t2_set := t2_set + c2_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + bag_to_set(USEDIN(t2_set[i],
            'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'));
```

```
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;

  violation := t1_set <> t4_set;

  WHERE
    wr1: NOT violation;

END_RULE;

RULE property_definition_for_tonnage_definition
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'tonnage definition');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.name = 'tonnage definition parameters')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE property_definition_for_zone_function
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'zone functional definition');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
```

```
 t2_inst.name = 'zone function parameters')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE property_definition_has_references_with_name_loadline
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'loadline');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
 t2_inst.name = 'loadline')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE property_definition_representation_for_date_of_measurement
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition_representation := [];
    t2_set:  SET OF applied_date_and_time_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'tonnage measurement');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
'date of  measurement')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE property_definition_representation_for_gross_tonnage
```

```
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition_representation := [];
    t2_set:  SET OF applied_date_and_time_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'gross tonnage');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
                'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
'date of measurement')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE property_definition_representation_for_net_tonnage
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition_representation := [];
    t2_set:  SET OF applied_date_and_time_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'net tonnage');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
'date of  measurement')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE property_definition_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF applied_identification_assignment := [];
```



```
    arg_list:  LIST OF STRING :=
['cargo bay definition', 'compartment functional definition', 'deck zone
functional definition', 'design requirement', 'loading condition
definition', 'local co ordinate system', 'spacing table', 'stability
definition', 'tonnage definition', 'zone functional definition'];
        violation: LOGICAL := FALSE;
  END_LOCAL;


    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
      t2_set := QUERY ( j <* t2_set |
              j.role.name = 'globally unambiguous identifier');
violation := NOT (SIZEOF(T2_SET) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE property_definition_with_lightship_weight_item
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'lightship weight item');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.' + 'DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name =
'lightship weight item')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE property_definition_with_weight_and_centre_of_gravity
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
```

```
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'weight and centre of gravity');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name =
'weight and centre of gravity')) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE representation_for_absolute_cargo FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘position', 'orientation’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'absolute cargo position parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_adjacent_space_surface_area FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘surface area’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'adjacent space surface area parameters'))) >
0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
```



```
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_bulk_cargo FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘natural angle of repose', 'pollution
code', 'required carriage temperature', 'type of', 'un type code’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'bulk cargo parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_bulk_cargo_assignment FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘actual angle of repose', 'assignment
context', 'cargo height', 'cargo identifier', 'trimmed’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'bulk cargo assignment parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_capacity_properties FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘capacity level origin', 'capacity
centre', 'capacity level', 'capacity trim angle', 'capacity heel angle',
'capacity volume', 'capacity context’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'capacity properties'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
```

```
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_cargo_compartment_property FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'design stowage density’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'cargo compartment property'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_cargo_footprint FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘contact material', 'shape',
'transferred mass’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'cargo footprint'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_class_and_statutory_designation
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['class number'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  reps := QUERY(
```



```
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'class and statutory designation')
                  )
               ) > 0
            );

  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                     rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE representation_for_class_bulk_load_requirement_definition FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘ambient temperature', 'angle of
repose', 'bulk cargo mass', 'cargo density', 'cargo height', 'coating',
'damage waterline', 'max pressure', 'max temperature', 'min pressure', 'min
temperature', 'permeability', 'top of hatch’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'class bulk load requirement definition parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_class_compartment_requirement_definition FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘ambient temperature', 'cargo density',
'cargo height', 'coating', 'damage waterline', 'max pressure', 'max
temperature', 'min pressure', 'min temperature’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'class compartment requirement definition parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
```

```
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_class_notation FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘service area', 'approval required for
oil cargo', 'approval required for loading unloading aground', 'approval
required for unloading grabs’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'class notation'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_class_parameters FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘length class', 'length solas',
'scantlings draught', 'block coefficient class', 'design speed ahead',
'design speed astern’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'class parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_class_tank_requirement_definition FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘ambient temperature', 'cargo density',
'cargo height', 'coating', 'damage waterline', 'max pressure', 'max
```



```
temperature', 'min pressure', 'min temperature', 'overflow height',
'partial filling', 'pressure relief setting’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'class tank requirement definition parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_coating FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘dry film thickness', 'number of
coats', 'surface preparation’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'coating parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_coating_level FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘lower extent', 'upper extent’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'coating level'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;
```

```
RULE representation_for_compartment_abbreviated_name FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'name’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment abbreviated name'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_acceleration FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'acceleration g force’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment acceleration'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_access_authorization FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'authorization
classification’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment access authorization'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
```



```
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_air_circulation_rate FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'air circulation rate’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment air circulation rate'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_cargo_assignment FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING :=
     [‘assignment context', 'cargo identifier’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'compartment cargo assignment parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_coating FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment coating'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
```

```
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_design_requirement FOR (representation
);
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘description', 'requirement type’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'compartment design requirement parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_function FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘used for’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment function parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_group FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘tonnage volume’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment group parameters'))) > 0);
```

```
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_horizontal_cross_sectional_area FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'horizontal cross sectional
area’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'compartment horizontal cross sectional area property'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_illumination FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'illumination value’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment illumination'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_insulation FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'insulation category’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
```

```
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment insulation'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_noise_category FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'noise category’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment noise category'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_nuclear_classification FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'nuclear classification’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment nuclear classification'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_occupancy FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'occupancy’];
      violation : LOGICAL := FALSE;
```



```
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment occupancy'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_safety_class FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'safety category’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment safety class'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_security_classification FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'security classification’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment security classification'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_stiffened_surface_area_property FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
```

```
      arg_list : LIST OF STRING := [‘context', 'stiffened surface area’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'compartment stiffened surface area property'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_tightness FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'required bulkhead
tightness’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment tightness'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_unstiffened_surface_area_property FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'unstiffened surface area’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'compartment unstiffened surface area property'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
```

```
         NOT violation;
END_RULE;

RULE representation_for_compartment_vertical_longitudinal_sectional_area
FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context',
 'vertical  longitudinal cross sectional area’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'compartment vertical longitudinal sectional area property'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_vertical_transverse_sectional_area FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'vertical transverse cross
  sectional area’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'compartment vertical transverse sectional area property'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_volume_permeability_property FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'permeability’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'compartment volume permeability property'))) > 0);
```

```
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_volume_property FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘centre of volume', 'context',
'volume’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment volume property'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compartment_ziplist_number FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'department ziplist number',
'division ziplist number’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment ziplist number'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_compensated_gross_tonnage FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘compensation factor', 'tonnage
value’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
```

```
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compensated gross tonnage'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_corrosion_control_coating FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘applicability', 'dry film thickness',
'number of coats', 'surface preparation', 'type of’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'corrosion control coating parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_corrosion_protection FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘cathodic protection’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'corrosion protection'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_damage_case FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘damage cause', 'relative damage
position’];
```

```
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'damage case parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_damage_position FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘centre of damage', 'position
accuracy’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'damage position parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_dangerous_goods_code FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘class', 'subsidiary risks’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'dangerous goods code parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_deck_cargo_assignment FOR (representation );
```

```
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘assignment context', 'cargo
identifier’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'deck cargo assignment parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_deck_zone_function FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘used for’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'deck zone function parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_dry_cargo FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘pollution code', 'un type code’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'dry cargo parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;
```

```
RULE representation_for_fire_safe_coating FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘dry film thickness', 'low flame
spread', 'nitro cellulose based', 'number of coats', 'surface
preparation’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'fire safe coating parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_freeboard_characteristics FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘assigned code', 'freeboard’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'freeboard characteristics'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_gaseous_cargo FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘cargo type', 'carried in liquid
state', 'pollution code', 'un type code’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'gaseous cargo parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
```



```
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_gaseous_cargo_assignment FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘assignment context', 'cargo
identifier’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'gaseous cargo assignment parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_global_axis_placement
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['global axes and origin', 'after
perpendicular offset', 'orientation'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'global axis placement')
                   )
             ) > 0
          );

  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE representation_for_gross_tonnage FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘overdeck tonnage', 'tonnage value',
'underdeck tonnage’];
      violation : LOGICAL := FALSE;
```

```
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'gross tonnage'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_lightship_definition FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘lightship weight', 'lightship centre
of gravity’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'lightship definition parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_lightship_weight_item FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘aft weight extent', 'fwd weight
extent’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'lightship weight item'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_liquid_cargo FOR (representation );
```



```
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘cargo type', 'pollution code', 'un
type code’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'liquid cargo parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_liquid_cargo_assignment FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘assignment context', 'cargo height',
'cargo identifier’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'liquid cargo assignment parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_loading_condition_design_definition FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘type of’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'loading condition design definition parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
```

```
         NOT violation;
END_RULE;

RULE representation_for_loading_condition_operating_definition FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘type of’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'loading condition operating definition parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_loadline FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘load line length', 'load line depth',
'load line displacement', 'load line block coefficient', 'load line
regulation’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'loadline'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_local_coordinate_system
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['local axes and origin'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
```

```
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'local coordinate system')
                    )
               ) > 0
          );

  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                    rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE representation_for_moment_3d_restricts_representation_item FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘longitudinal moment', 'transverse
moment', 'vertical moment', 'origin’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (i <* representation| i.NAME = 'moment 3d');
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_moments_of_inertia FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘long moment of inertia', 'trans moment
of inertia’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'moments of inertia'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_net_tonnage FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘tonnage value’];
      violation : LOGICAL := FALSE;
```

```
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'net tonnage'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_person_group FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘number of people', 'person type',
'volume’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'person group parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_primer_coating FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘dry film thickness', 'number of
coats', 'surface preparation’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'primer coating parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_principal_characteristics FOR (representation );
```



```
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘length between perpendiculars',
'moulded breadth', 'moulded depth’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'principal characteristics'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_space_adjacency_relationship FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘adjacency access', 'adjacency type’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'space adjacency relationship parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_space_positional_relationship FOR (representation
);
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘relationship type’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'space positional relationship parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
```

```
         NOT violation;
END_RULE;

RULE representation_for_stability_table_restricted
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c2_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF representation := [];
    t2_set:  SET OF representation_item := [];
    t3_set:  SET OF representation_item := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.Assigned_class.name = 'stability table');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.Assigned_class.name = 'stability properties for one
floating position');

  REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
      t2_set := t2_set + c2_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].items);
      t3_set := t3_set + t1_set[i].items[j];
    END_REPEAT;
  violation := (SIZEOF(t3_set* t2_set) < 1);
  t3_set:= [];
  END_REPEAT;

WHERE
    wr1: NOT violation;
  END_RULE;

RULE representation_for_stability_table_restricted_by_class_id
FOR (applied_classification_assignment);
LOCAL
  c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
  t1_set: SET OF REPRESENTATION := [];
  arg_list: LIST OF STRING := ['mean shell thickness'];
  violation: LOGICAL := FALSE;
END_LOCAL;


c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
  i.assigned_class.NAME = 'stability table');


REPEAT i := 1 TO HIINDEX(c_a_set);
  REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
    t1_set := t1_set + c_a_set[i].items[j];
  END_REPEAT;
END_REPEAT;
```



```
REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
  REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
    violation:= (SIZEOF(QUERY(rep_item<* t1_set[i].items |
    rep_item.name = arg_list[j])) <> 1);
  END_REPEAT;
END_REPEAT;
WHERE
  WR1: NOT violation;
END_RULE;

RULE representation_for_tank_compartment_property FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘context', 'design stowage density’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'tank compartment property'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_tonnage_definition FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘tonnage regulation’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'tonnage definition'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_tonnage_measurement FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘tonnage value’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'tonnage measurement'))) > 0);
```

```
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_unit_cargo FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘cargo type', 'pollution code', 'un
type code’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'unit cargo parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_unit_cargo_assignment FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘assignment context', 'cargo
identifier’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'unit cargo assignment parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_unit_cargo_bounding_box FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘point max', 'point min’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
```



```
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'unit cargo bounding box'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_unit_cargo_group FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := ['volume'];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'unit cargo group parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_vehicle_load_description FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘load handling', 'load per wheel',
'number of wheels', 'type of vehicle’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'vehicle load description'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_for_zone_function FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘used for’];
      violation : LOGICAL := FALSE;
```

```
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'zone function parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE representation_has_global_uncertainty_assigned_context
FOR (SHAPE_REPRESENTATION);
LOCAL
 has_gunac: LOGICAL := TRUE;
END_LOCAL;

REPEAT i := 1 TO HIINDEX(SHAPE_REPRESENTATION) WHILE has_gunac;
 has_gunac :=
('SHIP_ARRANGEMENT_SCHEMA.GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT' IN
               TYPEOF(SHAPE_REPRESENTATION[i].CONTEXT_OF_ITEMS));
END_REPEAT;

WHERE
 WR1: has_gunac;
END_RULE;

RULE representation_has_global_unit_assigned_context
FOR (representation );
   LOCAL
      has_guac : LOGICAL := TRUE;
   END_LOCAL;
      REPEAT i := 1 TO HIINDEX(representation) WHILE has_guac;
         REPEAT j := 1 TO SIZEOF(representation[i].items) WHILE has_guac;
            IF ('SHIP_ARRANGEMENT_SCHEMA.VALUE_REPRESENTATION_ITEM' IN
TYPEOF(representation[i].items[j])) OR
('SHIP_ARRANGEMENT_SCHEMA.GEOMETRIC_REPRESENTATION_ITEM' IN
TYPEOF(representation[i].items[j])) THEN
               has_guac :=
'SHIP_ARRANGEMENT_SCHEMA.GLOBAL_UNIT_ASSIGNED_CONTEXT' IN
TYPEOF(representation[i].CONTEXT_OF_ITEMS);
            END_IF;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         has_guac;
END_RULE;

RULE representation_item_for_transformation_to_parent
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF property_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    t3_set:  SET OF representation := [];
    t4_set:  SET OF representation_map := [];
    t5_set:  SET OF mapped_item := [];
```



```
    arg_list:  LIST OF STRING :=
['local coordinate system position in global coordinate system',
'local coordinate system position in parent local coordinate system',
'local coordinate system position in parent local coordinate system with
position reference'];
    violation1: LOGICAL := FALSE;
    violation2: LOGICAL := FALSE;

  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
            i.assigned_class.NAME =  'local co ordinate system');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation1;
    t2_set := bag_to_set(USEDIN(t1_set[i],

'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.DEFINITION'));
    violation1 := NOT (SIZEOF(QUERY(t2_inst <* t2_set |
t2_inst.used_representation.name =
      'local axis representation')) = 1);
    t3_set := t3_set + t2_set[i].used_representation;
  END_REPEAT;

   REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation1;
    t4_set := bag_to_set(USEDIN(t3_set[i],
      'SHIP_ARRANGEMENT_SCHEMA.REPRESENTATION_MAP.MAPPED_REPRESENTATION'));
  END_REPEAT;

   REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation1;
    t5_set := bag_to_set(USEDIN(t4_set[i],
      'SHIP_ARRANGEMENT_SCHEMA.MAPPED_ITEM.MAPPING_SOURCE'));
      REPEAT j := 1 TO 3 WHILE NOT violation2;
      violation2 := NOT (SIZEOF(QUERY(t2_inst <* t5_set | t2_inst.name =
      ARG_LIST[j])) = 1);
      END_REPEAT;
  END_REPEAT;

  WHERE
    WR1: NOT violation1;
    WR2: NOT violation2;
END_RULE;

RULE representation_items_optional_for_bulk_cargo FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘flash point', 'required carriage
temperature', 'permeability', 'stowage factor', 'user def cargo’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'bulk cargo parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
```

```
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_capacity_properties FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘user defined capacity context’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'capacity properties'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE
representation_items_optional_for_class_deck_load_requirement_definition
FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘grab weight', 'stowage height',
'stowage rate', 'vehicle load’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'class deck load requirement definition parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_class_notation
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['ice class notation','service factor',
'approval required for heavy cargo'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  reps := QUERY(
```



```
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name = 'class notation'))) > 0
          );

  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT found);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT found);
      found := (SIZEOF(QUERY(rep_item <* reps[i].items |
                rep_item.name=arg_list[j])) > 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT found;
END_RULE;

RULE representation_items_optional_for_compartment_access_authorization FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘user defined value’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment access authorization'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_compartment_design_requirement FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘user defined value’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'compartment design requirement parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;
```

```
RULE representation_items_optional_for_compartment_function FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘user def function’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment function parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_compartment_insulation FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘user defined value’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment insulation'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_compartment_noise_category FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘user defined value’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment noise category'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
```



```
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_compartment_safety_class FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘user defined value’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment safety class'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_compartment_security FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘user defined value’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment security classification'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_compartment_tightness FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘user defined value’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'compartment tightness'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
```

```
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_corrosion_control_coating FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘user defined type’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'corrosion control coating parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_damage_case FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘user defined’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'damage case parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_deck_zone_function FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘user def function’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
```



```
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'deck zone function parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_detailed_cargo_material_properties
FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘density', 'expansion coefficient',
'specific heat capacity', 'thermal conductivity', 'viscosity’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'detailed cargo material properties parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_dry_cargo FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘flash point', 'required carriage
temperature', 'user def cargo', 'permeability', 'stowage factor’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'dry cargo parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_gaseous_cargo FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
```

```
      arg_list : LIST OF STRING := [‘flash point', 'required carriage
pressure', 'required carriage temperature', 'user def cargo’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'gaseous cargo parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_general_cargo_material_properties
FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘density’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'general cargo material properties parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_liquid_cargo FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘flash point', 'required carriage
pressure', 'required carriage temperature', 'user def cargo’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'liquid cargo parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
```



```
         NOT found;
END_RULE;

RULE
representation_items_optional_for_loading_condition_operating_definition
FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘place of loading’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'loading condition operating definition parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_principal_characteristics
FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘block coefficient', 'design draught',
'design deadweight', 'min draught at fp', 'max draught at fp',
'min draught at ap', 'max draught at ap’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'principal characteristics'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_space_connection_relationship FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘connecting system’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME =
 'space connection relationship parameters'))) > 0);
```

```
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_tank_geometric_parameters FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘breadth wash', 'length wash’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'tank geometric parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_tank_piping_design_properties FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘airpipe height', 'filling height',
'relief valve pressure setting', 'sounding pipe height’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'tank piping design properties'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_unit_cargo FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘flash point', 'permeability',
'stowage factor', 'required carriage temperature', 'stack limit',
'user def cargo', 'volume’];
```



```
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'unit cargo parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_vehicle_load_description FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘max tyre pressure', 'print area’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'vehicle load description'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_items_optional_for_zone_function FOR (representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘user def function’];
      found : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'zone function parameters'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT found;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT found;
            found := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) > 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT found;
END_RULE;

RULE representation_local_coordinate_system_with_position_reference
FOR (representation);
  LOCAL
```

```
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['local axes and origin'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION')) |
 (temp_prop_def_rep.name =
      'local coordinate system with position reference')
                   )
             ) > 0
          );

  REPEAT i:=1 TO HIINDEX(reps) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
      violation := (SIZEOF(QUERY(rep_item <* reps[i].items |
                   rep_item.name = arg_list[j])) <> 1);
    END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE representation_restricted_weight_and_centre_of_gravity FOR
(representation );
   LOCAL
      reps : BAG OF representation := [];
      arg_list : LIST OF STRING := [‘mass ', 'centre of gravity’];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      reps := QUERY (temp_rep <* representation| SIZEOF(QUERY
(temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_ARRANGEMENT_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
'USED_REPRESENTATION'))|
 (temp_prop_def_rep.NAME = 'weight and centre of gravity'))) > 0);
      REPEAT i := 1 TO HIINDEX(reps) WHILE NOT violation;
         REPEAT j := 1 TO HIINDEX(arg_list) WHILE NOT violation;
            violation := SIZEOF(QUERY (rep_item <* reps[i].items |
(rep_item.NAME = arg_list[j]))) <> 1;
         END_REPEAT;
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE revision_has_mandatory_attribute_description
FOR (group);
LOCAL
    t1_set:  SET OF group := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

t1_set := QUERY(i <* group | VALUE_IN(WHICH_CLASS(i), 'revision'));

violate := (SIZEOF(QUERY(k <* t1_set |
NOT EXISTS (k.description))) > 0);
```



```
WHERE
    wr1: NOT violate;
END_RULE;

RULE revision_with_context_referenced_for_context_of_revision
FOR (applied_group_assignment, group );
   LOCAL
      t1_set : SET OF group := [];
      a_set : SET OF applied_group_assignment := [];
      violate : LOGICAL := FALSE;
   END_LOCAL;
      t1_set := QUERY (a <* group|
 VALUE_IN(WHICH_CLASS(a), 'revision with context'));
      REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
         a_set := QUERY (b <* applied_group_assignment|
       (b.ASSIGNED_GROUP = t1_set[i]) AND
       (b.role.NAME = 'context of revision'));
         violate := SIZEOF(a_set) <> 1;
      END_REPEAT;
   WHERE
      WR1:
         NOT violate;
END_RULE;

RULE shape_representation_subtype_exclusiveness
FOR (shape_representation );
   WHERE
      WR1:
         SIZEOF(QUERY (sr <* shape_representation| NOT (SIZEOF(TYPEOF(sr) *
[‘SHIP_ARRANGEMENT_SCHEMA.NON_MANIFOLD_SURFACE_SHAPE_REPRESENTATION’]) <=
2))) = 0;
END_RULE;

RULE ship_designation_has_one_specified_names
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF product_definition := [];
    t2_set:  SET OF applied_identification_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.name = 'ship designation');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i], 'SHIP_ARRANGEMENT_SCHEMA.' +
'APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | (t2_inst.role.name =
'imo number')OR (t2_inst.role.name = 'pennant hull number') )) = 1);
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE spacing_position_compound_representation_has_name
FOR (applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
```

```
    t1_set:  SET OF Compound_representation_item := [];
    t2_set:  SET Of representation_item := [];
    arg_list:  LIST OF STRING := ['position number', 'position'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME = 'spacing position');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set |
           items.name = arg_list[j])) <> 1);
 END_REPEAT;
  END_REPEAT;
  WHERE
    WR1: NOT violation;
END_RULE;

RULE spacing_position_with_offset_compound_representation_has_class
FOR (applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t3_set:  SET Of representation_item := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME = 'spacing position with offset');
REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;
  c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME = 'spacing position');

  REPEAT i := 1 TO HIINDEX(c_a_set2);
    REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
      t2_set := t2_set + c_a_set2[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
      t3_set := t3_set + t1_set[i].item_element;
    END_REPEAT;
    violation := (SIZEOF(t3_set * t2_set) <> 1);
    t3_set:= [];
  END_REPEAT;
  WHERE
    WR1: NOT violation;
END_RULE;
```



```
RULE spacing_position_with_offset_compound_representation_has_name
FOR (applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF Compound_representation_item := [];
    t2_set:  SET Of representation_item := [];
    arg_list:  LIST OF STRING := ['offset'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.assigned_class.NAME =
                  'spacing position with offset');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
        violation := (SIZEOF(QUERY(items <* t2_set |
        items.name = arg_list[j])) <> 1);
 END_REPEAT;
  END_REPEAT;
  WHERE
    WR1: NOT violation;
END_RULE;

RULE stability_properties_for_floating_position_has_class
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    c_a_set2 : SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t3_set:  SET OF REPRESENTATION_ITEM := [];
    l_rep_item : list_representation_item;
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.Assigned_class.name = 'stability properties for one
floating position');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.Assigned_class.name = 'stability property');

  REPEAT i := 1 TO HIINDEX(c_a_set2);
    REPEAT j := 1 TO HIINDEX(c_a_set2[i].items);
      t2_set := t2_set + c_a_set2[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
          l_rep_item := t1_set[i].item_element;
      t3_set := t3_set + l_rep_item[j];
```

```
    END_REPEAT;
    violation := (SIZEOF(t3_set * t2_set) < 1);
    t3_set:= [];
  END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE stability_properties_for_floating_position_has_name
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF Compound_representation_item := [];
    t2_set:  SET OF representation_item := [];
    arg_list:  LIST OF STRING := ['centre of gravity above keel',
'definition of starting floating position'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.Assigned_class.name =
 'stability properties for one floating position');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
  violation := (SIZEOF(QUERY(items <* t2_set | items.name = arg_list[j]))
<> 1);
 END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;

END_RULE;

RULE stability_property_has_name
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF Compound_representation_item := [];
    t2_set:  SET OF representation_item := [];
    arg_list:  LIST OF STRING := ['angle of heel', 'righting arm',
 'centre of buoyancy'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.Assigned_class.name = 'stability property');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
```



```
    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        t2_set := t1_set[i].item_element;
   violation := (SIZEOF(QUERY(items <* t2_set |
 items.name = arg_list[j])) <> 1);
 END_REPEAT;
  END_REPEAT;

  WHERE
    wr1: NOT violation;

END_RULE;

RULE tonnage_definition_has_properties
FOR (property_definition_representation,
applied_classification_assignment);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  LIST OF product_definition := [];
    t2_set:  SET OF property_definition_representation := [];
    t3_set:  LIST OF property_definition := [];
    t4_set:  LIST OF product_definition := [];

    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.name = 'tonnage definition');

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;

  t2_set:= QUERY(i <* PROPERTY_DEFINITION_REPRESENTATION |
                   i.NAME = 'tonnage definition');

  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + t2_set[i].definition;
  END_REPEAT;

  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;

  violation := t1_set <> t4_set;

  WHERE
    wr1: NOT violation;

END_RULE;

RULE unique_approvals_in_approval_history
FOR (GROUP, APPLIED_GROUP_ASSIGNMENT);
LOCAL
    t1_set:  SET OF GROUP := [];
    t2_set:  SET OF APPLIED_GROUP_ASSIGNMENT := [];
    t3_set:  SET OF approval :=[];
    violate: LOGICAL := FALSE;
END_LOCAL;
  t1_set := QUERY(i <* GROUP | VALUE_IN(WHICH_CLASS(i),
    'approval history'));
REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
   t2_set := QUERY(a <* APPLIED_GROUP_ASSIGNMENT |
     a.ASSIGNED_GROUP = t1_set[i]);
   t3_set := QUERY(b <* t2_set[1].items |
```

```
     'SHIP_ARRANGEMENT_SCHEMA.APPROVAL' IN TYPEOF(b));
    violate := NOT (VALUE_UNIQUE(t3_set));
END_REPEAT;
WHERE
    WR1: NOT violate;
END_RULE;

RULE user_def_cargo_description_required_for_cargo_type
FOR (representation );
   LOCAL
      violation : LOGICAL := FALSE;
   END_LOCAL;
      REPEAT i := 1 TO HIINDEX(representation) WHILE NOT violation;
         violation := (SIZEOF(QUERY (r <* representation[i].items |
(('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM' IN TYPEOF(r))
AND (r.NAME = 'cargo type')) AND
(r\descriptive_representation_item.description = 'user defined'))) > 0) AND
(SIZEOF(QUERY (r <* representation[i].items |
 (r.NAME = 'user def cargo'))) = 0);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE user_def_cargo_description_required_for_type_of FOR (representation );
   LOCAL
      violation : LOGICAL := FALSE;
   END_LOCAL;
      REPEAT i := 1 TO HIINDEX(representation) WHILE NOT violation;
         violation := (SIZEOF(QUERY (r <* representation[i].items |
(('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM' IN TYPEOF(r))
AND (r.NAME = 'type of')) AND
(r\descriptive_representation_item.description = 'user defined'))) > 0) AND
(SIZEOF(QUERY (r <* representation[i].items |
 (r.NAME = 'user def cargo'))) = 0);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE user_def_function_description_required FOR (representation );
   LOCAL
      violation : LOGICAL := FALSE;
   END_LOCAL;
      REPEAT i := 1 TO HIINDEX(representation) WHILE NOT violation;
         violation := (SIZEOF(QUERY (r <* representation[i].items |
(('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM' IN TYPEOF(r))
AND (r.NAME = 'used for')) AND
(r\descriptive_representation_item.description = 'user defined'))) > 0) AND
(SIZEOF(QUERY (r <* representation[i].items |
 (r.NAME = 'user def function'))) = 0);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE
user_defined_capacity_context_description_required_for_capacity_context FOR
(representation );
   LOCAL
      violation : LOGICAL := FALSE;
```



```
   END_LOCAL;
      REPEAT i := 1 TO HIINDEX(representation) WHILE NOT violation;
         violation := (SIZEOF(QUERY (r <* representation[i].items |
(('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM' IN TYPEOF(r))
AND (r.NAME = 'capacity context')) AND
(r\descriptive_representation_item.description = 'user defined'))) > 0) AND
(SIZEOF(QUERY (r <* representation[i].items |
 (r.NAME = 'user defined capacity context'))) = 0);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE user_defined_description_required_for_damage_cause FOR (representation
);
   LOCAL
      violation : LOGICAL := FALSE;
   END_LOCAL;
      REPEAT i := 1 TO HIINDEX(representation) WHILE NOT violation;
         violation := (SIZEOF(QUERY (r <* representation[i].items |
(('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM' IN TYPEOF(r))
AND (r.NAME = 'damage cause')) AND
(r\descriptive_representation_item.description = 'user defined'))) > 0) AND
(SIZEOF(QUERY (r <* representation[i].items |
 (r.NAME = 'user defined'))) = 0);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE user_defined_type_description_required_for_type_of FOR (representation
);
   LOCAL
      violation : LOGICAL := FALSE;
   END_LOCAL;
      REPEAT i := 1 TO HIINDEX(representation) WHILE NOT violation;
         violation := (SIZEOF(QUERY (r <* representation[i].items |
(('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM' IN TYPEOF(r))
AND (r.NAME = 'type of')) AND
(r\descriptive_representation_item.description = 'user defined'))) > 0) AND
(SIZEOF(QUERY (r <* representation[i].items |
 (r.NAME = 'user defined type'))) = 0);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE
user_defined_value_description_required_for_authorization_classification
FOR (representation );
   LOCAL
      violation : LOGICAL := FALSE;
   END_LOCAL;
      REPEAT i := 1 TO HIINDEX(representation) WHILE NOT violation;
         violation := (SIZEOF(QUERY (r <* representation[i].items |
(('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM' IN TYPEOF(r))
AND (r.NAME = 'authorization classification')) AND
(r\descriptive_representation_item.description = 'user defined'))) > 0) AND
(SIZEOF(QUERY (r <* representation[i].items |
 (r.NAME = 'user defined value'))) = 0);
      END_REPEAT;
   WHERE
      WR1:
```

```
         NOT violation;
END_RULE;

RULE user_defined_value_description_required_for_compartment_insulation FOR
(representation );
   LOCAL
      rep_set : SET OF representation := [];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      REPEAT i := 1 TO HIINDEX(representation) WHILE NOT violation;
         violation := (SIZEOF(QUERY (r <* representation[i].items |
(('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM' IN TYPEOF(r))
AND (r.NAME = 'compartment insulation')) AND
(r\descriptive_representation_item.description = 'user defined'))) > 0) AND
(SIZEOF(QUERY (r <* representation[i].items |
 (r.NAME = 'user defined value'))) = 0);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE user_defined_value_description_required_for_compartment_noise_category
FOR (representation );
   LOCAL
      rep_set : SET OF representation := [];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      REPEAT i := 1 TO HIINDEX(representation) WHILE NOT violation;
         violation := (SIZEOF(QUERY (r <* representation[i].items |
(('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM' IN TYPEOF(r))
AND (r.NAME = 'compartment noise category')) AND
(r\descriptive_representation_item.description = 'user defined'))) > 0) AND
(SIZEOF(QUERY (r <* representation[i].items |
 (r.NAME = 'user defined value'))) = 0);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE user_defined_value_description_required_for_compartment_safety_class
FOR (representation );
   LOCAL
      rep_set : SET OF representation := [];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      REPEAT i := 1 TO HIINDEX(representation) WHILE NOT violation;
         violation := (SIZEOF(QUERY (r <* representation[i].items |
(('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM' IN TYPEOF(r))
AND (r.NAME = 'compartment safety class')) AND
(r\descriptive_representation_item.description = 'user defined'))) > 0) AND
(SIZEOF(QUERY (r <* representation[i].items |
 (r.NAME = 'user defined value'))) = 0);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE user_defined_value_description_required_for_compartment_security FOR
(representation );
   LOCAL
      rep_set : SET OF representation := [];
```



```
      violation : LOGICAL := FALSE;
   END_LOCAL;
      REPEAT i := 1 TO HIINDEX(representation) WHILE NOT violation;
         violation := (SIZEOF(QUERY (r <* representation[i].items |
(('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM' IN TYPEOF(r))
AND (r.NAME = 'compartment security classification')) AND
(r\descriptive_representation_item.description = 'user defined'))) > 0) AND
(SIZEOF(QUERY (r <* representation[i].items |
 (r.NAME = 'user defined value'))) = 0);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE user_defined_value_description_required_for_compartment_tightness FOR
(representation );
   LOCAL
      rep_set : SET OF representation := [];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      REPEAT i := 1 TO HIINDEX(representation) WHILE NOT violation;
         violation := (SIZEOF(QUERY (r <* representation[i].items |
(('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM' IN TYPEOF(r))
AND (r.NAME = 'compartment tightness')) AND
(r\descriptive_representation_item.description = 'user defined'))) > 0) AND
(SIZEOF(QUERY (r <* representation[i].items |
 (r.NAME = 'user defined value'))) = 0);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE user_defined_value_description_required_for_requirement_type FOR
(representation );
   LOCAL
      rep_set : SET OF representation := [];
      violation : LOGICAL := FALSE;
   END_LOCAL;
      REPEAT i := 1 TO HIINDEX(representation) WHILE NOT violation;
         violation := (SIZEOF(QUERY (r <* representation[i].items |
(('SHIP_ARRANGEMENT_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM' IN TYPEOF(r))
AND (r.NAME = 'requirement type')) AND
(r\descriptive_representation_item.description = 'user defined'))) > 0) AND
(SIZEOF(QUERY (r <* representation[i].items |
 (r.NAME = 'user defined value'))) = 0);
      END_REPEAT;
   WHERE
      WR1:
         NOT violation;
END_RULE;

RULE version_creation_has_mandatory_attribute_description FOR (action );
   LOCAL
      t1_set : SET OF action := [];
      violate : LOGICAL := FALSE;
   END_LOCAL;
      t1_set := QUERY (i <* action|
 VALUE_IN(WHICH_CLASS(i), 'version creation'));
      violate := SIZEOF(QUERY (k <* t1_set |
 NOT EXISTS(k.description))) > 0;
   WHERE
      WR1:
         NOT violate;
END_RULE;
```

```
RULE version_deletion_has_mandatory_attribute_description FOR (action );
   LOCAL
      t1_set : SET OF action := [];
      violate : LOGICAL := FALSE;
   END_LOCAL;
      t1_set := QUERY (i <* action|
 VALUE_IN(WHICH_CLASS(i), 'version deletion'));
      violate := SIZEOF(QUERY (k <* t1_set |
 NOT EXISTS(k.description))) > 0;
   WHERE
      WR1:
         NOT violate;
END_RULE;

RULE version_history_has_exactly_one_assigned_group
FOR(applied_group_assignment, group);
  LOCAL
    t1_set:  SET OF group := [];
    set_1, set_2: SET OF applied_group_assignment := [];
    set_3: SET OF group_item := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

t1_set := QUERY(a <* group | VALUE_IN(WHICH_CLASS(a), 'version history'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
   set_1 := QUERY(b <* applied_group_assignment |
            (b.assigned_group = t1_set[i]) AND
           (b.role.name = 'current version'));

   set_2 := QUERY(c <* applied_group_assignment |
             (c.assigned_group = t1_set[i]) AND
             (c.role.name = 'members'));

   violate := ((SIZEOF(set_1) <> 1) OR (SIZEOF(set_2) <> 1));

   IF NOT violate THEN
      set_3 := set_1[1].items * set_2[1].items;

      violate := (SIZEOF(set_3) <> 1) OR
                  NOT (VALUE_IN(WHICH_CLASS(set_3[1]),
            'versionable object'));
   END_IF;
END_REPEAT;

WHERE
    wr1: NOT violate;
END_RULE;

RULE version_history_is_referenced_by_at_least_one_versions
FOR(applied_group_assignment, group);
  LOCAL
    t1_set:  SET OF group := [];
    a_set:  SET OF applied_group_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

t1_set := QUERY(a <* group | VALUE_IN(WHICH_CLASS(a), 'version history'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
   a_set := QUERY(b <* applied_group_assignment |
            (b.assigned_group = t1_set[i]) AND
           (b.role.name = 'versions'));
```



```
    violate := SIZEOF(a_set) < 1;
  END_REPEAT;

WHERE
    wr1: NOT violate;
END_RULE;

RULE version_history_referenced_by_exactly_one_current_version FOR
(applied_group_assignment, group );
   LOCAL
      t1_set : SET OF group := [];
      a_set : SET OF applied_group_assignment := [];
      violate : LOGICAL := FALSE;
   END_LOCAL;
      t1_set := QUERY (a <* group| VALUE_IN(WHICH_CLASS(a), 'version
history'));
      REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
         a_set := QUERY (b <* applied_group_assignment| (b.ASSIGNED_GROUP =
t1_set[i]) AND (b.role.NAME = 'current version'));
         violate := SIZEOF(a_set) <> 1;
      END_REPEAT;
   WHERE
      WR1:
         NOT violate;
END_RULE;

RULE version_history_referenced_by_multiple_roles
FOR(applied_group_assignment, group);
  LOCAL
    t1_set:  SET OF group := [];
    a_set:  SET OF applied_group_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;
t1_set := QUERY(a <* group | VALUE_IN(WHICH_CLASS(a), 'version history'));
REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
   a_set := QUERY(b <* applied_group_assignment |
    (b.assigned_group = t1_set[i]) AND NOT (b.role.name IN ['versions',
'current version', 'relationships']));
    violate := SIZEOF(a_set) < 1;
END_REPEAT;
WHERE
    wr1: NOT violate;
END_RULE;

RULE version_modification_has_mandatory_attribute_description
FOR (action );
   LOCAL
      t1_set : SET OF action := [];
      violate : LOGICAL := FALSE;
   END_LOCAL;
      t1_set := QUERY (i <* action| VALUE_IN(WHICH_CLASS(i), 'version
modification'));
      violate := SIZEOF(QUERY (k <* t1_set |
   NOT EXISTS(k.description))) > 0;
   WHERE
      WR1:
         NOT violate;
END_RULE;

RULE version_relationship_associates_with_versionable_object
FOR (applied_identification_assignment);
LOCAL
    violate: LOGICAL := FALSE;
END_LOCAL;
```

```
REPEAT i := 1 TO HIINDEX(applied_identification_assignment) BY 1 WHILE NOT
violate;
 IF ( (SIZEOF(USEDIN(applied_identification_assignment[i],
('SHIP_ARRANGEMENT_SCHEMA.IDENTIFICATION_ASSIGNMENT_RELATIONSHIP.' +
'RELATING_IDENTIFICATION_ASSIGNMENT'))) > 0) OR
      (SIZEOF(USEDIN(applied_identification_assignment[i],
      ('SHIP_ARRANGEMENT_SCHEMA.IDENTIFICATION_ASSIGNMENT_RELATIONSHIP.' +
'RELATED_IDENTIFICATION_ASSIGNMENT'))) > 0) ) THEN
  REPEAT j := 1 to HIINDEX(applied_identification_assignment[i].items) BY 1
WHILE NOT violate;
   violate := NOT
VALUE_IN(which_class(applied_identification_assignment[i].items[j]),
'versionable object');
  END_REPEAT;
 END_IF;

END_REPEAT;
WHERE
    WR1: NOT violate;
END_RULE;

RULE version_relationship_has_mandatory_attribute_description FOR
(identification_assignment_relationship );
   LOCAL
      t1_set : SET OF identification_assignment_relationship := [];
      violate : LOGICAL := FALSE;
   END_LOCAL;
      t1_set := QUERY (i <* identification_assignment_relationship|
VALUE_IN(WHICH_CLASS(i), 'version relationship'));
      violate := SIZEOF(QUERY (k <* t1_set | NOT EXISTS(k.description))) >
0;
   WHERE
      WR1:
         NOT violate;
END_RULE;

RULE version_relationship_has_unique_versions
FOR (identification_assignment_relationship);
LOCAL
    t1_set:  SET OF identification_assignment_relationship := [];
    violate: LOGICAL := FALSE;
END_LOCAL;
   t1_set := QUERY(a <* identification_assignment_relationship |
   VALUE_IN(WHICH_CLASS(a), 'version relationship'));
REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;
    violate :=
          ( t1_set[i].relating_identification_assignment.assigned_id =
            t1_set[i].related_identification_assignment.assigned_id );
END_REPEAT;
WHERE
    WR1: NOT violate;
END_RULE;

RULE versionable_object_has_one_version_id
FOR(APPLIED_IDENTIFICATION_ASSIGNMENT);
  LOCAL
    version_ids:         SET OF APPLIED_IDENTIFICATION_ASSIGNMENT := [];
    versionable_objects: BAG OF identification_item := [];
    duplicate:           LOGICAL := FALSE;
  END_LOCAL;

  version_ids := QUERY(i <* APPLIED_IDENTIFICATION_ASSIGNMENT |
                       i.ROLE.NAME = 'version identifier');
```



```
  REPEAT i := 1 TO HIINDEX(version_ids);
    versionable_objects := versionable_objects + version_ids[i].items;
  END_REPEAT;
   REPEAT i := 1 TO HIINDEX(versionable_objects) WHILE NOT duplicate;
    REPEAT j := i + 1 TO HIINDEX(versionable_objects) WHILE NOT duplicate;
      duplicate := versionable_objects[i] :=: versionable_objects[j];
    END_REPEAT;
  END_REPEAT;
  WHERE
    WR1: NOT duplicate;
END_RULE;

RULE versioned_action_request_with_identification_assignment
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF versioned_action_request := [];
    t2_set:  SET OF applied_identification_assignment := [];
    arg_list:  LIST OF STRING := ['change request'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

    REPEAT j:=1 TO HIINDEX(arg_list) WHILE (NOT violation);
        c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.assigned_class.NAME =  arg_LIST[j]);
   END_REPEAT;

  REPEAT i := 1 TO HIINDEX(c_a_set);
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items);
      t1_set := t1_set + c_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;


    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      t2_set := bag_to_set(USEDIN(t1_set[i],
'SHIP_ARRANGEMENT_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
      t2_set := QUERY ( j <* t2_set |
                j.role.name = 'globally unambiguous identifier');
violation := NOT (SIZEOF(T2_SET) = 1);
    END_REPEAT;

  WHERE
    wr1: NOT violation;
END_RULE;

RULE versions_is_referenced_by_at_least_one_version_history
FOR(applied_group_assignment, group);
  LOCAL
    t1_set:  SET OF group := [];
    a_set:  SET OF applied_group_assignment := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

t1_set := QUERY(a <* group |
 VALUE_IN(WHICH_CLASS(a), 'versions'));

REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violate;

   a_set := QUERY(b <* applied_group_assignment |
      (b.assigned_group = t1_set[i]) AND
      (b.role.name = 'version history'));

    violate := SIZEOF(a_set) < 1;
  END_REPEAT;
```

```
WHERE
    wr1: NOT violate;
END_RULE;


FUNCTION acyclic_curve_replica(
             rep: curve_replica;
             parent: curve
    ): BOOLEAN;
  IF NOT ('SHIP_ARRANGEMENT_SCHEMA.CURVE_REPLICA' IN TYPEOF(parent))
       THEN
    RETURN(TRUE);
  END_IF;
  IF parent :=: rep THEN
    RETURN(FALSE);
  ELSE
    RETURN(acyclic_curve_replica(rep,parent\curve_replica.parent_curve));
  END_IF;

END_FUNCTION; -- acyclic_curve_replica

FUNCTION acyclic_mapped_representation(
             parent_set: SET OF representation;
             children_set: SET OF representation_item
    ): BOOLEAN;

  LOCAL
    x : SET OF representation_item;
    y : SET OF representation_item;
  END_LOCAL;
  x := QUERY ( z <* children_set | (
      'SHIP_ARRANGEMENT_SCHEMA.MAPPED_ITEM' IN TYPEOF(z)) );
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x) BY 1;
      IF x[i]\mapped_item.mapping_source.mapped_representation IN
          parent_set THEN
        RETURN(FALSE);
      END_IF;
      IF NOT acyclic_mapped_representation(parent_set + x[i]\mapped_item
          .mapping_source.mapped_representation,x[i]\mapped_item.
          mapping_source.mapped_representation.items) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  x := children_set - x;
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x) BY 1;
      y := QUERY ( z <* bag_to_set(USEDIN(x[i],'')) | (
          'SHIP_ARRANGEMENT_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z)) );
      IF NOT acyclic_mapped_representation(parent_set,y) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(TRUE);

END_FUNCTION; -- acyclic_mapped_representation

FUNCTION acyclic_surface_replica(
             rep: surface_replica;
             parent: surface
    ): BOOLEAN;
  IF NOT ('SHIP_ARRANGEMENT_SCHEMA.SURFACE_REPLICA' IN TYPEOF(parent))
```



```
      THEN
    RETURN(TRUE);
  END_IF;
  IF parent :=: rep THEN
    RETURN(FALSE);
  ELSE
    RETURN(acyclic_surface_replica(rep,parent\surface_replica.
        parent_surface));
  END_IF;

END_FUNCTION; -- acyclic_surface_replica

FUNCTION associated_surface(
            arg: pcurve_or_surface
    ): surface;

  LOCAL
    surf : surface;
  END_LOCAL;
  IF 'SHIP_ARRANGEMENT_SCHEMA.PCURVE' IN TYPEOF(arg) THEN
    surf := arg.basis_surface;
  ELSE
    surf := arg;
  END_IF;
  RETURN(surf);

END_FUNCTION; -- associated_surface

FUNCTION bag_to_set(
            the_bag: BAG OF GENERIC:intype
    ): SET OF GENERIC:intype;

  LOCAL
    the_set : SET OF GENERIC:intype := [];
  END_LOCAL;
  IF SIZEOF(the_bag) > 0 THEN
    REPEAT i := 1 TO HIINDEX(the_bag) BY 1;
      the_set := the_set + the_bag[i];
    END_REPEAT;
  END_IF;
  RETURN(the_set);

END_FUNCTION; -- bag_to_set

FUNCTION base_axis(
            dim: INTEGER;
            axis1, axis2, axis3: direction
    ): LIST [2:3] OF direction;

  LOCAL
    u      : LIST [2:3] OF direction;
    d1     : direction;
    d2     : direction;
    factor : REAL;
  END_LOCAL;
  IF dim = 3 THEN
    d1 := NVL(normalise(axis3),dummy_gri || direction([0,0,1]));
    d2 := first_proj_axis(d1,axis1);
    u := [d2,second_proj_axis(d1,d2,axis2),d1];
  ELSE
    IF EXISTS(axis1) THEN
      d1 := normalise(axis1);
      u := [d1,orthogonal_complement(d1)];
      IF EXISTS(axis2) THEN
        factor := dot_product(axis2,u[2]);
        IF factor < 0 THEN
```

```
          u[2].direction_ratios[1] := -u[2].direction_ratios[1];
          u[2].direction_ratios[2] := -u[2].direction_ratios[2];
        END_IF;
      END_IF;
    ELSE
      IF EXISTS(axis2) THEN
        d1 := normalise(axis2);
        u := [orthogonal_complement(d1),d1];
        u[1].direction_ratios[1] := -u[1].direction_ratios[1];
        u[1].direction_ratios[2] := -u[1].direction_ratios[2];
      ELSE
        u := [dummy_gri || direction([1,0]),dummy_gri || direction([0,1])];
      END_IF;
    END_IF;
  END_IF;
  RETURN(u);

END_FUNCTION; -- base_axis

FUNCTION boolean_choose(
             b: BOOLEAN;
             choice1, choice2: GENERIC:item
    ): GENERIC:item;
  IF b THEN
    RETURN(choice1);
  ELSE
    RETURN(choice2);
  END_IF;

END_FUNCTION; -- boolean_choose

FUNCTION build_2axes(
             ref_direction: direction
    ): LIST [2:2] OF direction;

  LOCAL
    d : direction := NVL(normalise(ref_direction),dummy_gri ||
         direction([1,0]));
  END_LOCAL;
  RETURN([d,orthogonal_complement(d)]);

END_FUNCTION; -- build_2axes

FUNCTION build_axes(
             axis, ref_direction: direction
    ): LIST [3:3] OF direction;

  LOCAL
    d1 : direction;
    d2 : direction;
  END_LOCAL;
  d1 := NVL(normalise(axis),dummy_gri || direction([0,0,1]));
  d2 := first_proj_axis(d1,ref_direction);
  RETURN([d2,normalise(cross_product(d1,d2)).orientation,d1]);

END_FUNCTION; -- build_axes

FUNCTION closed_shell_reversed(
             a_shell: closed_shell
    ): oriented_closed_shell;

  LOCAL
    the_reverse : oriented_closed_shell;
  END_LOCAL;
```



```
  IF 'SHIP_ARRANGEMENT_SCHEMA.ORIENTED_CLOSED_SHELL' IN TYPEOF(a_shell)
       THEN
    the_reverse := ((dummy_tri || connected_face_set(a_shell\
        connected_face_set.cfs_faces)) || closed_shell()) ||
        oriented_closed_shell(a_shell\oriented_closed_shell.
        closed_shell_element,NOT a_shell\oriented_closed_shell.
        orientation);
  ELSE
    the_reverse := ((dummy_tri || connected_face_set(a_shell\
        connected_face_set.cfs_faces)) || closed_shell()) ||
        oriented_closed_shell(a_shell,FALSE);
  END_IF;
  RETURN(the_reverse);

END_FUNCTION; -- closed_shell_reversed

FUNCTION conditional_reverse(
            p: BOOLEAN;
            an_item: reversible_topology
    ): reversible_topology;
  IF p THEN
    RETURN(an_item);
  ELSE
    RETURN(topology_reversed(an_item));
  END_IF;

END_FUNCTION; -- conditional_reverse

FUNCTION constraints_composite_curve_on_surface(
            c: composite_curve_on_surface
    ): BOOLEAN;

  LOCAL
    n_segments : INTEGER := SIZEOF(c.segments);
  END_LOCAL;
  REPEAT k := 1 TO n_segments BY 1;
    IF ((NOT ('SHIP_ARRANGEMENT_SCHEMA.PCURVE' IN TYPEOF(c\
        composite_curve.segments[k].parent_curve))) AND (NOT (
        'SHIP_ARRANGEMENT_SCHEMA.SURFACE_CURVE' IN TYPEOF(c\
        composite_curve.segments[k].parent_curve)))) AND (NOT (
        'SHIP_ARRANGEMENT_SCHEMA.COMPOSITE_CURVE_ON_SURFACE' IN TYPEOF(c
        \composite_curve.segments[k].parent_curve))) THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  RETURN(TRUE);

END_FUNCTION; -- constraints_composite_curve_on_surface

FUNCTION constraints_param_b_spline(
            degree, up_knots, up_cp: INTEGER;
            knot_mult: LIST OF INTEGER;
            knots: LIST OF parameter_value
    ): BOOLEAN;

  LOCAL
    k      : INTEGER;
    sum    : INTEGER;
    result : BOOLEAN := TRUE;
  END_LOCAL;
  sum := knot_mult[1];
  REPEAT i := 2 TO up_knots BY 1;
    sum := sum + knot_mult[i];
  END_REPEAT;
  IF (((degree < 1) OR (up_knots < 2)) OR (up_cp < degree)) OR (sum <> (
      (degree + up_cp) + 2)) THEN
```

```
    result := FALSE;
    RETURN(result);
  END_IF;
  k := knot_mult[1];
  IF (k < 1) OR (k > (degree + 1)) THEN
    result := FALSE;
    RETURN(result);
  END_IF;
  REPEAT i := 2 TO up_knots BY 1;
    IF (knot_mult[i] < 1) OR (knots[i] <= knots[i - 1]) THEN
      result := FALSE;
      RETURN(result);
    END_IF;
    k := knot_mult[i];
    IF (i < up_knots) AND (k > degree) THEN
      result := FALSE;
      RETURN(result);
    END_IF;
    IF (i = up_knots) AND (k > (degree + 1)) THEN
      result := FALSE;
      RETURN(result);
    END_IF;
  END_REPEAT;
  RETURN(result);

END_FUNCTION; -- constraints_param_b_spline

FUNCTION cross_product(
            arg1, arg2: direction
    ): vector;

  LOCAL
    v2     : LIST [3:3] OF REAL;
    v1     : LIST [3:3] OF REAL;
    mag    : REAL;
    res    : direction;
    result : vector;
  END_LOCAL;
  IF ((NOT EXISTS(arg1)) OR (arg1.dim = 2)) OR ((NOT EXISTS(arg2)) OR (
      arg2.dim = 2)) THEN
    RETURN(?);
  ELSE
    BEGIN
      v1 := normalise(arg1).direction_ratios;
      v2 := normalise(arg2).direction_ratios;
      res := dummy_gri || direction([(v1[2] * v2[3]) - (v1[3] * v2[2]),(
          v1[3] * v2[1]) - (v1[1] * v2[3]),(v1[1] * v2[2]) - (v1[2] * v2[
          1])]);
      mag := 0;
      REPEAT i := 1 TO 3 BY 1;
        mag := mag + (res.direction_ratios[i] * res.direction_ratios[i]);
      END_REPEAT;
      IF mag > 0 THEN
        result := dummy_gri || vector(res,SQRT(mag));
      ELSE
        result := dummy_gri || vector(arg1,0);
      END_IF;
      RETURN(result);
    END;
  END_IF;

END_FUNCTION; -- cross_product

FUNCTION curve_weights_positive(
```



```
            b: rational_b_spline_curve
    ): BOOLEAN;

  LOCAL
    result : BOOLEAN := TRUE;
  END_LOCAL;
  REPEAT i := 0 TO b.upper_index_on_control_points BY 1;
    IF b.weights[i] <= 0 THEN
      result := FALSE;
      RETURN(result);
    END_IF;
  END_REPEAT;
  RETURN(result);

END_FUNCTION; -- curve_weights_positive

FUNCTION derive_dimensional_exponents(
            x: unit
    ): dimensional_exponents;

  LOCAL
    result : dimensional_exponents := dimensional_exponents(0,0,0,0,0,0,
             0);
  END_LOCAL;
  IF 'SHIP_ARRANGEMENT_SCHEMA.DERIVED_UNIT' IN TYPEOF(x) THEN
    REPEAT i := LOINDEX(x.elements) TO HIINDEX(x.elements) BY 1;
      result.length_exponent := result.length_exponent + (x.elements[i].
          exponent * x.elements[i].unit.dimensions.length_exponent);
      result.mass_exponent := result.mass_exponent + (x.elements[i].
          exponent * x.elements[i].unit.dimensions.mass_exponent);
      result.time_exponent := result.time_exponent + (x.elements[i].
          exponent * x.elements[i].unit.dimensions.time_exponent);
      result.electric_current_exponent := result.
          electric_current_exponent + (x.elements[i].exponent * x.
          elements[i].unit.dimensions.electric_current_exponent);
      result.thermodynamic_temperature_exponent := result.
          thermodynamic_temperature_exponent + (x.elements[i].exponent *
          x.elements[i].unit.dimensions.
          thermodynamic_temperature_exponent);
      result.amount_of_substance_exponent := result.
          amount_of_substance_exponent + (x.elements[i].exponent * x.
          elements[i].unit.dimensions.amount_of_substance_exponent);
      result.luminous_intensity_exponent := result.
          luminous_intensity_exponent + (x.elements[i].exponent * x.
          elements[i].unit.dimensions.luminous_intensity_exponent);
    END_REPEAT;
  ELSE
    result := x.dimensions;
  END_IF;
  RETURN(result);

END_FUNCTION; -- derive_dimensional_exponents

FUNCTION dimension_of(
            item: geometric_representation_item
    ): dimension_count;

  LOCAL
    x   : SET OF representation;
    y   : representation_context;
    dim : dimension_count;
  END_LOCAL;
  IF 'SHIP_ARRANGEMENT_SCHEMA.CARTESIAN_POINT' IN TYPEOF(item) THEN
    dim := SIZEOF(item\cartesian_point.coordinates);
    RETURN(dim);
  END_IF;
```

```
  IF 'SHIP_ARRANGEMENT_SCHEMA.DIRECTION' IN TYPEOF(item) THEN
    dim := SIZEOF(item\direction.direction_ratios);
    RETURN(dim);
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.VECTOR' IN TYPEOF(item) THEN
    dim := SIZEOF(item\vector.orientation\direction.direction_ratios);
    RETURN(dim);
  END_IF;
  x := using_representations(item);
  y := x[1].context_of_items;
  dim := y\geometric_representation_context.coordinate_space_dimension;
  RETURN(dim);

END_FUNCTION; -- dimension_of

FUNCTION dimensions_for_si_unit(
             n: si_unit_name
    ): dimensional_exponents;
  CASE n OF
    metre         :         RETURN(dimensional_exponents(1,0,0,0,0,0,0));
    gram          :         RETURN(dimensional_exponents(0,1,0,0,0,0,0));
    second        :         RETURN(dimensional_exponents(0,0,1,0,0,0,0));
    ampere        :         RETURN(dimensional_exponents(0,0,0,1,0,0,0));
    kelvin        :         RETURN(dimensional_exponents(0,0,0,0,1,0,0));
    mole          :         RETURN(dimensional_exponents(0,0,0,0,0,1,0));
    candela       :         RETURN(dimensional_exponents(0,0,0,0,0,0,1));
    radian        :         RETURN(dimensional_exponents(0,0,0,0,0,0,0));
    steradian     :         RETURN(dimensional_exponents(0,0,0,0,0,0,0));
    hertz         :         RETURN(dimensional_exponents(0,0,-1,0,0,0,0));
    newton        :         RETURN(dimensional_exponents(1,1,-2,0,0,0,0));
    pascal        :         RETURN(dimensional_exponents(-1,1,-2,0,0,0,0));
    joule         :         RETURN(dimensional_exponents(2,1,-2,0,0,0,0));
    watt          :         RETURN(dimensional_exponents(2,1,-3,0,0,0,0));
    coulomb       :         RETURN(dimensional_exponents(0,0,1,1,0,0,0));
    volt          :         RETURN(dimensional_exponents(2,1,-3,-1,0,0,0));
    farad         :         RETURN(dimensional_exponents(-2,-1,4,1,0,0,0));
    ohm           :         RETURN(dimensional_exponents(2,1,-3,-2,0,0,0));
    siemens       :         RETURN(dimensional_exponents(-2,-1,3,2,0,0,0));
    weber         :         RETURN(dimensional_exponents(2,1,-2,-1,0,0,0));
    tesla         :         RETURN(dimensional_exponents(0,1,-2,-1,0,0,0));
    henry         :         RETURN(dimensional_exponents(2,1,-2,-2,0,0,0));
    degree_celsius  :         RETURN(dimensional_exponents(0,0,0,0,1,0,0));
    lumen         :         RETURN(dimensional_exponents(0,0,0,0,0,0,1));
    lux           :         RETURN(dimensional_exponents(-2,0,0,0,0,0,1));
    becquerel     :         RETURN(dimensional_exponents(0,0,-1,0,0,0,0));
    gray          :         RETURN(dimensional_exponents(2,0,-2,0,0,0,0));
    sievert       :         RETURN(dimensional_exponents(2,0,-2,0,0,0,0));
    END_CASE;

END_FUNCTION; -- dimensions_for_si_unit

FUNCTION dot_product(
             arg1, arg2: direction
    ): REAL;

  LOCAL
    ndim   : INTEGER;
    scalar : REAL;
    vec1   : direction;
    vec2   : direction;
  END_LOCAL;
  IF (NOT EXISTS(arg1)) OR (NOT EXISTS(arg2)) THEN
    scalar := ?;
  ELSE
```



```
    IF arg1.dim <> arg2.dim THEN
      scalar := ?;
    ELSE
      BEGIN
        vec1 := normalise(arg1);
        vec2 := normalise(arg2);
        ndim := arg1.dim;
        scalar := 0;
        REPEAT i := 1 TO ndim BY 1;
          scalar := scalar + (vec1.direction_ratios[i] * vec2.
              direction_ratios[i]);
        END_REPEAT;
      END;
    END_IF;
  END_IF;
  RETURN(scalar);

END_FUNCTION; -- dot_product

FUNCTION edge_reversed(
            an_edge: edge
    ): oriented_edge;

  LOCAL
    the_reverse : oriented_edge;
  END_LOCAL;
  IF 'SHIP_ARRANGEMENT_SCHEMA.ORIENTED_EDGE' IN TYPEOF(an_edge) THEN
    the_reverse := (dummy_tri || edge(an_edge.edge_end,an_edge.
        edge_start)) || oriented_edge(an_edge\oriented_edge.edge_element,
        NOT an_edge\oriented_edge.orientation);
  ELSE
    the_reverse := (dummy_tri || edge(an_edge.edge_end,an_edge.
        edge_start)) || oriented_edge(an_edge,FALSE);
  END_IF;
  RETURN(the_reverse);

END_FUNCTION; -- edge_reversed

FUNCTION face_bound_reversed(
            a_face_bound: face_bound
    ): face_bound;

  LOCAL
    the_reverse : face_bound;
  END_LOCAL;
  IF 'SHIP_ARRANGEMENT_SCHEMA.FACE_OUTER_BOUND' IN TYPEOF(a_face_bound)
       THEN
    the_reverse := (dummy_tri || face_bound(a_face_bound\face_bound.
        bound,NOT a_face_bound\face_bound.orientation)) ||
        face_outer_bound();
  ELSE
    the_reverse := dummy_tri || face_bound(a_face_bound.bound,NOT
        a_face_bound.orientation);
  END_IF;
  RETURN(the_reverse);

END_FUNCTION; -- face_bound_reversed

FUNCTION face_reversed(
            a_face: face
    ): oriented_face;

  LOCAL
    the_reverse : oriented_face;
  END_LOCAL;
  IF 'SHIP_ARRANGEMENT_SCHEMA.ORIENTED_FACE' IN TYPEOF(a_face) THEN
```

```
    the_reverse := (dummy_tri || face(set_of_topology_reversed(a_face.
        bounds))) || oriented_face(a_face\oriented_face.face_element,NOT
        a_face\oriented_face.orientation);
  ELSE
    the_reverse := (dummy_tri || face(set_of_topology_reversed(a_face.
        bounds))) || oriented_face(a_face,FALSE);
  END_IF;
  RETURN(the_reverse);

END_FUNCTION; -- face_reversed

FUNCTION first_proj_axis(
             z_axis, arg: direction
    ): direction;

  LOCAL
    x_vec  : vector;
    v      : direction;
    z      : direction;
    x_axis : direction;
  END_LOCAL;
  IF NOT EXISTS(z_axis) THEN
    RETURN(?);
  ELSE
    z := normalise(z_axis);
    IF NOT EXISTS(arg) THEN
      IF z.direction_ratios <> [1,0,0] THEN
        v := dummy_gri || direction([1,0,0]);
      ELSE
        v := dummy_gri || direction([0,1,0]);
      END_IF;
    ELSE
      IF arg.dim <> 3 THEN
        RETURN(?);
      END_IF;
      IF cross_product(arg,z).magnitude = 0 THEN
        RETURN(?);
      ELSE
        v := normalise(arg);
      END_IF;
    END_IF;
    x_vec := scalar_times_vector(dot_product(v,z),z);
    x_axis := vector_difference(v,x_vec).orientation;
    x_axis := normalise(x_axis);
  END_IF;
  RETURN(x_axis);

END_FUNCTION; -- first_proj_axis

FUNCTION get_basis_surface(
             c: curve_on_surface
    ): SET [0:2] OF surface;

  LOCAL
    surfs : SET [0:2] OF surface;
    n     : INTEGER;
  END_LOCAL;
  surfs := [];
  IF 'SHIP_ARRANGEMENT_SCHEMA.PCURVE' IN TYPEOF(c) THEN
    surfs := [c\pcurve.basis_surface];
  ELSE
    IF 'SHIP_ARRANGEMENT_SCHEMA.SURFACE_CURVE' IN TYPEOF(c) THEN
      n := SIZEOF(c\surface_curve.associated_geometry);
      REPEAT i := 1 TO n BY 1;
```

```
        surfs := surfs + associated_surface(c\surface_curve.
            associated_geometry[i]);
      END_REPEAT;
    END_IF;
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.COMPOSITE_CURVE_ON_SURFACE' IN TYPEOF(c)
       THEN
    n := SIZEOF(c\composite_curve.segments);
    surfs := get_basis_surface(c\composite_curve.segments[1].
        parent_curve);
    IF n > 1 THEN
      REPEAT i := 2 TO n BY 1;
        surfs := surfs * get_basis_surface(c\composite_curve.segments[i]
            .parent_curve);
      END_REPEAT;
    END_IF;
  END_IF;
  RETURN(surfs);

END_FUNCTION; -- get_basis_surface

FUNCTION get_description_value(
             obj: description_attribute_select
    ): text;

  LOCAL
    description_bag : BAG OF description_attribute := USEDIN(obj,(
                        'SHIP_ARRANGEMENT_SCHEMA.' +
                        'DESCRIPTION_ATTRIBUTE.') + 'DESCRIBED_ITEM');
  END_LOCAL;
  IF SIZEOF(description_bag) = 1 THEN
    RETURN(description_bag[1].attribute_value);
  ELSE
    RETURN(?);
  END_IF;

END_FUNCTION; -- get_description_value

FUNCTION get_id_value(
             obj: id_attribute_select
    ): identifier;

  LOCAL
    id_bag : BAG OF id_attribute := USEDIN(obj,(
               'SHIP_ARRANGEMENT_SCHEMA.' + 'ID_ATTRIBUTE.') +
               'IDENTIFIED_ITEM');
  END_LOCAL;
  IF SIZEOF(id_bag) = 1 THEN
    RETURN(id_bag[1].attribute_value);
  ELSE
    RETURN(?);
  END_IF;

END_FUNCTION; -- get_id_value

FUNCTION get_name_value(
             obj: name_attribute_select
    ): label;

  LOCAL
    name_bag : BAG OF name_attribute := USEDIN(obj,(
                 'SHIP_ARRANGEMENT_SCHEMA.' + 'NAME_ATTRIBUTE.') +
                 'NAMED_ITEM');
  END_LOCAL;
  IF SIZEOF(name_bag) = 1 THEN
    RETURN(name_bag[1].attribute_value);
```

```
  ELSE
    RETURN(?);
  END_IF;

END_FUNCTION; -- get_name_value

FUNCTION get_role(
             obj: role_select
    ): object_role;

  LOCAL
    role_bag : BAG OF role_association := USEDIN(obj,(
                'SHIP_ARRANGEMENT_SCHEMA.' + 'ROLE_ASSOCIATION.') +
                'ITEM_WITH_ROLE');
  END_LOCAL;
  IF SIZEOF(role_bag) = 1 THEN
    RETURN(role_bag[1].role);
  ELSE
    RETURN(?);
  END_IF;

END_FUNCTION; -- get_role

FUNCTION item_in_context(
             item: representation_item;
             cntxt: representation_context
    ): BOOLEAN;

  LOCAL
    y : BAG OF representation_item;
  END_LOCAL;
  IF SIZEOF(USEDIN(item,'SHIP_ARRANGEMENT_SCHEMA.REPRESENTATION.ITEMS')
      * cntxt.representations_in_context) > 0 THEN
    RETURN(TRUE);
  ELSE
    y := QUERY ( z <* USEDIN(item,'') | (
        'SHIP_ARRANGEMENT_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z)) );
    IF SIZEOF(y) > 0 THEN
      REPEAT i := 1 TO HIINDEX(y) BY 1;
        IF item_in_context(y[i],cntxt) THEN
          RETURN(TRUE);
        END_IF;
      END_REPEAT;
    END_IF;
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- item_in_context

FUNCTION leap_year(
             year: year_number
    ): BOOLEAN;
  IF (((year MOD 4) = 0) AND ((year MOD 100) <> 0)) OR ((year MOD 400) =
       0) THEN
    RETURN(TRUE);
  ELSE
    RETURN(FALSE);
  END_IF;

END_FUNCTION; -- leap_year

FUNCTION list_face_loops(
             f: face
    ): LIST [0:?] OF loop;
```

```
  LOCAL
    loops : LIST [0:?] OF loop := [];
  END_LOCAL;
  REPEAT i := 1 TO SIZEOF(f.bounds) BY 1;
    loops := loops + f.bounds[i].bound;
  END_REPEAT;
  RETURN(loops);

END_FUNCTION; -- list_face_loops

FUNCTION list_of_topology_reversed(
            a_list: list_of_reversible_topology_item
    ): list_of_reversible_topology_item;

  LOCAL
    the_reverse : list_of_reversible_topology_item;
  END_LOCAL;
  the_reverse := [];
  REPEAT i := 1 TO SIZEOF(a_list) BY 1;
    the_reverse := topology_reversed(a_list[i]) + the_reverse;
  END_REPEAT;
  RETURN(the_reverse);

END_FUNCTION; -- list_of_topology_reversed

FUNCTION list_to_array(
            lis: LIST [0:?] OF GENERIC:t;
            low, u: INTEGER
    ): ARRAY OF GENERIC:t;

  LOCAL
    n   : INTEGER;
    res : ARRAY [low:u] OF GENERIC:t;
  END_LOCAL;
  n := SIZEOF(lis);
  IF n <> ((u - low) + 1) THEN
    RETURN(?);
  ELSE
    res := [lis[1],n];
    REPEAT i := 2 TO n BY 1;
      res[(low + i) - 1] := lis[i];
    END_REPEAT;
    RETURN(res);
  END_IF;

END_FUNCTION; -- list_to_array

FUNCTION list_to_set(
            l: LIST [0:?] OF GENERIC:t
    ): SET OF GENERIC:t;

  LOCAL
    s : SET OF GENERIC:t := [];
  END_LOCAL;
  REPEAT i := 1 TO SIZEOF(l) BY 1;
    s := s + l[i];
  END_REPEAT;
  RETURN(s);

END_FUNCTION; -- list_to_set

FUNCTION make_array_of_array(
            lis: LIST [1:?] OF LIST [1:?] OF GENERIC:t;
            low1, u1, low2, u2: INTEGER
    ): ARRAY OF ARRAY OF GENERIC:t;
```

```
  LOCAL
    res : ARRAY [low1:u1] OF ARRAY [low2:u2] OF GENERIC:t;
  END_LOCAL;
  IF ((u1 - low1) + 1) <> SIZEOF(lis) THEN
    RETURN(?);
  END_IF;
  IF ((u2 - low2) + 1) <> SIZEOF(lis[1]) THEN
    RETURN(?);
  END_IF;
  res := [list_to_array(lis[1],low2,u2),(u1 - low1) + 1];
  REPEAT i := 2 TO HIINDEX(lis) BY 1;
    IF ((u2 - low2) + 1) <> SIZEOF(lis[i]) THEN
      RETURN(?);
    END_IF;
    res[(low1 + i) - 1] := list_to_array(lis[i],low2,u2);
  END_REPEAT;
  RETURN(res);

END_FUNCTION; -- make_array_of_array

FUNCTION mixed_loop_type_set(
             l: SET [0:?] OF loop
    ): LOGICAL;

  LOCAL
    poly_loop_type : LOGICAL;
  END_LOCAL;
  IF SIZEOF(l) <= 1 THEN
    RETURN(FALSE);
  END_IF;
  poly_loop_type := 'SHIP_ARRANGEMENT_SCHEMA.POLY_LOOP' IN TYPEOF(l[1]);
  REPEAT i := 2 TO SIZEOF(l) BY 1;
    IF ('SHIP_ARRANGEMENT_SCHEMA.POLY_LOOP' IN TYPEOF(l[i])) <>
        poly_loop_type THEN
      RETURN(TRUE);
    END_IF;
  END_REPEAT;
  RETURN(FALSE);

END_FUNCTION; -- mixed_loop_type_set

FUNCTION nmsf_curve_check(
             cv: curve
    ): BOOLEAN;
  IF SIZEOF(['SHIP_ARRANGEMENT_SCHEMA.BOUNDED_CURVE',
      'SHIP_ARRANGEMENT_SCHEMA.CONIC',
      'SHIP_ARRANGEMENT_SCHEMA.CURVE_REPLICA',
      'SHIP_ARRANGEMENT_SCHEMA.LINE',
      'SHIP_ARRANGEMENT_SCHEMA.OFFSET_CURVE_3D'] * TYPEOF(cv)) > 1 THEN
    RETURN(FALSE);
  ELSE
    IF (('SHIP_ARRANGEMENT_SCHEMA.B_SPLINE_CURVE' IN TYPEOF(cv)) AND (
        cv\b_spline_curve.self_intersect = FALSE)) OR (cv\b_spline_curve.
        self_intersect = UNKNOWN) THEN
      RETURN(TRUE);
    ELSE
      IF SIZEOF(['SHIP_ARRANGEMENT_SCHEMA.CONIC',
          'SHIP_ARRANGEMENT_SCHEMA.LINE'] * TYPEOF(cv)) = 1 THEN
        RETURN(TRUE);
      ELSE
        IF 'SHIP_ARRANGEMENT_SCHEMA.CURVE_REPLICA' IN TYPEOF(cv) THEN
          RETURN(nmsf_curve_check(cv\curve_replica.parent_curve));
        ELSE
```

```
          IF (('SHIP_ARRANGEMENT_SCHEMA.OFFSET_CURVE_3D' IN TYPEOF(cv))
              AND ((cv\offset_curve_3d.self_intersect = FALSE) OR (cv\
              offset_curve_3d.self_intersect = UNKNOWN))) AND (NOT (
              'SHIP_ARRANGEMENT_SCHEMA.POLYLINE' IN TYPEOF(cv.
              basis_curve))) THEN
            RETURN(nmsf_curve_check(cv\offset_curve_3d.basis_curve));
          ELSE
            IF 'SHIP_ARRANGEMENT_SCHEMA.PCURVE' IN TYPEOF(cv) THEN
              RETURN(nmsf_curve_check(cv\pcurve.reference_to_curve\
                  representation.items[1]) AND nmsf_surface_check(cv\
                  pcurve.basis_surface));
            ELSE
              IF 'SHIP_ARRANGEMENT_SCHEMA.SURFACE_CURVE' IN TYPEOF(cv)
                   THEN
                IF nmsf_curve_check(cv\surface_curve.curve_3d) THEN
                  REPEAT i := 1 TO SIZEOF(cv\surface_curve.
                      associated_geometry) BY 1;
                    IF 'SHIP_ARRANGEMENT_SCHEMA.SURFACE' IN TYPEOF(cv\
                        surface_curve.associated_geometry[i]) THEN
                      IF NOT nmsf_surface_check(cv\surface_curve.
                          associated_geometry[i]) THEN
                        RETURN(FALSE);
                      END_IF;
                    ELSE
                      IF 'SHIP_ARRANGEMENT_SCHEMA.PCURVE' IN TYPEOF(cv\
                          surface_curve.associated_geometry[i]) THEN
                        IF NOT nmsf_curve_check(cv\surface_curve.
                            associated_geometry[i]) THEN
                          RETURN(FALSE);
                        END_IF;
                      END_IF;
                    END_IF;
                  END_REPEAT;
                  RETURN(TRUE);
                END_IF;
              ELSE
                IF 'SHIP_ARRANGEMENT_SCHEMA.POLYLINE' IN TYPEOF(cv)
                     THEN
                  IF SIZEOF(cv\polyline.points) >= 3 THEN
                    RETURN(TRUE);
                  END_IF;
                END_IF;
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- nmsf_curve_check

FUNCTION nmsf_surface_check(
             surf: surface
    ): BOOLEAN;
  IF 'SHIP_ARRANGEMENT_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF(surf) THEN
    RETURN(TRUE);
  ELSE
    IF 'SHIP_ARRANGEMENT_SCHEMA.SWEPT_SURFACE' IN TYPEOF(surf) THEN
      RETURN(nmsf_curve_check(surf\swept_surface.swept_curve));
    ELSE
      IF (('SHIP_ARRANGEMENT_SCHEMA.OFFSET_SURFACE' IN TYPEOF(surf))
          AND (surf\offset_surface.self_intersect = FALSE)) OR (surf\
          offset_surface.self_intersect = UNKNOWN) THEN
        RETURN(nmsf_surface_check(surf\offset_surface.basis_surface));
```

```
      ELSE
        IF 'SHIP_ARRANGEMENT_SCHEMA.SURFACE_REPLICA' IN TYPEOF(surf)
             THEN
          RETURN(nmsf_surface_check(surf\surface_replica.parent_surface));
        ELSE
          IF (('SHIP_ARRANGEMENT_SCHEMA.B_SPLINE_SURFACE' IN TYPEOF(
              surf)) AND (surf\b_spline_surface.self_intersect = FALSE))
              OR (surf\b_spline_surface.self_intersect = UNKNOWN) THEN
            RETURN(TRUE);
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- nmsf_surface_check

FUNCTION normalise(
             arg: vector_or_direction
    ): vector_or_direction;

  LOCAL
    ndim   : INTEGER;
    v      : direction;
    vec    : vector;
    mag    : REAL;
    result : vector_or_direction;
  END_LOCAL;
  IF NOT EXISTS(arg) THEN
    result := ?;
  ELSE
    ndim := arg.dim;
    IF 'SHIP_ARRANGEMENT_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
      BEGIN
        v := dummy_gri || direction(arg.orientation.direction_ratios);
        IF arg.magnitude = 0 THEN
          RETURN(?);
        ELSE
          vec := dummy_gri || vector(v,1);
        END_IF;
      END;
    ELSE
      v := dummy_gri || direction(arg.direction_ratios);
    END_IF;
    mag := 0;
    REPEAT i := 1 TO ndim BY 1;
      mag := mag + (v.direction_ratios[i] * v.direction_ratios[i]);
    END_REPEAT;
    IF mag > 0 THEN
      mag := SQRT(mag);
      REPEAT i := 1 TO ndim BY 1;
        v.direction_ratios[i] := v.direction_ratios[i] / mag;
      END_REPEAT;
      IF 'SHIP_ARRANGEMENT_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
        vec.orientation := v;
        result := vec;
      ELSE
        result := v;
      END_IF;
    ELSE
      RETURN(?);
    END_IF;
  END_IF;
```

```
  RETURN(result);

END_FUNCTION; -- normalise

FUNCTION open_shell_reversed(
             a_shell: open_shell
    ): oriented_open_shell;

  LOCAL
    the_reverse : oriented_open_shell;
  END_LOCAL;
  IF 'SHIP_ARRANGEMENT_SCHEMA.ORIENTED_OPEN_SHELL' IN TYPEOF(a_shell)
       THEN
    the_reverse := ((dummy_tri || connected_face_set(a_shell\
        connected_face_set.cfs_faces)) || open_shell()) ||
        oriented_open_shell(a_shell\oriented_open_shell.
        open_shell_element,NOT a_shell\oriented_open_shell.orientation);
  ELSE
    the_reverse := ((dummy_tri || connected_face_set(a_shell\
        connected_face_set.cfs_faces)) || open_shell()) ||
        oriented_open_shell(a_shell,FALSE);
  END_IF;
  RETURN(the_reverse);

END_FUNCTION; -- open_shell_reversed

FUNCTION orthogonal_complement(
             vec: direction
    ): direction;

  LOCAL
    result : direction;
  END_LOCAL;
  IF (vec.dim <> 2) OR (NOT EXISTS(vec)) THEN
    RETURN(?);
  ELSE
    result := dummy_gri || direction([-vec.direction_ratios[2],vec.
        direction_ratios[1]]);
    RETURN(result);
  END_IF;

END_FUNCTION; -- orthogonal_complement

FUNCTION path_head_to_tail(
             a_path: path
    ): BOOLEAN;

  LOCAL
    n : INTEGER;
    p : BOOLEAN := TRUE;
  END_LOCAL;
  n := SIZEOF(a_path.edge_list);
  REPEAT i := 2 TO n BY 1;
    p := p AND (a_path.edge_list[i - 1].edge_end :=: a_path.edge_list[i]
        .edge_start);
  END_REPEAT;
  RETURN(p);

END_FUNCTION; -- path_head_to_tail

FUNCTION path_reversed(
             a_path: path
    ): oriented_path;

  LOCAL
    the_reverse : oriented_path;
```

```
  END_LOCAL;
  IF 'SHIP_ARRANGEMENT_SCHEMA.ORIENTED_PATH' IN TYPEOF(a_path) THEN
    the_reverse := (dummy_tri || path(list_of_topology_reversed(a_path.
        edge_list))) || oriented_path(a_path\oriented_path.path_element,
        NOT a_path\oriented_path.orientation);
  ELSE
    the_reverse := (dummy_tri || path(list_of_topology_reversed(a_path.
        edge_list))) || oriented_path(a_path,FALSE);
  END_IF;
  RETURN(the_reverse);

END_FUNCTION; -- path_reversed

FUNCTION scalar_times_vector(
             scalar: REAL;
             vec: vector_or_direction
    ): vector;

  LOCAL
    v      : direction;
    mag    : REAL;
    result : vector;
  END_LOCAL;
  IF (NOT EXISTS(scalar)) OR (NOT EXISTS(vec)) THEN
    RETURN(?);
  ELSE
    IF 'SHIP_ARRANGEMENT_SCHEMA.VECTOR' IN TYPEOF(vec) THEN
      v := dummy_gri || direction(vec.orientation.direction_ratios);
      mag := scalar * vec.magnitude;
    ELSE
      v := dummy_gri || direction(vec.direction_ratios);
      mag := scalar;
    END_IF;
    IF mag < 0 THEN
      REPEAT i := 1 TO SIZEOF(v.direction_ratios) BY 1;
        v.direction_ratios[i] := -v.direction_ratios[i];
      END_REPEAT;
      mag := -mag;
    END_IF;
    result := dummy_gri || vector(normalise(v),mag);
  END_IF;
  RETURN(result);

END_FUNCTION; -- scalar_times_vector

FUNCTION second_proj_axis(
             z_axis, x_axis, arg: direction
    ): direction;

  LOCAL
    temp   : vector;
    v      : direction;
    y_axis : vector;
  END_LOCAL;
  IF NOT EXISTS(arg) THEN
    v := dummy_gri || direction([0,1,0]);
  ELSE
    v := arg;
  END_IF;
  temp := scalar_times_vector(dot_product(v,z_axis),z_axis);
  y_axis := vector_difference(v,temp);
  temp := scalar_times_vector(dot_product(v,x_axis),x_axis);
  y_axis := vector_difference(y_axis,temp);
  y_axis := normalise(y_axis);
```



```
  RETURN(y_axis.orientation);

END_FUNCTION; -- second_proj_axis

FUNCTION set_of_topology_reversed(
             a_set: set_of_reversible_topology_item
    ): set_of_reversible_topology_item;

  LOCAL
    the_reverse : set_of_reversible_topology_item;
  END_LOCAL;
  the_reverse := [];
  REPEAT i := 1 TO SIZEOF(a_set) BY 1;
    the_reverse := the_reverse + topology_reversed(a_set[i]);
  END_REPEAT;
  RETURN(the_reverse);

END_FUNCTION; -- set_of_topology_reversed

FUNCTION shell_reversed(
             a_shell: shell
    ): shell;
  IF 'SHIP_ARRANGEMENT_SCHEMA.OPEN_SHELL' IN TYPEOF(a_shell) THEN
    RETURN(open_shell_reversed(a_shell));
  ELSE
    IF 'SHIP_ARRANGEMENT_SCHEMA.CLOSED_SHELL' IN TYPEOF(a_shell) THEN
      RETURN(closed_shell_reversed(a_shell));
    ELSE
      RETURN(?);
    END_IF;
  END_IF;

END_FUNCTION; -- shell_reversed

FUNCTION surface_weights_positive(
             b: rational_b_spline_surface
    ): BOOLEAN;

  LOCAL
    result : BOOLEAN := TRUE;
  END_LOCAL;
  REPEAT i := 0 TO b.u_upper BY 1;
    REPEAT j := 0 TO b.v_upper BY 1;
      IF b.weights[i][j] <= 0 THEN
        result := FALSE;
        RETURN(result);
      END_IF;
    END_REPEAT;
  END_REPEAT;
  RETURN(result);

END_FUNCTION; -- surface_weights_positive

FUNCTION topology_reversed(
             an_item: reversible_topology
    ): reversible_topology;
  IF 'SHIP_ARRANGEMENT_SCHEMA.EDGE' IN TYPEOF(an_item) THEN
    RETURN(edge_reversed(an_item));
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.PATH' IN TYPEOF(an_item) THEN
    RETURN(path_reversed(an_item));
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.FACE_BOUND' IN TYPEOF(an_item) THEN
    RETURN(face_bound_reversed(an_item));
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.FACE' IN TYPEOF(an_item) THEN
```

```
    RETURN(face_reversed(an_item));
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.SHELL' IN TYPEOF(an_item) THEN
    RETURN(shell_reversed(an_item));
  END_IF;
  IF 'SET' IN TYPEOF(an_item) THEN
    RETURN(set_of_topology_reversed(an_item));
  END_IF;
  IF 'LIST' IN TYPEOF(an_item) THEN
    RETURN(list_of_topology_reversed(an_item));
  END_IF;
  RETURN(?);

END_FUNCTION; -- topology_reversed

FUNCTION using_items(
             item: founded_item_select;
             checked_items: SET OF founded_item_select
    ): SET OF founded_item_select;

  LOCAL
    next_items      : SET OF founded_item_select;
    new_check_items : SET OF founded_item_select;
    result_items    : SET OF founded_item_select;
  END_LOCAL;
  result_items := [];
  new_check_items := checked_items + item;
  next_items := QUERY ( z <* bag_to_set(USEDIN(item,'')) | ((
      'SHIP_ARRANGEMENT_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z)) OR (
      'SHIP_ARRANGEMENT_SCHEMA.FOUNDED_ITEM' IN TYPEOF(z))) );
  IF SIZEOF(next_items) > 0 THEN
    REPEAT i := 1 TO HIINDEX(next_items) BY 1;
      IF NOT (next_items[i] IN new_check_items) THEN
        result_items := (result_items + next_items[i]) + using_items(
            next_items[i],new_check_items);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(result_items);

END_FUNCTION; -- using_items

FUNCTION using_representations(
             item: founded_item_select
    ): SET OF representation;

  LOCAL
    results            : SET OF representation;
    intermediate_items : SET OF founded_item_select;
    result_bag         : BAG OF representation;
  END_LOCAL;
  results := [];
  result_bag := USEDIN(item,
      'SHIP_ARRANGEMENT_SCHEMA.REPRESENTATION.ITEMS');
  IF SIZEOF(result_bag) > 0 THEN
    REPEAT i := 1 TO HIINDEX(result_bag) BY 1;
      results := results + result_bag[i];
    END_REPEAT;
  END_IF;
  intermediate_items := using_items(item,[]);
  IF SIZEOF(intermediate_items) > 0 THEN
    REPEAT i := 1 TO HIINDEX(intermediate_items) BY 1;
      result_bag := USEDIN(intermediate_items[i],
          'SHIP_ARRANGEMENT_SCHEMA.REPRESENTATION.ITEMS');
```



```
      IF SIZEOF(result_bag) > 0 THEN
        REPEAT j := 1 TO HIINDEX(result_bag) BY 1;
          results := results + result_bag[j];
        END_REPEAT;
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(results);

END_FUNCTION; -- using_representations

FUNCTION valid_calendar_date(
            date: calendar_date
    ): LOGICAL;
  CASE date.month_component OF
    1 :         RETURN((1 <= date.day_component) AND (date.day_component
          <= 31));
    2 :         BEGIN
        IF leap_year(date.year_component) THEN
          RETURN((1 <= date.day_component) AND (date.day_component <= 29));
        ELSE
          RETURN((1 <= date.day_component) AND (date.day_component <= 28));
        END_IF;
      END;
    3 :         RETURN((1 <= date.day_component) AND (date.day_component
          <= 31));
    4 :         RETURN((1 <= date.day_component) AND (date.day_component
          <= 30));
    5 :         RETURN((1 <= date.day_component) AND (date.day_component
          <= 31));
    6 :         RETURN((1 <= date.day_component) AND (date.day_component
          <= 30));
    7 :         RETURN((1 <= date.day_component) AND (date.day_component
          <= 31));
    8 :         RETURN((1 <= date.day_component) AND (date.day_component
          <= 31));
    9 :         RETURN((1 <= date.day_component) AND (date.day_component
          <= 30));
    10  :         RETURN((1 <= date.day_component) AND (date.
          day_component <= 31));
    11  :         RETURN((1 <= date.day_component) AND (date.
          day_component <= 30));
    12  :         RETURN((1 <= date.day_component) AND (date.
          day_component <= 31));
    END_CASE;
  RETURN(FALSE);

END_FUNCTION; -- valid_calendar_date

FUNCTION valid_measure_value(
            m: measure_value
    ): BOOLEAN;
  IF 'REAL' IN TYPEOF(m) THEN
    RETURN(m > 0);
  ELSE
    IF 'INTEGER' IN TYPEOF(m) THEN
      RETURN(m > 0);
    ELSE
      RETURN(TRUE);
    END_IF;
  END_IF;

END_FUNCTION; -- valid_measure_value

FUNCTION valid_time(
            time: local_time
```

```
    ): BOOLEAN;
  IF EXISTS(time.second_component) THEN
    RETURN(EXISTS(time.minute_component));
  ELSE
    RETURN(TRUE);
  END_IF;

END_FUNCTION; -- valid_time

FUNCTION valid_units(
             m: measure_with_unit
    ): BOOLEAN;
  IF 'SHIP_ARRANGEMENT_SCHEMA.LENGTH_MEASURE' IN TYPEOF(m.
      value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(1,0,0,0,0,0,0) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.MASS_MEASURE' IN TYPEOF(m.value_component)
       THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,1,0,0,0,0,0) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.TIME_MEASURE' IN TYPEOF(m.value_component)
       THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,1,0,0,0,0) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.ELECTRIC_CURRENT_MEASURE' IN TYPEOF(m.
      value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,1,0,0,0) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.THERMODYNAMIC_TEMPERATURE_MEASURE' IN
      TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,0,1,0,0) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.AMOUNT_OF_SUBSTANCE_MEASURE' IN TYPEOF(m.
      value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,0,0,1,0) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.LUMINOUS_INTENSITY_MEASURE' IN TYPEOF(m.
      value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,0,0,0,1) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.PLANE_ANGLE_MEASURE' IN TYPEOF(m.
      value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
```



```
        dimensional_exponents(0,0,0,0,0,0,0) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.SOLID_ANGLE_MEASURE' IN TYPEOF(m.
      value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,0,0,0,0) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.AREA_MEASURE' IN TYPEOF(m.value_component)
       THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(2,0,0,0,0,0,0) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.VOLUME_MEASURE' IN TYPEOF(m.
      value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(3,0,0,0,0,0,0) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.RATIO_MEASURE' IN TYPEOF(m.
      value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,0,0,0,0) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.POSITIVE_LENGTH_MEASURE' IN TYPEOF(m.
      value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(1,0,0,0,0,0,0) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  IF 'SHIP_ARRANGEMENT_SCHEMA.POSITIVE_PLANE_ANGLE_MEASURE' IN TYPEOF(m
      .value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,0,0,0,0) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  RETURN(TRUE);

END_FUNCTION; -- valid_units

FUNCTION vector_difference(
            arg1, arg2: vector_or_direction
    ): vector;

  LOCAL
    ndim   : INTEGER;
    mag2   : REAL;
    mag1   : REAL;
    mag    : REAL;
    res    : direction;
    vec1   : direction;
    vec2   : direction;
    result : vector;
  END_LOCAL;
  IF ((NOT EXISTS(arg1)) OR (NOT EXISTS(arg2))) OR (arg1.dim <> arg2.dim)
       THEN
```

```
    RETURN(?);
  ELSE
    BEGIN
      IF 'SHIP_ARRANGEMENT_SCHEMA.VECTOR' IN TYPEOF(arg1) THEN
        mag1 := arg1.magnitude;
        vec1 := arg1.orientation;
      ELSE
        mag1 := 1;
        vec1 := arg1;
      END_IF;
      IF 'SHIP_ARRANGEMENT_SCHEMA.VECTOR' IN TYPEOF(arg2) THEN
        mag2 := arg2.magnitude;
        vec2 := arg2.orientation;
      ELSE
        mag2 := 1;
        vec2 := arg2;
      END_IF;
      vec1 := normalise(vec1);
      vec2 := normalise(vec2);
      ndim := SIZEOF(vec1.direction_ratios);
      mag := 0;
      res := dummy_gri || direction(vec1.direction_ratios);
      REPEAT i := 1 TO ndim BY 1;
        res.direction_ratios[i] := (mag1 * vec1.direction_ratios[i]) + (
            mag2 * vec2.direction_ratios[i]);
        mag := mag + (res.direction_ratios[i] * res.direction_ratios[i]);
      END_REPEAT;
      IF mag > 0 THEN
        result := dummy_gri || vector(res,SQRT(mag));
      ELSE
        result := dummy_gri || vector(vec1,0);
      END_IF;
    END;
  END_IF;
  RETURN(result);

END_FUNCTION; -- vector_difference

FUNCTION which_class(
             t: GENERIC
    ): LIST OF STRING;

  LOCAL
    class_list : LIST OF STRING := [];
    elements   : BAG OF applied_classification_assignment;
  END_LOCAL;
  elements := USEDIN(t,
      'SHIP_ARRANGEMENT_SCHEMA.APPLIED_CLASSIFICATION_ASSIGNMENT.ITEMS');
  REPEAT i := 1 TO HIINDEX(elements) BY 1;
    IF elements[i]\classification_assignment.role.name =
        'class membership' THEN
      class_list := class_list + elements[i]\classification_assignment.
          assigned_class\group.name;
    END_IF;
  END_REPEAT;
  RETURN(class_list);

END_FUNCTION; -- which_class

END_SCHEMA; -- ship_arrangement_schema


(*
```


(Blank page)

Annex B

(normative)

AIM short names

Table B.1 provides the short names of entities specified in the AIM of this part of ISO 10303. Requirements on the use of the short names are found in the implementation methods included in ISO 10303.

Table B.1 — Short names

ACTION	ACTION
ACTION_ASSIGNMENT	ACTASS
ACTION_METHOD	ACTMTH
ACTION_REQUEST_ASSIGNMENT	ACRQAS
ACTION_REQUEST_SOLUTION	ACRQSL
ADDRESS	ADDRSS
ADVANCED_FACE	ADVFC
APPLICATION_CONTEXT	APPCNT
APPLICATION_CONTEXT_ELEMENT	APCNEL
APPLICATION_PROTOCOL_DEFINITION	APPRDF
APPLIED_ACTION_ASSIGNMENT	APACAS
APPLIED_ACTION_REQUEST_ASSIGNMENT	AARA
APPLIED_APPROVAL_ASSIGNMENT	APAPAS
APPLIED_CLASSIFICATION_ASSIGNMENT	APCLAS
APPLIED_DATE_AND_TIME_ASSIGNMENT	ADATA
APPLIED_DOCUMENT_REFERENCE	APDCRF
APPLIED_EFFECTIVITY_ASSIGNMENT	APEFAS
APPLIED_EXTERNAL_IDENTIFICATION_ASSIGNMENT	AEIA
APPLIED_GROUP_ASSIGNMENT	APGRAS
APPLIED_IDENTIFICATION_ASSIGNMENT	APIDAS
APPLIED_ORGANIZATION_ASSIGNMENT	APORAS

Table B.1 — Short names (continued)

APPLIED_PERSON_AND_ORGANIZATION_ASSIGNMENT	APAOA
APPLIED_PERSON_ASSIGNMENT	APPRAS
APPROVAL	APPRVL
APPROVAL_ASSIGNMENT	APPASS
APPROVAL_DATE_TIME	APDTTM
APPROVAL_PERSON_ORGANIZATION	APPROR
APPROVAL_ROLE	APPRL
APPROVAL_STATUS	APPSTT
AXIS1_PLACEMENT	AX1PLC
AXIS2_PLACEMENT_2D	A2PL2D
AXIS2_PLACEMENT_3D	A2PL3D
B_SPLINE_CURVE	BSPCR
B_SPLINE_CURVE_WITH_KNOTS	BSCWK
B_SPLINE_SURFACE	BSPSR
B_SPLINE_SURFACE_WITH_KNOTS	BSSWK
BEZIER_CURVE	BZRCRV
BEZIER_SURFACE	BZRSRF
BOUNDED_CURVE	BNDCRV
BOUNDED_PCURVE	BNDPCR
BOUNDED_SURFACE	BNDSRF
BOUNDED_SURFACE_CURVE	BNSRCR
CALENDAR_DATE	CLNDT
CARTESIAN_POINT	CRTPNT
CARTESIAN_TRANSFORMATION_OPERATOR	CRTROP
CARTESIAN_TRANSFORMATION_OPERATOR_3D	CTO3
CHARACTERIZED_OBJECT	CHROBJ
CIRCLE	CIRCLE

Table B.1 — Short names (continued)

CLASS	CLASS
CLASSIFICATION_ASSIGNMENT	CLSASS
CLASSIFICATION_ROLE	CLSRL
CLOSED_SHELL	CLSSHL
COMPOSITE_CURVE	CMPCRV
COMPOSITE_CURVE_ON_SURFACE	CCOS
COMPOSITE_CURVE_SEGMENT	CMCRSG
COMPOUND_REPRESENTATION_ITEM	CMRPIT
CONIC	CONIC
CONICAL_SURFACE	CNCSRF
CONNECTED_FACE_SET	CNFCST
CONTEXT_DEPENDENT_UNIT	CNDPUN
COORDINATED_UNIVERSAL_TIME_OFFSET	CUTO
CURVE	CURVE
CURVE_REPLICA	CRVRPL
CYLINDRICAL_SURFACE	CYLSRF
DATE	DATE
DATE_AND_TIME	DTANTM
DATE_AND_TIME_ASSIGNMENT	DATA
DATE_TIME_ROLE	DTTMRL
DEFINITIONAL_REPRESENTATION	DFNRPR
DEGENERATE_PCURVE	DGNPCR
DEGENERATE_TOROIDAL_SURFACE	DGTRSR
DERIVED_UNIT	DRVUNT
DERIVED_UNIT_ELEMENT	DRUNEL
DESCRIPTION_ATTRIBUTE	DSCATT
DESCRIPTIVE_REPRESENTATION_ITEM	DSRPIT

Table B.1 — Short names (continued)

DIMENSIONAL_EXPONENTS	DMNEXP
DIRECTION	DRCTN
DOCUMENT	DCMNT
DOCUMENT_REFERENCE	DCMRFR
DOCUMENT_REPRESENTATION_TYPE	DCRPTY
DOCUMENT_TYPE	DCMTYP
DOCUMENT_USAGE_CONSTRAINT	DCUSCN
EDGE	EDGE
EDGE_CURVE	EDGCRV
EDGE_LOOP	EDGLP
EFFECTIVITY	EFFCTV
EFFECTIVITY_ASSIGNMENT	EFFASS
ELEMENTARY_SURFACE	ELMSRF
ELLIPSE	ELLPS
EVALUATED_DEGENERATE_PCURVE	EVDGPC
EXECUTED_ACTION	EXCACT
EXTERNAL_IDENTIFICATION_ASSIGNMENT	EXIDAS
EXTERNAL_SOURCE	EXTSRC
EXTERNAL_SOURCE_RELATIONSHIP	EXSRRL
FACE	FACE
FACE_BASED_SURFACE_MODEL	FBSM
FACE_BOUND	FCBND
FACE_OUTER_BOUND	FCOTBN
FACE_SURFACE	FCSRF
FOUNDED_ITEM	FNDITM
FUNCTIONALLY_DEFINED_TRANSFORMATION	FNDFTR

Table B.1 — Short names (continued)

GEOMETRIC_REPRESENTATION_CONTEXT	GMRPCN
GEOMETRIC_REPRESENTATION_ITEM	GMRPIT
GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT	GC
GLOBAL_UNIT_ASSIGNED_CONTEXT	GUAC
GROUP	GROUP
GROUP_ASSIGNMENT	GRPASS
GROUP_RELATIONSHIP	GRPRLT
HYPERBOLA	HYPRBL
ID_ATTRIBUTE	IDATT
IDENTIFICATION_ASSIGNMENT	IDNASS
IDENTIFICATION_ASSIGNMENT_RELATIONSHIP	IDASRL
IDENTIFICATION_ROLE	IDNRL
INTERSECTION_CURVE	INTCRV
ITEM_DEFINED_TRANSFORMATION	ITDFTR
LENGTH_UNIT	LNGUNT
LINE	LINE
LOCAL_TIME	LCLTM
LOOP	LOOP
LUMINOUS_INTENSITY_UNIT	LMINUN
MAPPED_ITEM	MPPITM
MASS_UNIT	MSSUNT
MEASURE_WITH_UNIT	MSWTUN
NAME_ATTRIBUTE	NMATT
NAMED_UNIT	NMDUNT
NON_MANIFOLD_SURFACE_SHAPE_REPRESENTATION	NMSSR
OBJECT_ROLE	OBJRL
OFFSET_CURVE_3D	OFCR3D
OFFSET_SURFACE	OFFSRF

Table B.1 — Short names (continued)

OPEN_SHELL	OPNSHL
ORDINAL_DATE	ORDDT
ORGANIZATION	ORGNZT
ORGANIZATION_ASSIGNMENT	ORGASS
ORGANIZATION_ROLE	ORGRL
ORGANIZATIONAL_ADDRESS	ORGADD
ORGANIZATIONAL_PROJECT	ORGPRJ
ORIENTED_CLOSED_SHELL	ORCLSH
ORIENTED_EDGE	ORNEDG
ORIENTED_FACE	ORNFC
ORIENTED_OPEN_SHELL	OROPSH
ORIENTED_PATH	ORNPTH
ORIENTED_SURFACE	ORNSRF
PARABOLA	PRBL
PARAMETRIC_REPRESENTATION_CONTEXT	PRRPCN
PATH	PATH
PCURVE	PCURVE
PERSON	PERSON
PERSON_AND_ORGANIZATION	PRANOR
PERSON_AND_ORGANIZATION_ASSIGNMENT	PAOA
PERSON_AND_ORGANIZATION_ROLE	PAOR
PERSON_ASSIGNMENT	PRSASS
PERSON_ROLE	PRSRL
PERSONAL_ADDRESS	PRSADD
PLACEMENT	PLCMNT
PLANE	PLANE
PLANE_ANGLE_UNIT	PLANUN

Table B.1 — Short names (continued)

POINT	POINT
POINT_ON_CURVE	PNONCR
POINT_ON_SURFACE	PNONSR
POLY_LOOP	PLYLP
POLYLINE	PLYLN
PRODUCT	PRDCT
PRODUCT_CATEGORY	PRDCTG
PRODUCT_CONTEXT	PRDCNT
PRODUCT_DEFINITION	PRDDFN
PRODUCT_DEFINITION_CONTEXT	PRDFCN
PRODUCT_DEFINITION_FORMATION	PRDFFR
PRODUCT_DEFINITION_RELATIONSHIP	PRDFRL
PRODUCT_DEFINITION_SHAPE	PRDFSH
PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS	PDWAD
PRODUCT_RELATED_PRODUCT_CATEGORY	PRPC
PROPERTY_DEFINITION	PRPDFN
PROPERTY_DEFINITION_RELATIONSHIP	PRDFR
PROPERTY_DEFINITION_REPRESENTATION	PRDFRP
QUASI_UNIFORM_CURVE	QSUNCR
QUASI_UNIFORM_SURFACE	QSUNSR
RATIO_UNIT	RTUNT
RATIONAL_B_SPLINE_CURVE	RBSC
RATIONAL_B_SPLINE_SURFACE	RBSS
REPRESENTATION	RPRSNT
REPRESENTATION_CONTEXT	RPRCNT
REPRESENTATION_ITEM	RPRITM
REPRESENTATION_MAP	RPRMP
REPRESENTATION_RELATIONSHIP	RPRRLT

Table B.1 — Short names (concluded)

ROLE_ASSOCIATION	RLASS
SEAM_CURVE	SMCRV
SERIAL_NUMBERED_EFFECTIVITY	SRNMEF
SHAPE_ASPECT	SHPASP
SHAPE_DEFINITION_REPRESENTATION	SHDFRP
SHAPE_REPRESENTATION	SHPRPR
SI_UNIT	SUNT
SPHERICAL_SURFACE	SPHSRF
SURFACE	SRFC
SURFACE_CURVE	SRFCRV
SURFACE_OF_LINEAR_EXTRUSION	SL
SURFACE_OF_REVOLUTION	SROFRV
SURFACE_REPLICA	SRFRPL
SWEPT_SURFACE	SWPSRF
THERMODYNAMIC_TEMPERATURE_UNIT	THTMUN
TIME_UNIT	TMUNT
TOPOLOGICAL_REPRESENTATION_ITEM	TPRPIT
TOROIDAL_SURFACE	TRDSRF
UNCERTAINTY_MEASURE_WITH_UNIT	UMWU
UNIFORM_CURVE	UNFCRV
UNIFORM_SURFACE	UNFSRF
VALUE_REPRESENTATION_ITEM	VLRPIT
VECTOR	VECTOR
VERSIONED_ACTION_REQUEST	VRACRQ
VERTEX	VERTEX
VERTEX_LOOP	VRTLP
VERTEX_POINT	VRTPNT
WEEK_OF_YEAR_AND_DAY_DATE	WOYADD

Annex C

(normative)

Implemenation method specific requirements

The implementation method defines what types of exchange behavior are required with respect to this part of ISO 10303. Conformance to this part of ISO 10303 shall be realized in an exchange structure. The file format shall be encoded according to the syntax and EXPRESS language mapping defined in either ISO 10303-21 or ISO 10303-22 and in the AIM defined in annex A of this part of ISO 10303. The header exchange structure shall identify use of this part of ISO 10303 by the schema name ship_arrangement_schema.

Annex D

(normative)

Protocol implementation conformance statement (PICS) proforma

This clause lists the optional elements of this part of ISO 10303. An implementation may choose to support any combination of these optional elements. However, certain combinations of options are likely to be implemented together. These combinations are called conformance classes and are described in this annex.

This annex is in the form of a questionnaire. This questionnaire is intended to be filled out by the implementor and may be used in preparation for conformance testing by a testing laboratory. The completed PICS proforma is referred to as a PICS.

Four conformance classes are identified in this part of ISO 10303. A conforming implementation shall support at least one conformance class. Each class specifies a subset of ISO 10303-215 AIM constructs. These classes are detailed in clause 6 of ISO 10303-215.

Questions:

1. Please provide an identifier for the product or system for which conformance is claimed:

Product name and current version number:_____

2. Please indicate the implementation method chosen:

— ISO 10303-21 Exchange Structure -- preprocessor

— ISO 10303-22 SDAI -- preprocessor

Preprocessor name and current version number:_____

— ISO 10303-21 Exchange Structure -- postprocessor

— ISO 10303-22 SDAI -- postprocessor

Postprocessor name and current version number:_____

3. Please indicate the classes for which conformance is claimed:

— Class 1: Support for exchange of early design data

— Class 2: Support for exchange of detail design CAD geometry data

— Class 3: Support for exchange of detail design data

— Class 4: Support for exchange of operational data

— Class 5: Support for exchange of analysis data

Annex E

(normative)

Information object registration

E.1 Document identification

To provide for unambiguous identification of an information object in an open system, the object identifier

```
{ iso standard 10303 part(215) version(1) }
```

is assigned to this part of ISO 10303. The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

E.2 Schema identification

To provide for unambiguous identification of the schema-name in an open information system, the object identifier

```
{ iso standard 10303 part(215) version(1) schema(1) ship-arrangement-
                                schema(1) }
```

is assigned to the ship_arrangement_schema expanded schema (see annex A). The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

To provide for unambiguous identification of the schema-name in an open information system, the object identifier

```
{ iso standard 10303 part(215) version(1) schema(1) ship-arrangement-
                                schema(2) }
```

is assigned to the ship_arrangement_schema short form schema (see 5.2). The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

Annex F

(informative)

Application activity model

The application activity model (AAM) is provided to aid in understanding the scope and information requirements defined in this application protocol. The model is presented as a set of figures that contain the activity diagrams and a set of definitions of the activities and their data. The application activity model diagrams are given in Figures F.1 through F.16. Activities and data flows that are out of scope are marked with an asterisk.

NOTE The viewpoint of the application activity model is of an observer of the global ship development process. This activity model identifies the life cycle activities across all shipbuilding APs with extensions and emphasis appropriate to the ship arrangement. Activities related to the shipbuilding lifecycle that are not expanded in this activity model are detailed in other shipbuilding application protocols.

F.1 Application activity model definitions

The following terms are used in the application activity model. Terms marked with an asterisk are outside the scope of this application protocol.

The definitions in this annex do not supersede the definitions given in the main body of the text.

F.1.1 approve general arrangements: This is the top level activity for the approval of the general arrangements. It is the entry activity for both the Design Approval Preview and checking against rules and regulations. The ship is not certified by this activity alone.

F.1.2 approved design: The approved design is the final design to be submitted as an offer.

F.1.3 arrangements: The arrangements of the ship are the ship's compartments and spaces. Any description of arrangements will include associated definitions of purpose for the compartment or space.

F.1.4 availability, reliability and maintainability information*: The information about the components that is required to install them in the ship and is required for planned maintenance.

F.1.5 basic hull parameters: Estimated principal dimensions based on historical data or preliminary design development.

F.1.6 calculate capacities: This activity includes the calculation of capacities of compartments and holds such as underdeck space, bunker space, tanks, machinery room and double bottom peak.

F.1.7 calculate capacities, holds, bunker space: Calculation of all separate capacities. This could be done with the help of integral calculus or approximate formulae. For instance the hold capacity could be calculated from sectional areas and the integration over space's length.

F.1.8 calculate cost of ship*: This activity describes creation of negotiating documents based on technical product data and their estimated manufacturing cost. The results of this activity may contain sale price documents, financing support plan and documents describing funding and possible loans.

F.1.9 calculate lightship weight: This activity is necessary to summarise all relevant weight components. Together with the deadweight it is relevant for estimating the displacement.

F.1.10 calculate stability and trim: This activity deals with stability calculations (intact and damage stability), trim calculations, and calculations of centres of gravity in consideration of loading conditions.

F.1.11 calculate tonnage, freeboard: This activity deals with the calculation of tonnage and freeboard. As a result of the freeboard calculation a portion of ship volumes will be defined as reserve volumes.

F.1.12 calculate trim: This task involves the calculation of trim due to the weight of the ship and the weight and distribution of cargo.

F.1.13 calculate underdeck space: The calculation of all internal volumes.

F.1.14 cargo weights: The cargo weights used in defining loading conditions.

F.1.15 certificates*: The certificates issued by the classification society on completing the ship.

F.1.16 check arrangements for dangerous cargo: This activity checks for compliance with rule requirements with respect to arrangements for dangerous cargo (fire protection, detection, extinction, extinguisher).

F.1.17 check cofferdams and tank content: This activity checks the necessity for separating tanks from each other by cofferdams based on tank contents.

F.1.18 check design against rules and regulations: This is the top level activity for the approval of the primary design as part of the approval and certification process. The content of this activity is the same for all ships when it comes to conformance with Main Class Rules, but varies when it comes to additional class rules (type of vessel) and register notations. The activities performed are tailored to the rule requirements for general arrangement and global strength. This part of the approval is necessary before the yard can start ordering steel.

F.1.19 check internal doors and hatches for WT integrity: This activity checks for compliance with rule requirements with respect to doors and hatches and watertight integrity.

F.1.20 check position bulkheads: The checking of watertight integrity arrangements and stability conditions (intact and damage stability) to meet the relevant regulations given by IMO IA701E and IMO IC110E.

F.1.21 check stability (intact, damage): This activity includes the calculation of intact stability and damage stability. For the damage stability it is necessary to prove the buoyancy in damage conditions with the help of flooding curves (floodable and permissible length). The study of stability with calculation of different load conditions and damage conditions is necessary (i.e. lightship displacement and operation displacement).

F.1.22 classification society: An organization that enhances the safety of life and property at sea by providing rules, regulations and personnel for assessing and classifying ships during their lifecycle.

F.1.23 cog and lightship weight: Summarise all centres of gravity and all weight components relevant for lightship weight.

F.1.24 complete and approve design of machinery*: The selection, arrangement and approval of the power plant in terms of the main engine, associated propulsion system and its auxiliary machinery.

F.1.25 complete and approve design of outfitting and distribution systems*: The selection and

approval of the necessary outfitting equipment. The selection is based mainly on former designs and in accordance with the requirements. It also contains the layout of the different types of distribution systems such as piping and HVAC.

F.1.26 complete and approve design of ship structure*: The completion and approval of the ship structural design.

F.1.27 complete and approve ship design: The production and approval of ship design product data, documents and the classification drawings using the preliminary design from the bid preparation, as well as the required rules and regulations. The result of this activity is the approved design and the production and delivery schedule.

F.1.28 consultants: Organizations that provide specific services to shipyards, ship owners and classification societies during the ship lifecycle.

F.1.29 contract: The contract is the output from the activity which involves placing the order for the ship. The contract is used as a constraint in subsequent activities such as final design and approval and production.

F.1.30 cost*: The calculated cost of the ship based on the cost of material and labour.

F.1.31 create preliminary design: All design activities relevant in a very preliminary stage of ship design in consideration of classification rules, national/international demands, shipyard constraints and owner requirements. The aim of this task is to make a shipyard offer.

F.1.32 create preliminary general arrangements: The activity that produces the preliminary compartmentation plans from the preliminary hull form definition.

F.1.33 create preliminary hull form: The activity that is the first step of designing a ship. Using parent ships main dimensions and form parameters one or more preliminary hull forms will be generated.

F.1.34 create preliminary machinery design*: The activity that produces the preliminary designs for the ship machinery; including the prime mover, shaft system, fuel system, power systems and cargo handling equipment.

F.1.35 create preliminary outfitting design*: The activity that produces the preliminary design for the ship's outfitting, including distributed systems, such as piping and electrical systems.

F.1.36 create preliminary structure design*: The activity that produces the preliminary steel structure design, including the arrangement of the primary structural members.

F.1.37 critical design areas: The areas requiring thorough investigation and conformity checking identified by the Design Approval Preview.

F.1.38 decide post-sales & maintenance support*: The activity that puts together the maintenance package for the ship. This is part of the tender document and includes the post sales support.

F.1.39 decommission and disassemble*: All activities relating to the last stage of the ship's lifecycle. It consists of the decommissioning and dismantling of the ship.

F.1.40 define compartments: This activity deals with a preliminary establishment of main parameters. Main particulars are length between perpendiculars, breadth, depth, draught, Deadweight, Displacement and block coefficient. Also form parameters will be established like prismatic coefficient, waterline coefficient, midship section coefficient and angle of entrance of waterline.

F.1.41 define loading conditions: This activity deals with the loading conditions and is necessary to ascertain the payload as a function of the available capacities.

F.1.42 design modifications: Comments and recommendations on the design (red-marking).

F.1.43 design schedule: Data that controls the time from the design phase to production.

F.1.44 distribution and outfitting design *: The design of the distribution systems (electrical and piping) and the outfitting.

F.1.45 estimate hydrodynamics and powering*: The activity that approximates hydrodynamic properties data calculations such as resistance, propulsion, seakeeping and manoeuvrability for the preliminary hull form.

F.1.46 estimate weight: This task is necessary for calculating the lightship weight and consists of the calculation of the hull steel weights, machinery weights and weights of outfitting and accommodation.

F.1.47 evaluate hull steel weights: This activity defines the estimated steel weight with the help of empirical values in a very preliminary stage of the design.

F.1.48 evaluate machinery weights: This activity defines all separate weights belonging to the machinery plant, including auxiliary equipment.

F.1.49 evaluate request & schedule bid*: This describes the activities of the shipyard when evaluating the inquiry of the ship owner for a new ship.

F.1.50 evaluate weights of outfitting and accommodation: This activity defines all separate weights belonging to the outfitting and accommodation.

F.1.51 feedback: The outputs from activities which then feed back and modify previous activities in the lifecycle on the current or subsequent ships.

F.1.52 final compartment design: Approved design at the completion of the design of the compartment.

F.1.53 finalise and approve general arrangements: The activity that details the general arrangement after having created a draft layout. The ship's systems are described by a compartment and access drawing showing the location, the access, and the size of the different compartments.

F.1.54 finalise and approve hull form: The activity in which the hull form is finalised from the preliminary design. The result is a final and approved hull form design.

F.1.55 finalise and approve hydrodynamics and powering*: This includes all relevant hydrodynamic calculations such as resistance, propulsion, seakeeping and manoeuvrability.

F.1.56 finalise capacities calculations: The activity which produces the final volumes and centres results for the final calculation of stability and trim.

F.1.57 finalise compartment definition: The activity which gives the definition of the ship's compartments.

F.1.58 finalise general arrangements: The activity in which the general arrangements are finalised from the preliminary design.

F.1.59 finalise production planning*: This produces outputs relating to the final construction

sequence, the material supply and the management of time and people.

F.1.60 finalise stability and trim calculation: This activity produces a finalised trim and stability parameter.

F.1.61 finalise weight estimation: Produces the final weights and centres of gravity for the calculation of the final stability and trim.

F.1.62 floodable curves: Used in the activities which define compartments to establish the main bulkhead positions.

F.1.63 freeboard: The freeboard is the distance from the waterline to the upper surface of the freeboard deck at side.

F.1.64 fuel consumption: A fuel consumption calculation is used to estimate the needs of capacities for fuel.

F.1.65 general arrangements: The space arrangement plan from the preliminary design stage.

F.1.66 historical data from previous designs: Data held by the shipyard or model basin on previous ship designs and used to estimate the hydrodynamics, powering requirements and sea-keeping.

F.1.67 hull form sections: The design of the hull moulded form at planar sections taken along the longitudinal axis of the ship.

F.1.68 hull moulded form: The definition of the shape of the hull of the ship, resulting from the addition of the aft-body, mid-body and fore-body definitions, which does not take into account the thickness of the material from which the hull is made.

F.1.69 hull steel weights: These outputs are the results of several calculation and design activities which result in an estimated weight of the steel structure making up the hull.

F.1.70 hydrodynamics & powering results*: The results of calculations and model basin tests. They contain resistance, propulsion, propeller performance, brake power, service speed, sea keeping and manoeuvrability data.

F.1.71 hydrostatics*: Hydrostatic properties are used in checking of ship's stability.

F.1.72 knowledge and experience: The previous experience and knowledge of companies involved throughout the ship lifecycle.

F.1.73 laws, rules and regulations: National laws, statutory regulations and classification society rules that are used to control the design, manufacture, operation, maintenance and scrapping of the ship.

F.1.74 list of required certificates*: The result of placing an order, this is the list supplied by the owner for certificate requirements.

F.1.75 machinery design*: The design drawings and electronic models of the ship mechanical systems. An output from the final design process.

F.1.76 machinery weights: These outputs are the results of several calculation and design activities which result in an estimated weight for all machinery.

F.1.77 manufacturing restrictions: A constraint on the ship construction and design processes governed by available technology and shipyard facilities.

F.1.78 material list*: The list of raw materials needed to manufacture the ship. A result of the final design process.

F.1.79 material allocation/ordering: The data describing the necessary material supply for production.

F.1.80 modifications from machinery: Modifications to the hydrodynamics and powering due to feedback from the preliminary machinery design.

F.1.81 modifications to hull form: Modifications to the hull shape due to feedback from hydrodynamics and powering results and the final design process.

F.1.82 offer: The result of the preliminary design process. It will contain the shipyard's data for producing the requested ship.

F.1.83 offer guidelines: The offer guidelines include the data necessary to make an unconditional offer to the ship owner

F.1.84 operate and maintain a ship*: The activity that describes the running and maintenance of the ship during its service lifetime.

F.1.85 operational information: Accumulated information during the operation phase of the ship used for maintenance and in the final scrapping stage.

F.1.86 outfitting weights: These outputs are the result of several calculation and design activities which result in an estimated weight for all outfitting systems and furnishings.

F.1.87 owner: The organization which requests, orders and takes delivery of the ship.

F.1.88 owner request, requirements: The requirements document that is submitted to the shipyard by the owner upon the invitation to tender.

F.1.89 payload: This output calculates the payload as a function of the available capacities.

F.1.90 perform DAP (Design Approval Preview): This is the top level activity for the approval preview of ship design. This activity is a feasibility study conducted by a classification society, in which the design is checked very roughly to detect critical areas for thorough investigation and conformity checking both as a design comment and to draw attention to specific areas during design approval. The content of this activity may vary with contract specifications and type of ship.

F.1.91 perform ship lifecycle: All of the lifecycle activities associated with a ship.

F.1.92 place order*: The owner places an order for a ship from the bids that have been submitted. From this a contract is awarded.

F.1.93 planned maintenance system: Data created during the final design process and used during the operation and maintenance of the ship.

F.1.94 position of collision bulkhead: The position of collision bulkhead for passenger ships is usually constrained by the requirements of IMO IC110E for passenger ships and other rule constraints for other types of vessels.

F.1.95 pre layout: The very initial layout of the ship which is produced during the bid evaluation stage and is the basis for the preliminary design.

F.1.96 preliminary design: The preliminary design is that which is completed in the phases leading up to the submission of the tender.

F.1.97 preliminary hull form: The definition of the hull form, as a result of the preliminary design process. Used in the offer documents and for preliminary compartment design, hydrodynamics and powering calculations.

F.1.98 preliminary machinery design*: The definition of the ship mechanical systems. Used early to estimate the noise, speed and vibration and to estimate the machinery weights.

F.1.99 preliminary machinery, structure and outfitting design: Feedback consisting of the preliminary designs for machinery, structure and outfitting and furnishing. This allows the creation of preliminary general arrangements.

F.1.100 preliminary outfitting design: The definition of the ship's outfitting and accommodation, resulting from the preliminary design process.

F.1.101 preliminary structure design: The definition of the preliminary ship structure during the preliminary design process.

F.1.102 prepare bid: This activity includes all activities of the yard regarding preparation and submission of the offer to the ship owner for the ship to be built.

F.1.103 present offer*: The activity concerned with presentation of the offer to build the ship to the prospective ship owner.

F.1.104 produce and inspect a ship: This activity includes high-level activities such as produce, monitor and inspect ship production. Inspect, means the controlling of all activities throughout the whole production life cycle of a ship.

F.1.105 product component information*: The technical data about the components that will be incorporated into the ship. These are taken into consideration when the preliminary designs are being made.

F.1.106 propeller design*: The design of the propeller or propulsor as a result of the hydrodynamics and powering calculations. The design controls some of the machinery design activity.

F.1.107 refined design: The final compartment definitions.

F.1.108 request a ship*: The first activities of a ship owner when intending to order a ship. Having definite ideas regarding appearance and functionality of the ship, the owner expresses these ideas in an inquiry to the shipyard.

F.1.109 request for production changes: Changes that are requested to the ship design as a result of production experience or difficulties with the realisation of the ship design.

F.1.110 resistance and shaft power: The result of the activity to estimate hydrodynamics and powering. Resistance and shaft power is a constraint on the creation of the preliminary hull form.

F.1.111 resources: The shipyard, classification society, and outside consultants.

F.1.112 resources allocation: A result of production planning.

F.1.113 schedule: The schedule is formed as a part of the final design process. It governs the timing of the production phases.

F.1.114 scrapping plan*: The document used to schedule the time and resources required to dismantle the ship.

F.1.115 ship product model data: The product data of the accumulated throughout its lifecycle. Because scrapping is part of the lifecycle the ship is not an output, only the documented information and knowledge about the ship survives.

F.1.116 ship weight modifications: Modifications to ship weight due to the preliminary structure design. This is fed back to modify the preliminary hull form and revise the preliminary general arrangements.

F.1.117 shipyard: An organization that designs, builds, maintains, and repairs ships.

F.1.118 specify ship: All activities associated with the production of a detailed specification of the ship prior to a contract being placed.

F.1.119 stability parameter: Parameters including several results of stability calculations.

F.1.120 structural design*: The design of the hull structure including hull, bulkheads, decks and stiffeners.

F.1.121 technical requirements: The owner's specifications that must be realised by the completed ship.

F.1.122 technical documentation: In case of maintenance the technical documentation of a system means part of the product description required to perform preventative maintenance, repair and failure analysis of that system. Technical information is an output which includes more detail information about material parts needed for producing the ship/system.

F.1.123 transportation need: A constraint which determines the specification for the ship construction.

F.1.124 tonnage: Tonnage is a method of volume calculation applied to ships.

F.1.125 trim: The expected floating position of the ship resulting from calculation of the weights and their distribution throughout the ship.

F.1.126 volumes and centres: Volumes and centres of holds, bunkers, tanks and compartments.

F.1.127 weights and centres of gravity: Weights and centres of gravity necessary for further calculations.

F.1.128 weight distribution*: The details of the weight distribution taking into account steel weight, machinery weights, outfitting weights and cargo.

F.1.129 workload*: The total effort required to build the chosen ship design as estimated by the shipyard and assisting consultants.

F.2 Application activity model diagrams

The application activity model diagrams are given in Figures F.2 through F.16. The graphical form of the application activity model is presented in the IDEF0 activity modelling format [1]. Activities and data flows that are out of scope are marked with asterisks.

Figure F.1 describes the basic notation used in IDEF0 modelling. Each activity may be decomposed to provide more detail. If an activity has been decomposed, a separate diagram is included.

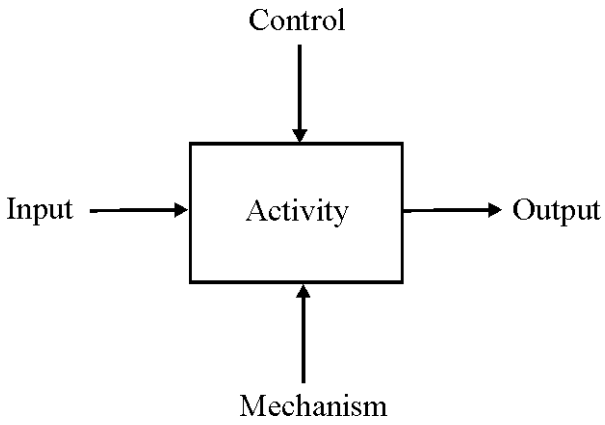

Figure F.1 — IDEF0 basic notation

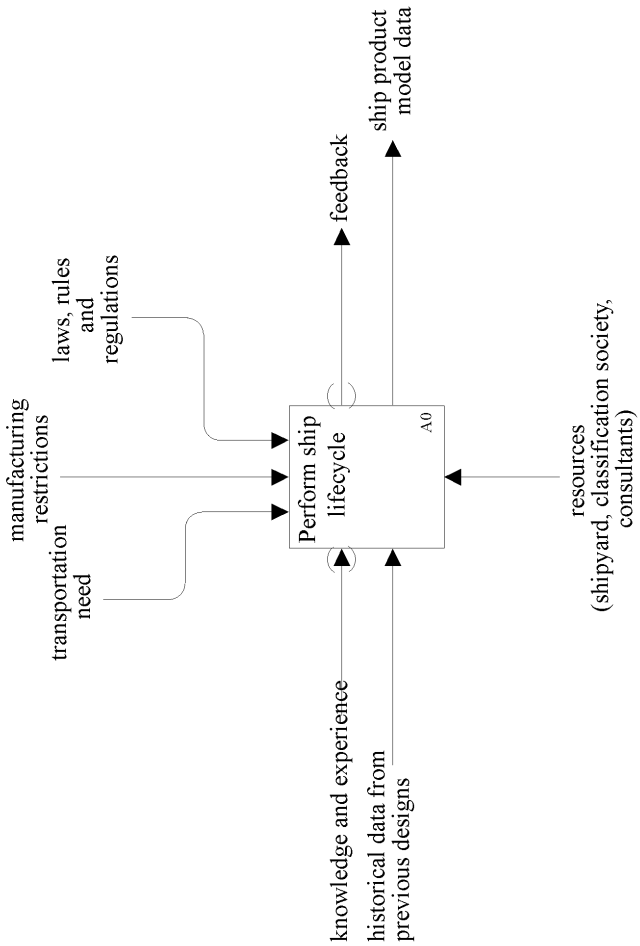

Figure F.2 — A-0 Ship arrangement AAM

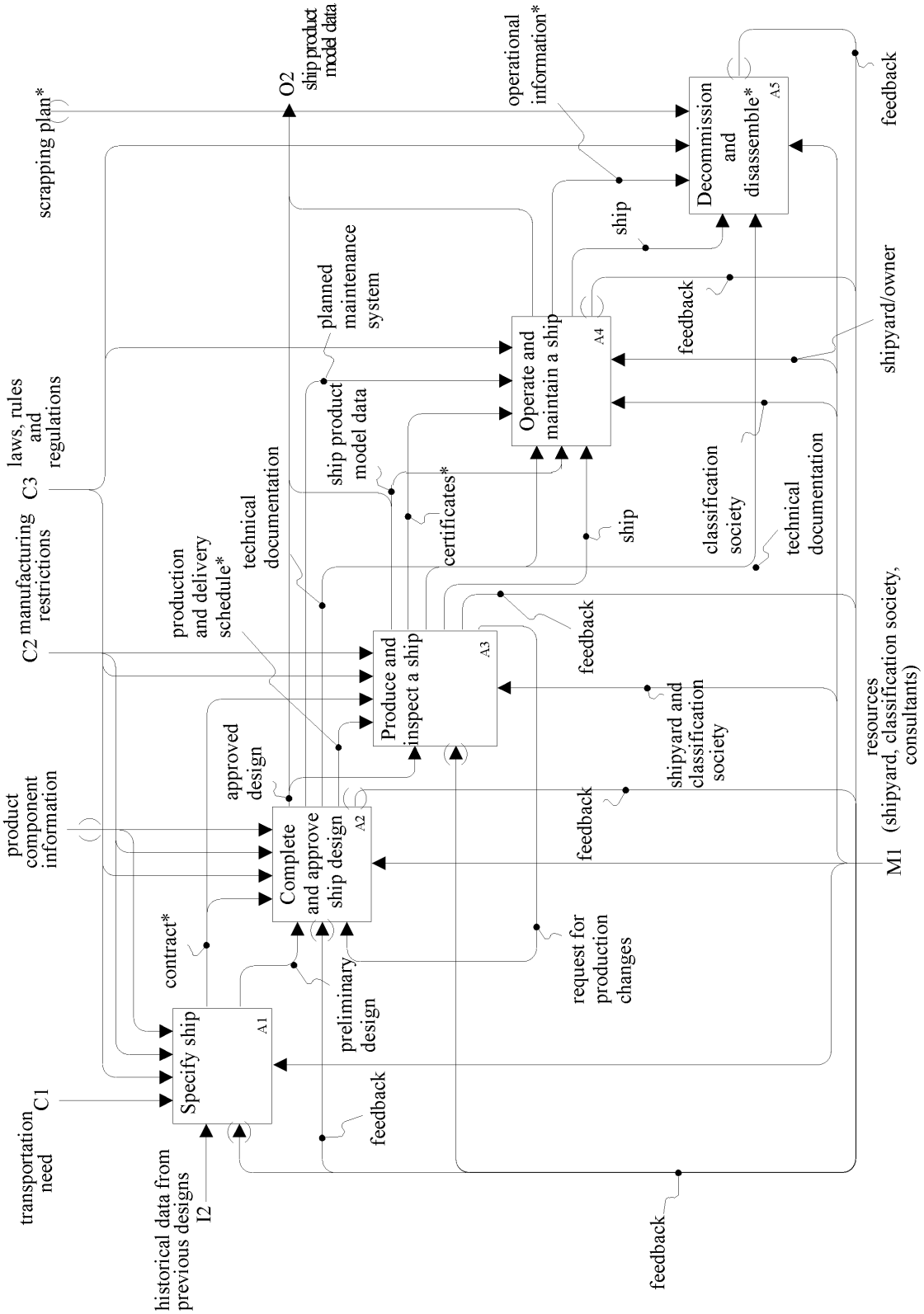

Figure F.3 — A0 Perform ship life cycle

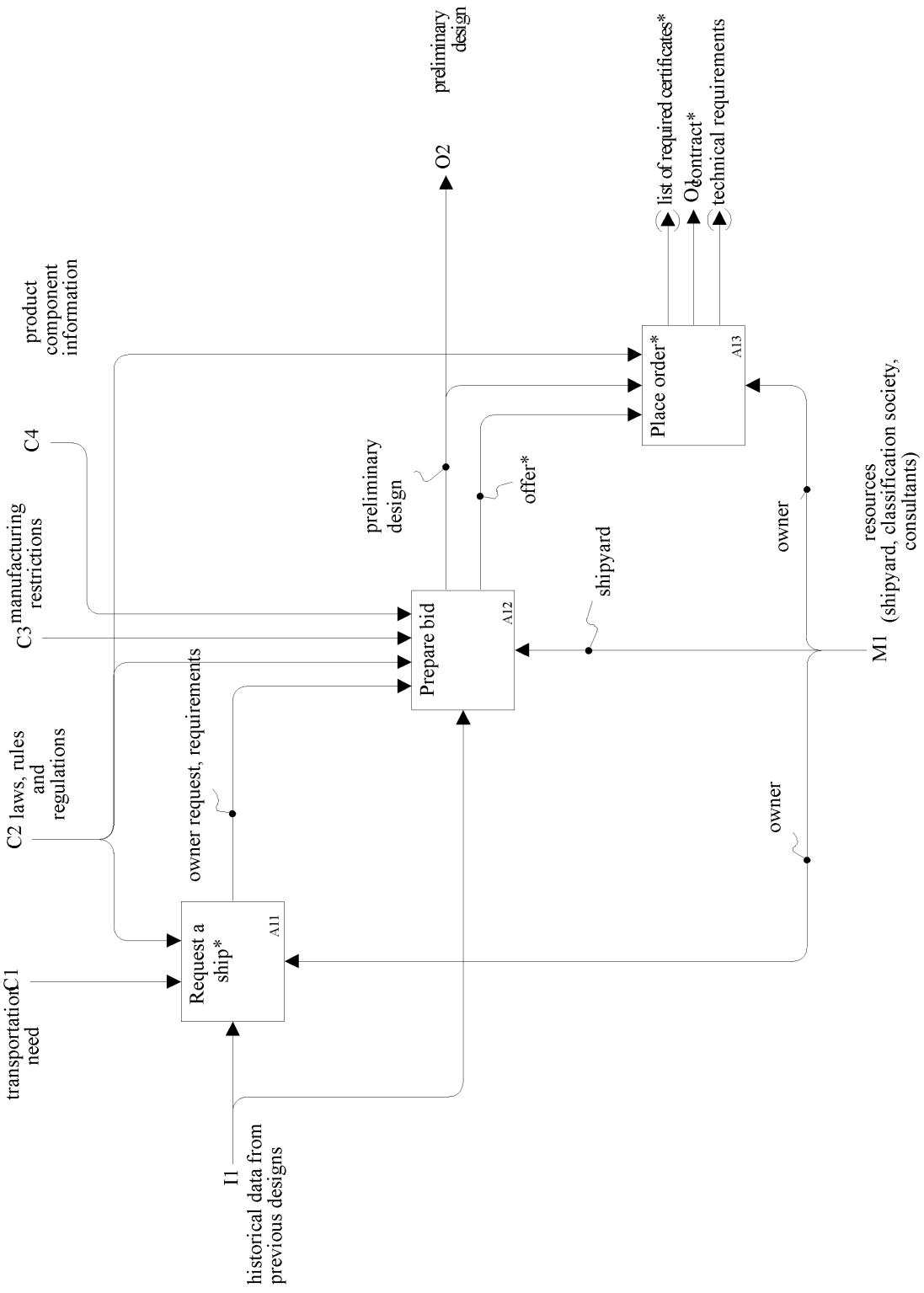

Figure F.4 — A1 Specify ship

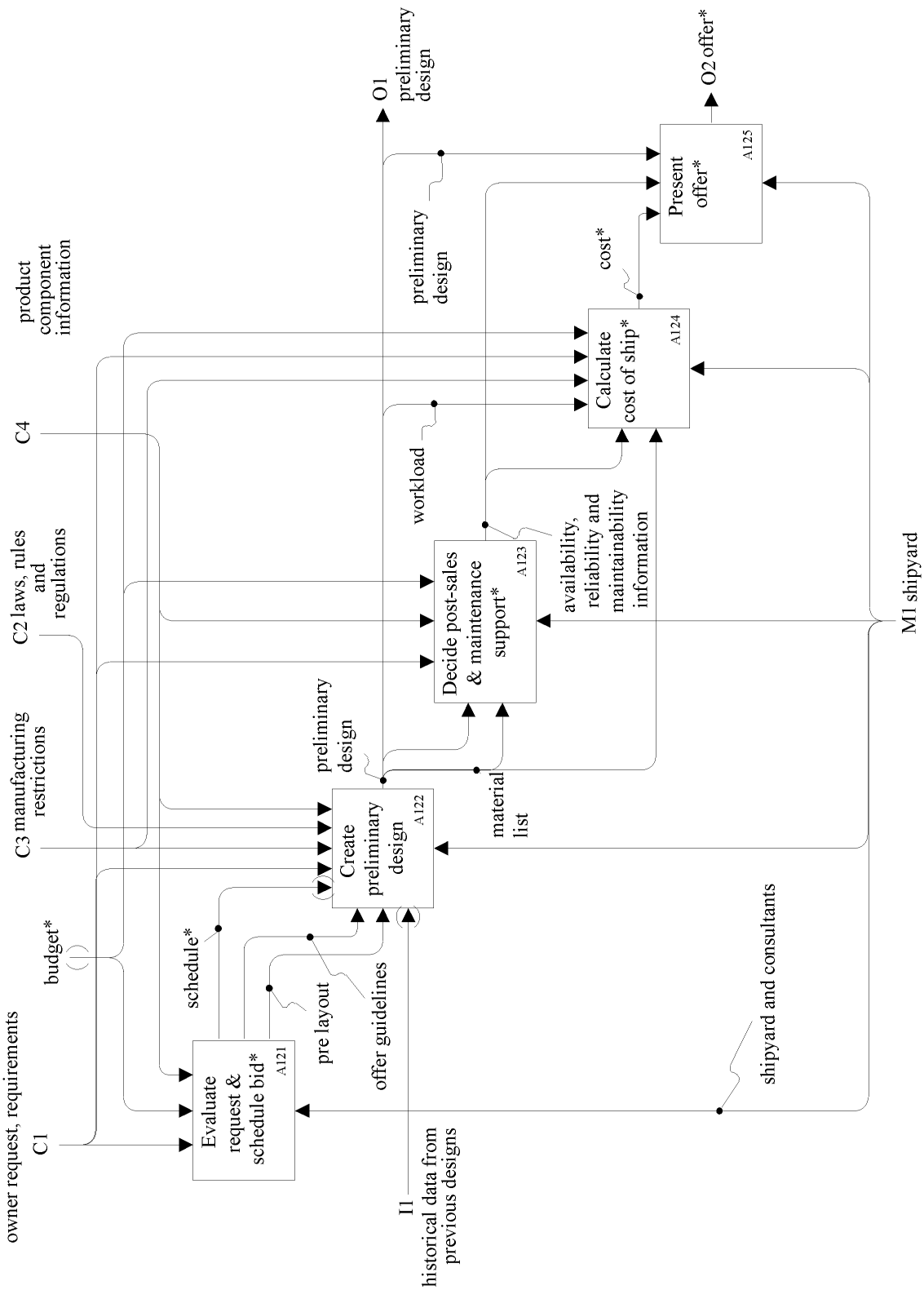

Figure F.5 — A12 Prepare bid

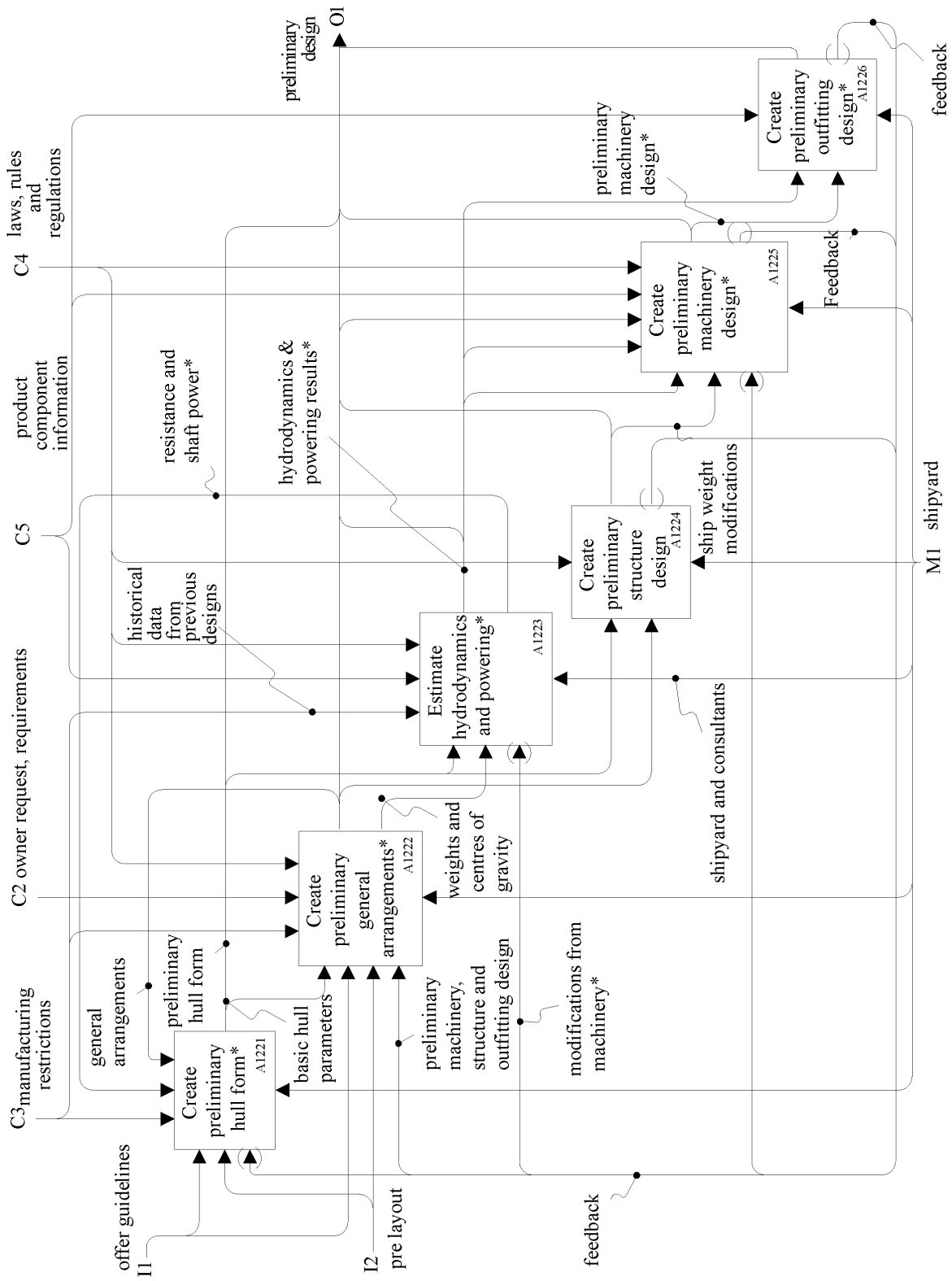

Figure F.6 — A122 Create preliminary design

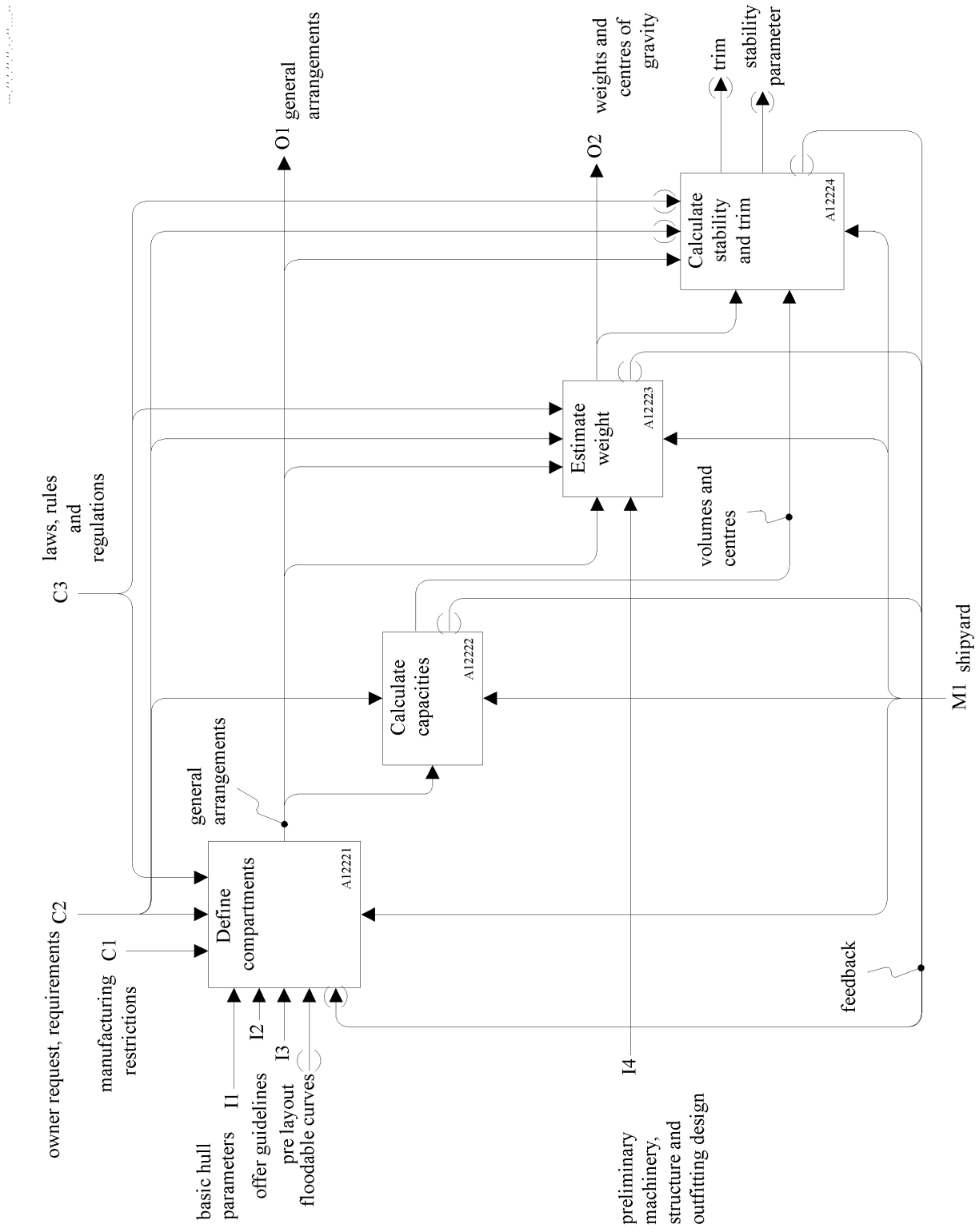

Figure F.7 — A1222 Create preliminary general arrangements

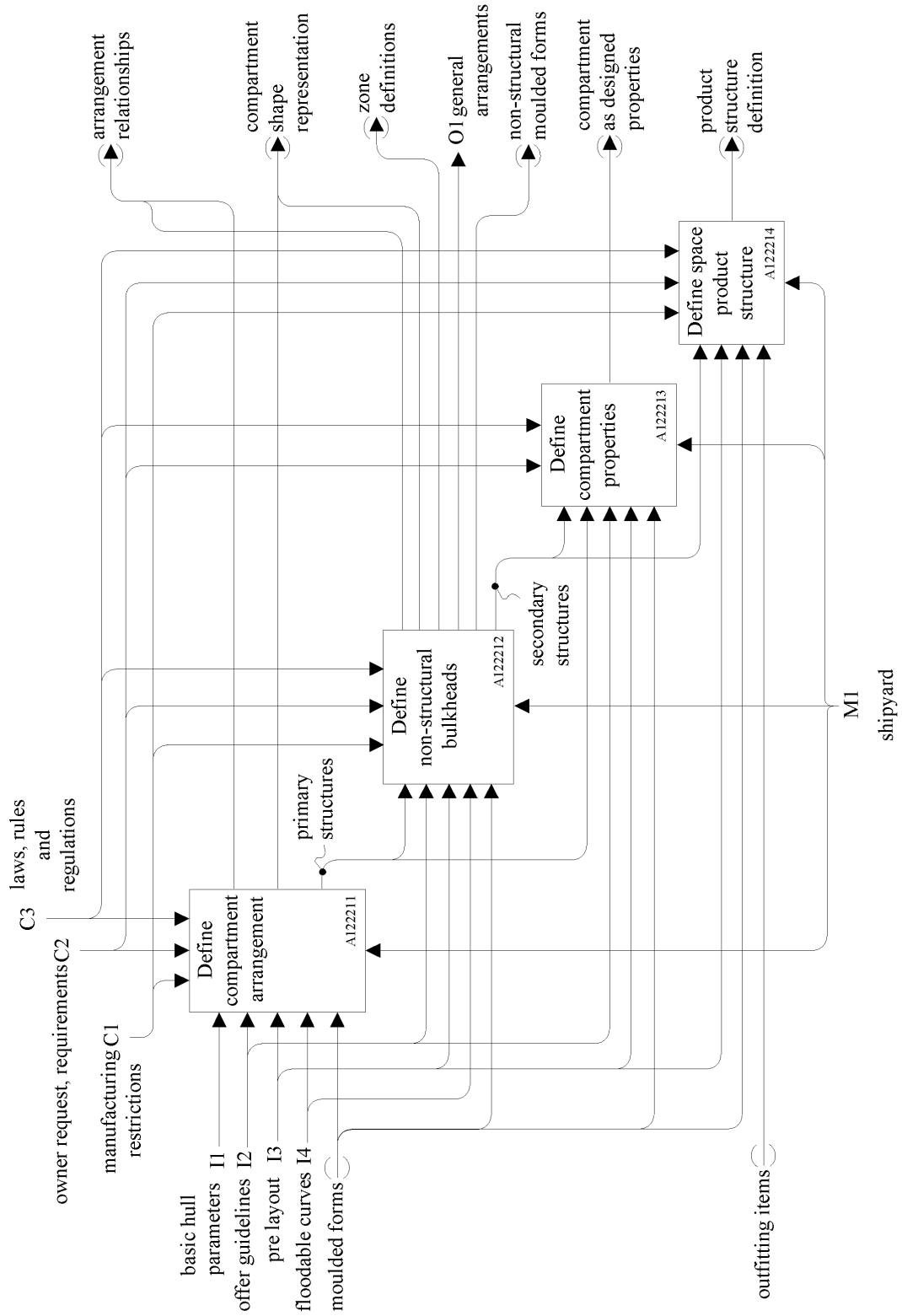

Figure F.8 — A12221 Define compartments

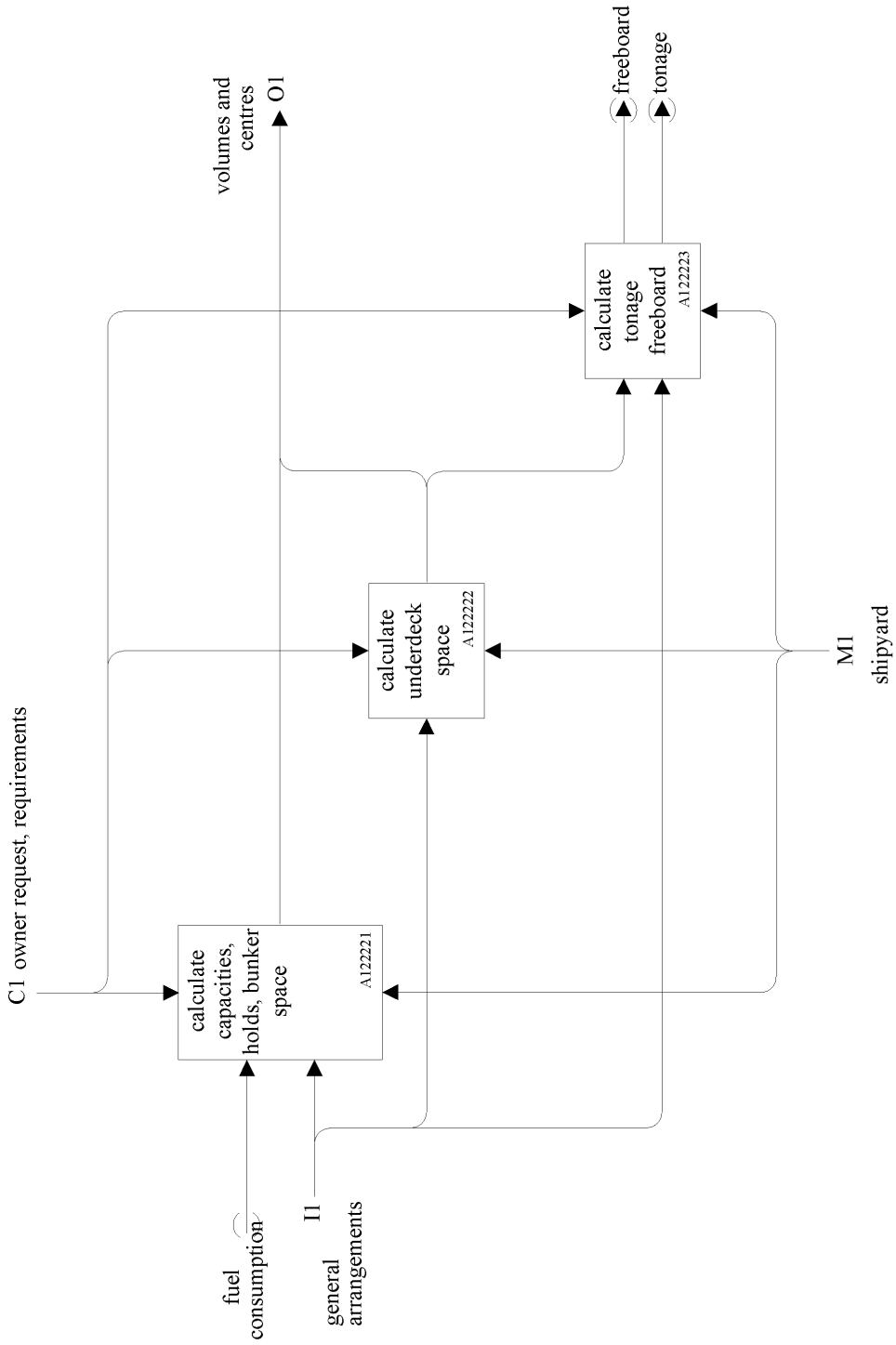

Figure F.9 — A12222 Calculate capacities

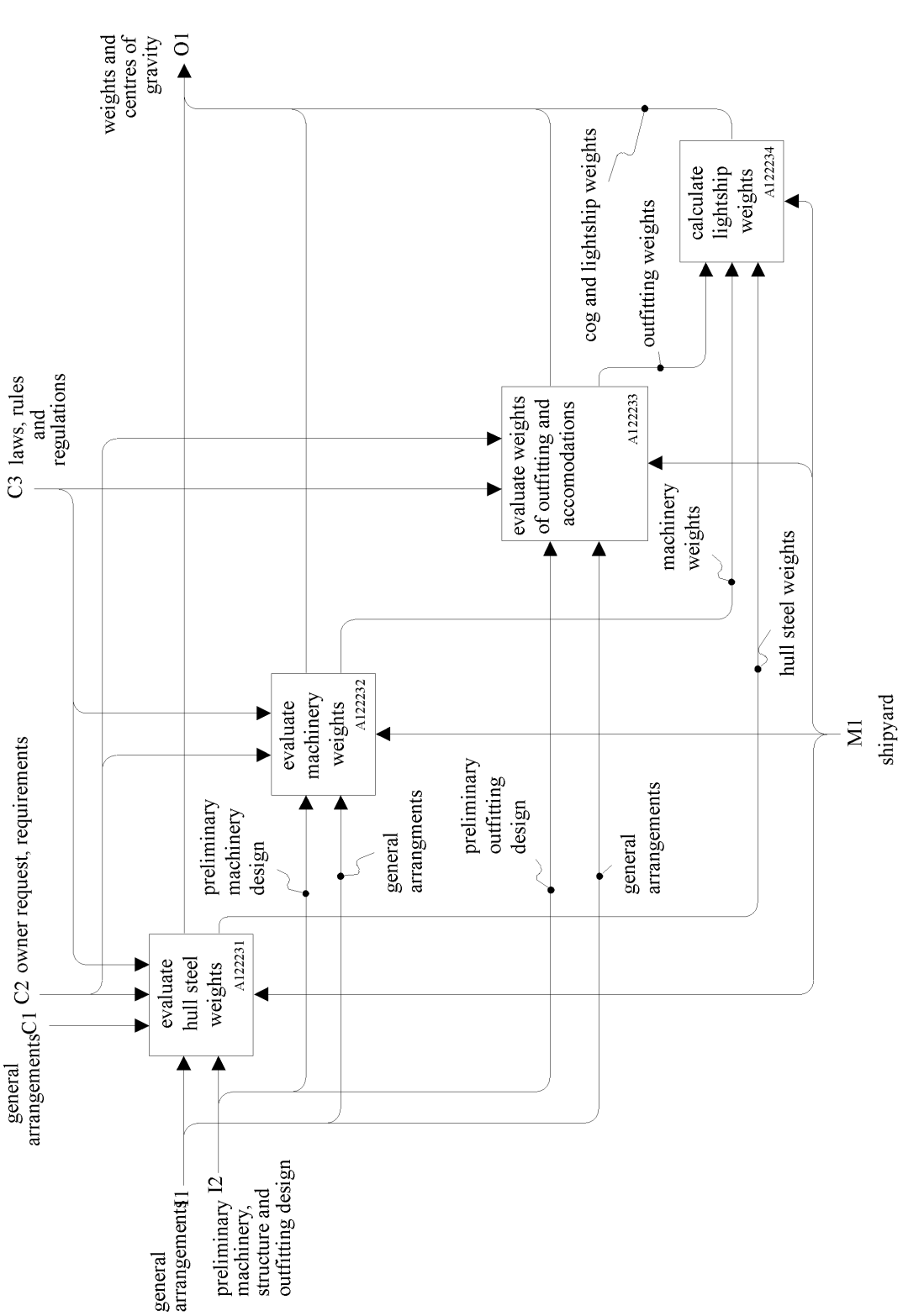

Figure F.10 — A12223 Estimate weight

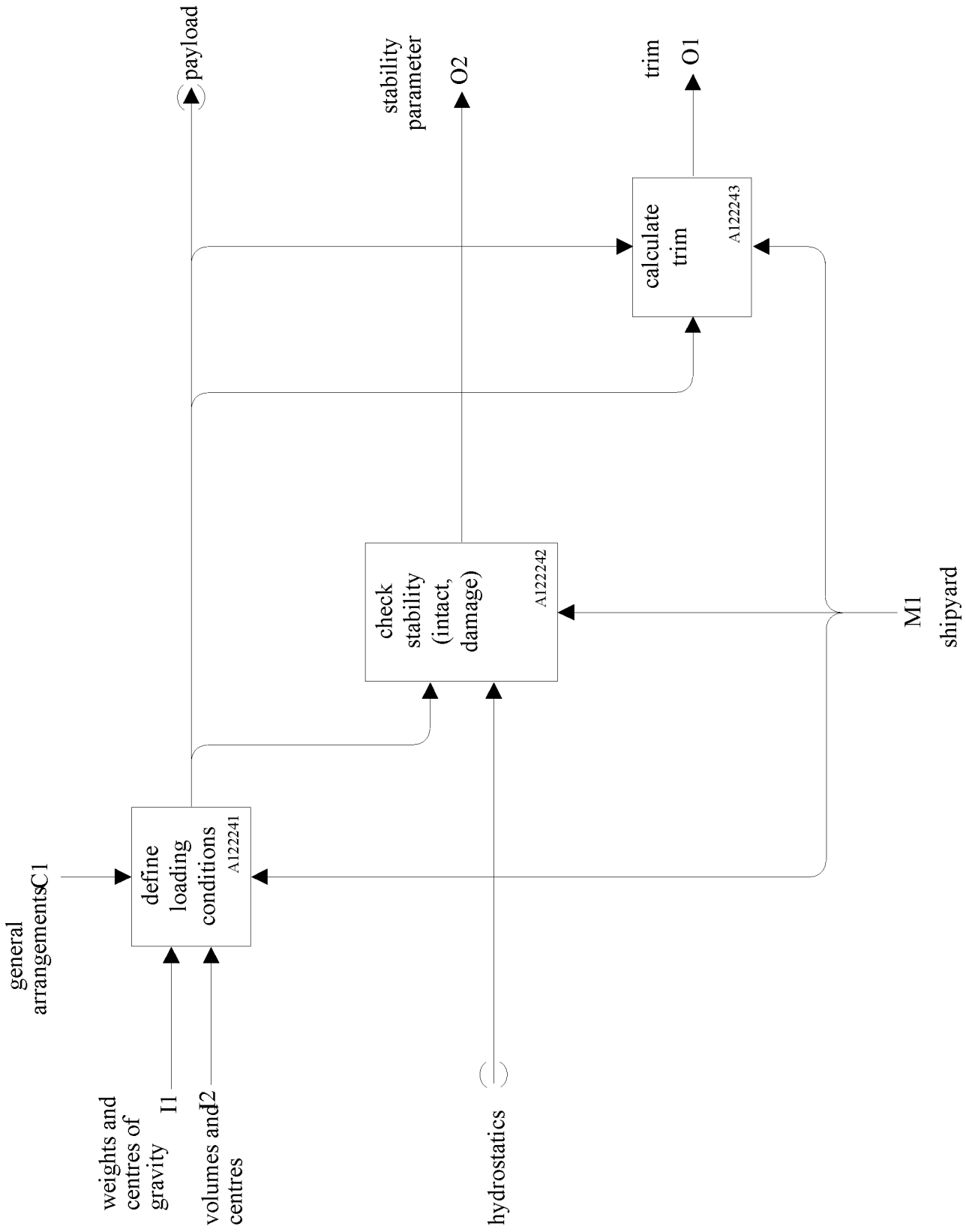

Figure F.11 — A12224 Calculate stability and trim

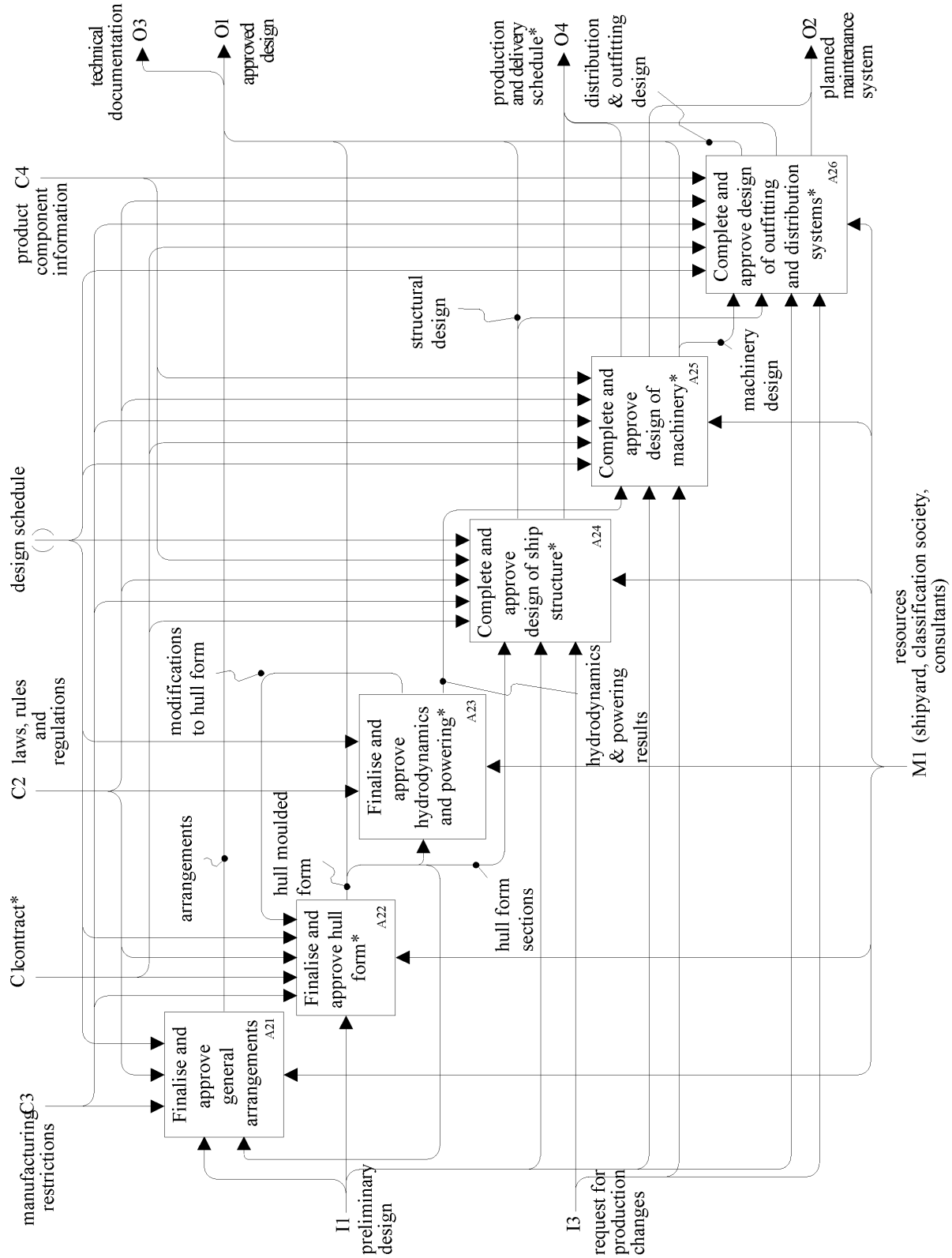

Figure F.12 — A2 Complete and approve ship design

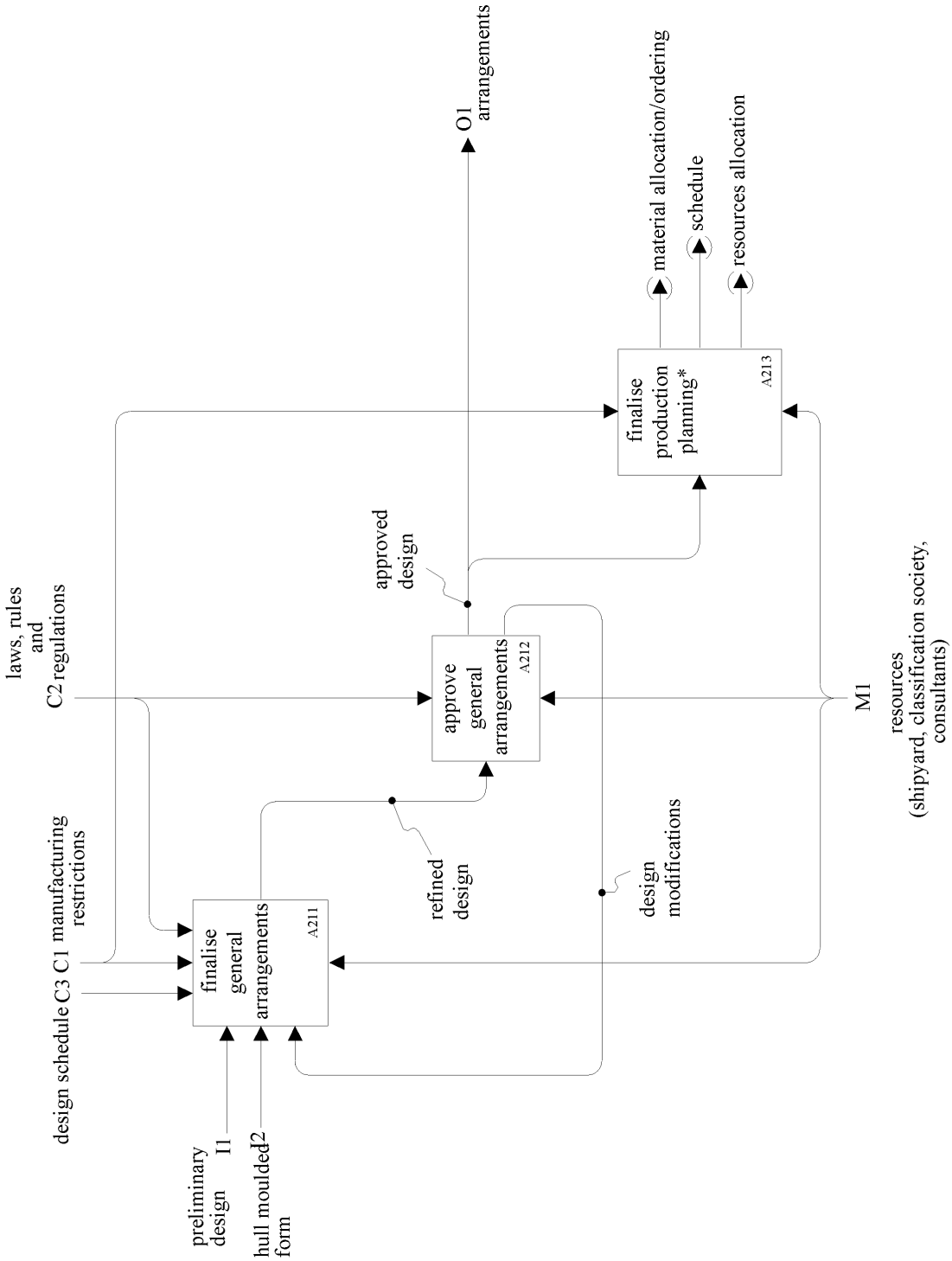

Figure F.13 — A21 Finalise and approve general arrangements

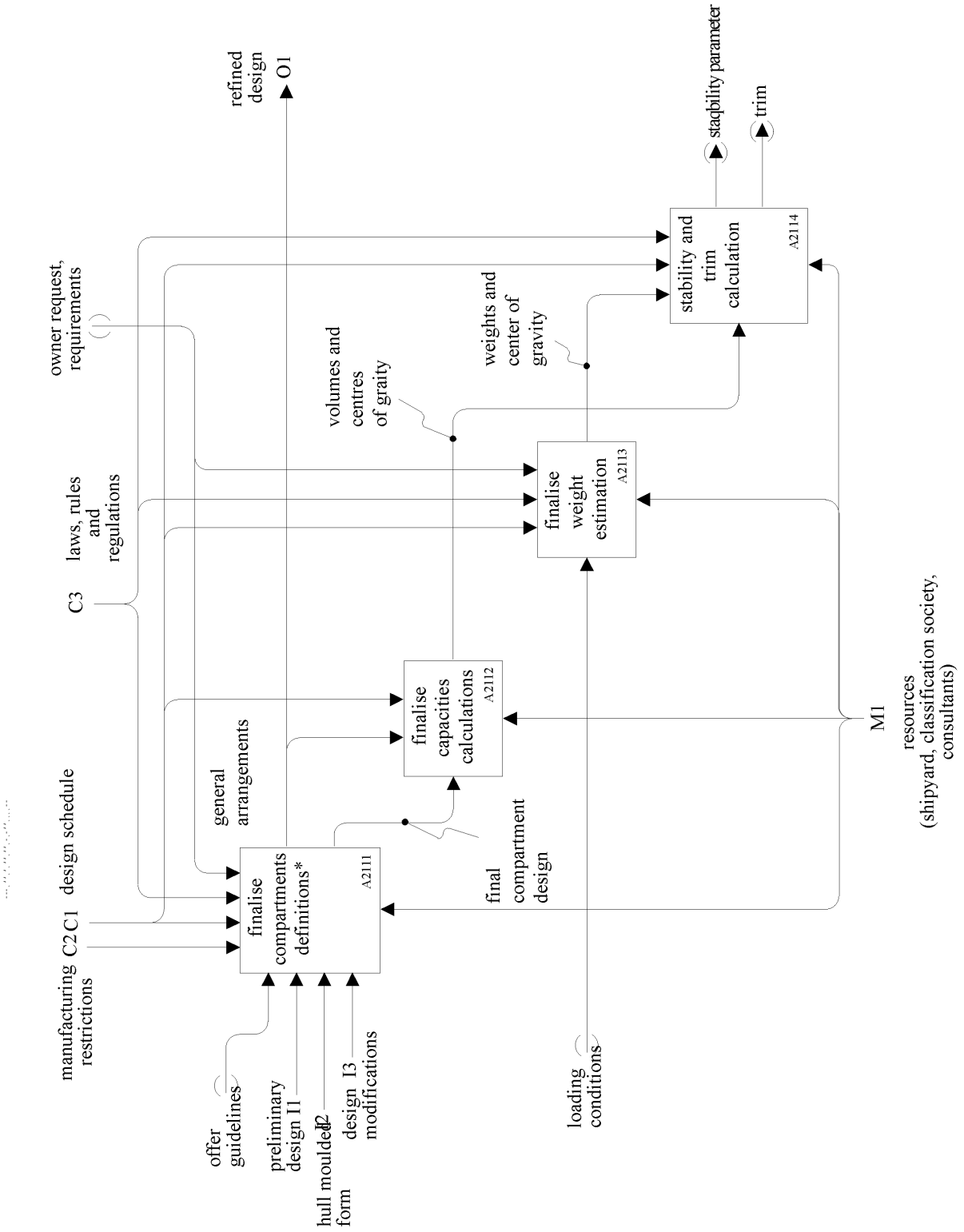

Figure F.14 — A211 Finalise general arrangements

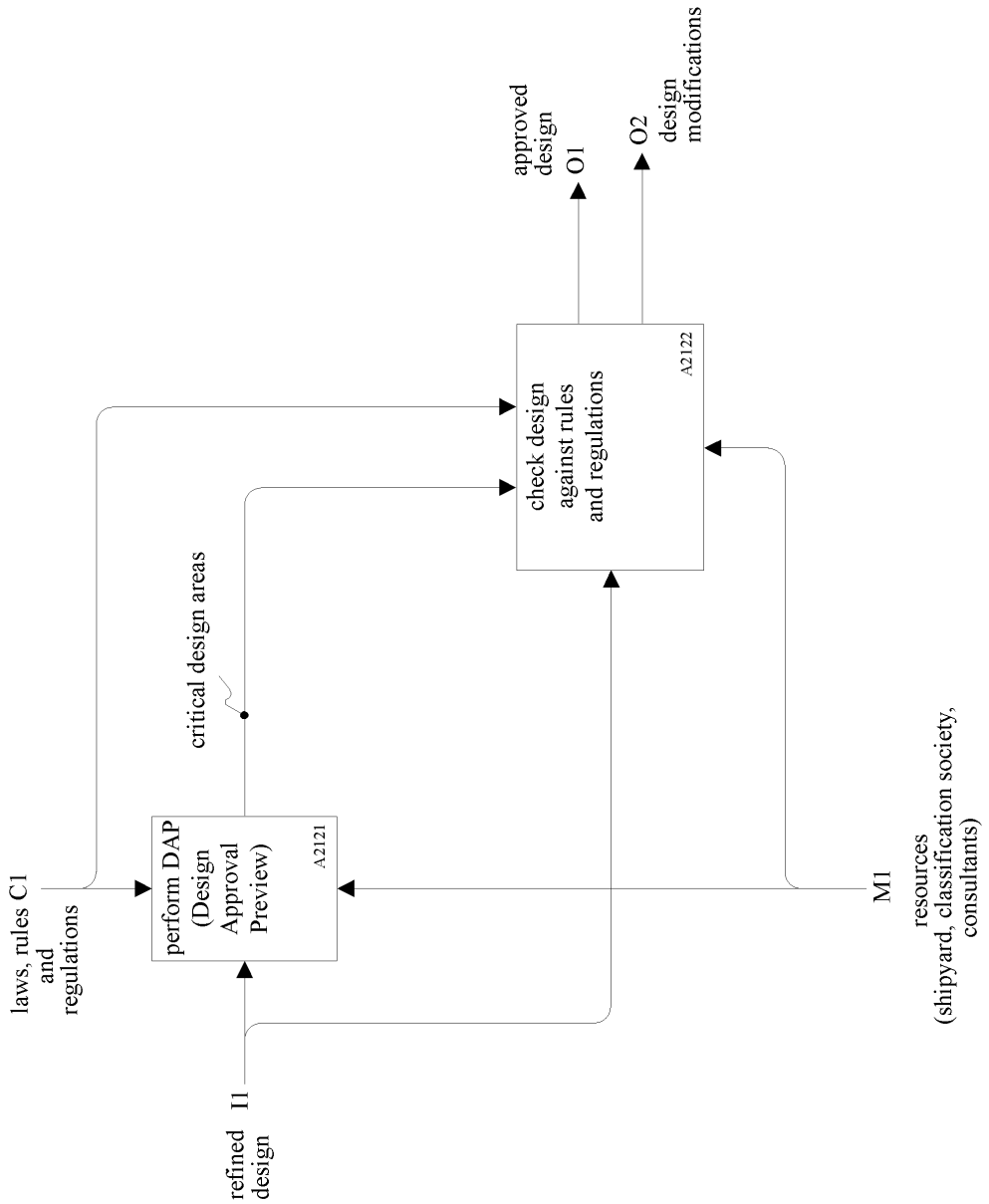

Figure F.15 — A212 Approve general arrangements

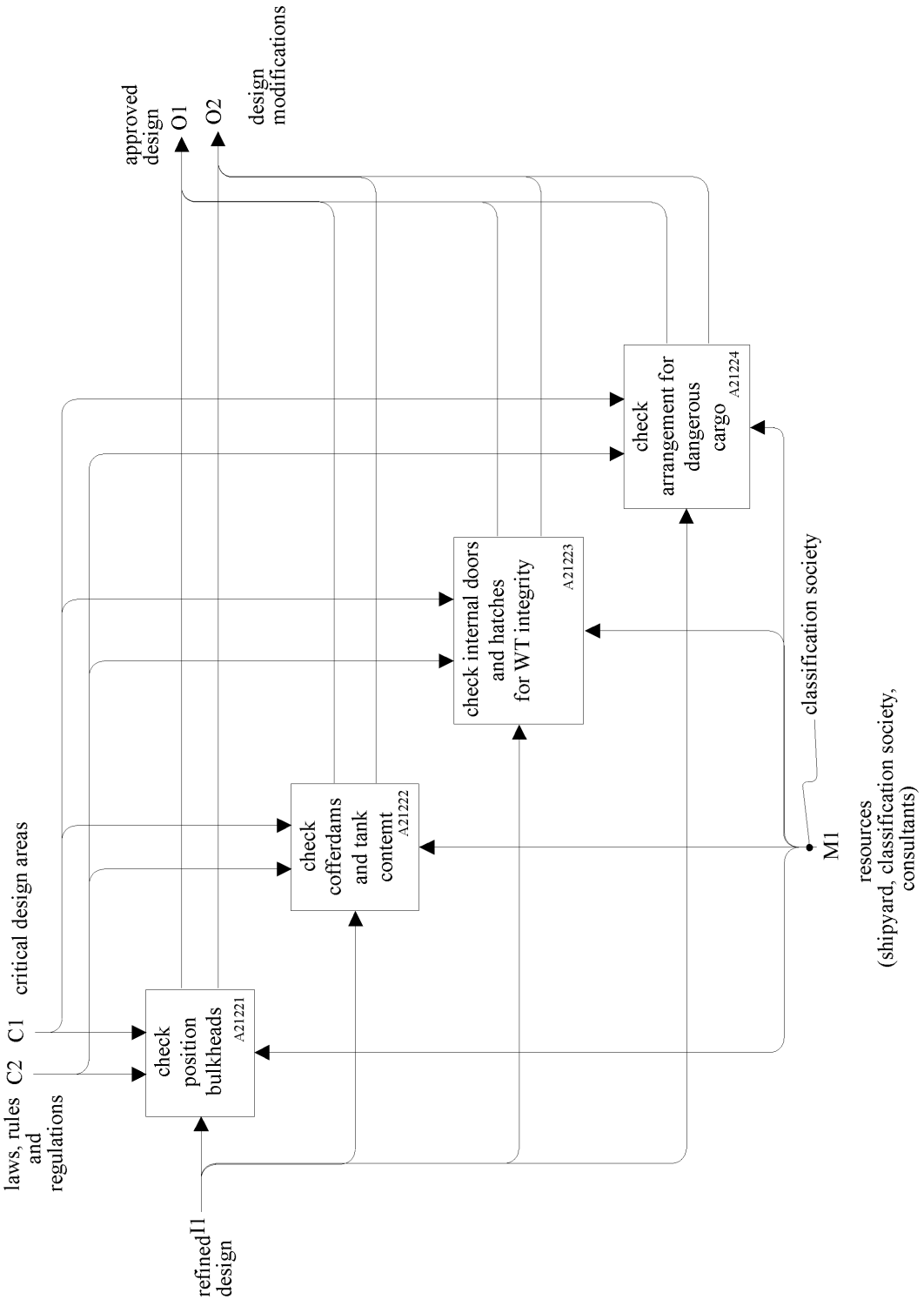

Figure F.16 — A2122 Check design against rules and regulations

Annex G

(normative)

Application reference model

This annex provides the application reference model for this part of ISO 10303 and is given in Figure G.1 to Figure G.25. The application reference model is a graphical representation of the structure and constraints of the application objects specified in clause 4. The graphical form of the application reference model is presented in EXPRESS-G. The application reference model is independent of any implementation method. EXPRESS-G is defined in annex A of ISO 10303-11.

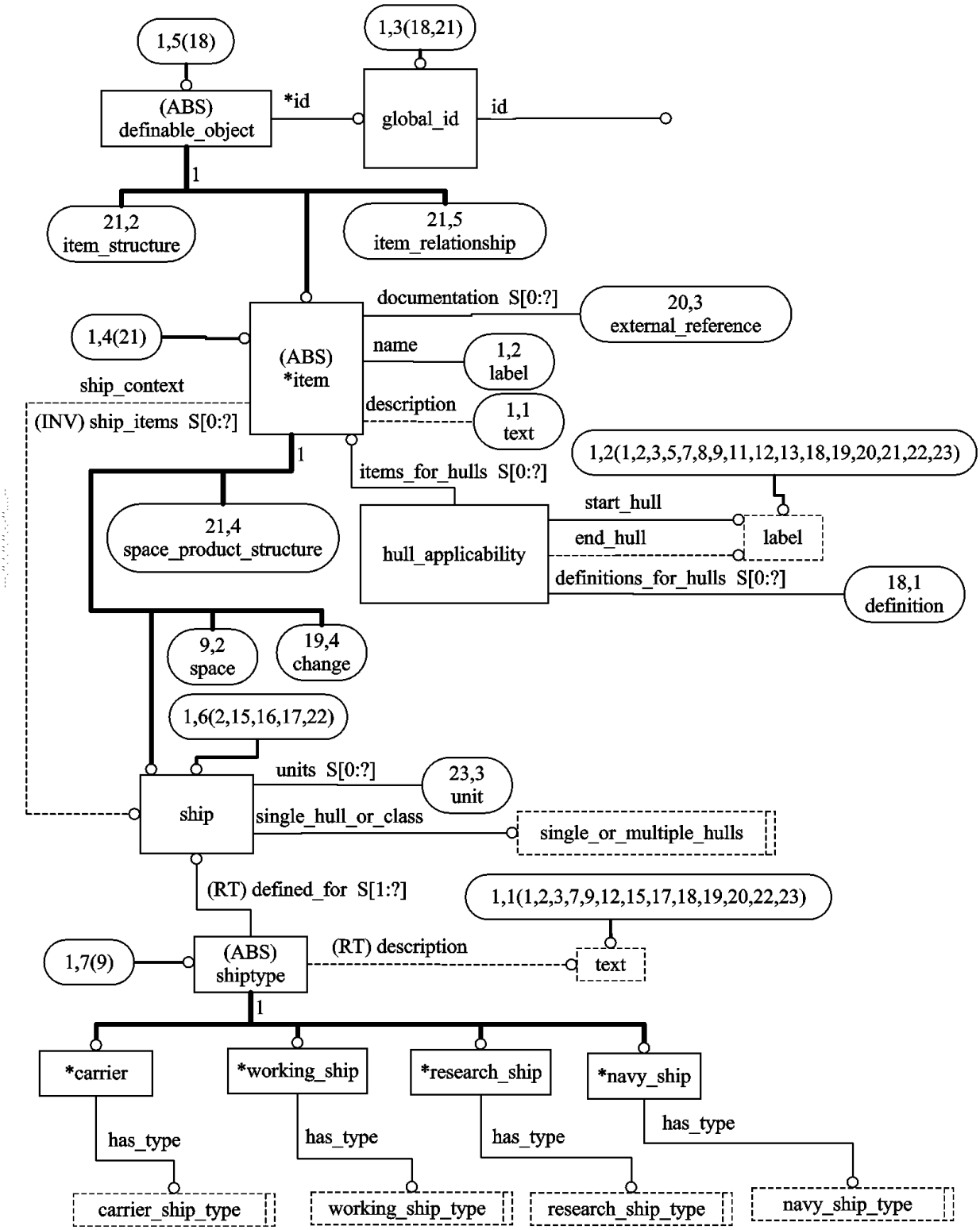

Figure G.1 — ARM diagram (1 of 23)

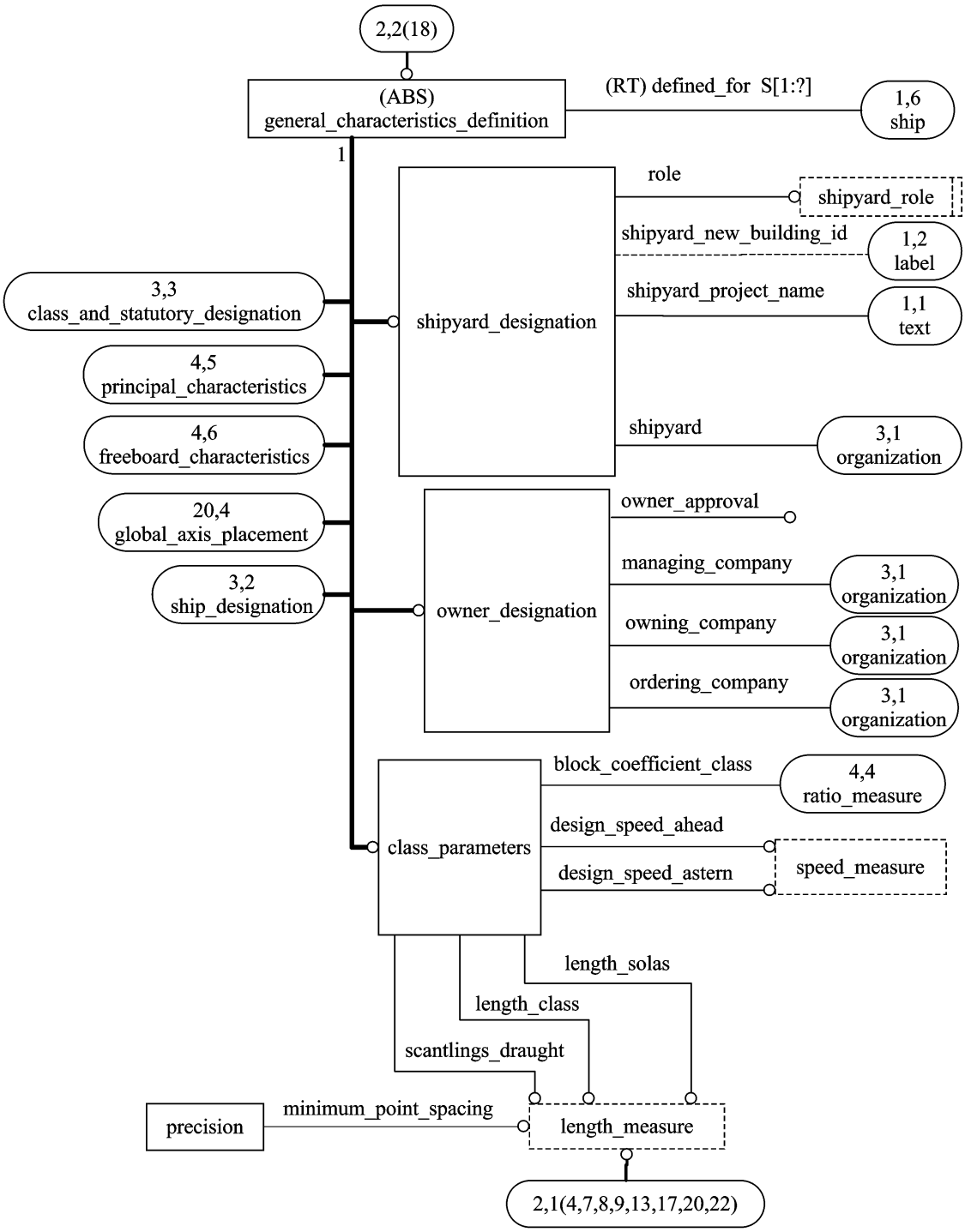

Figure G.2 — ARM diagram (2 of 23)

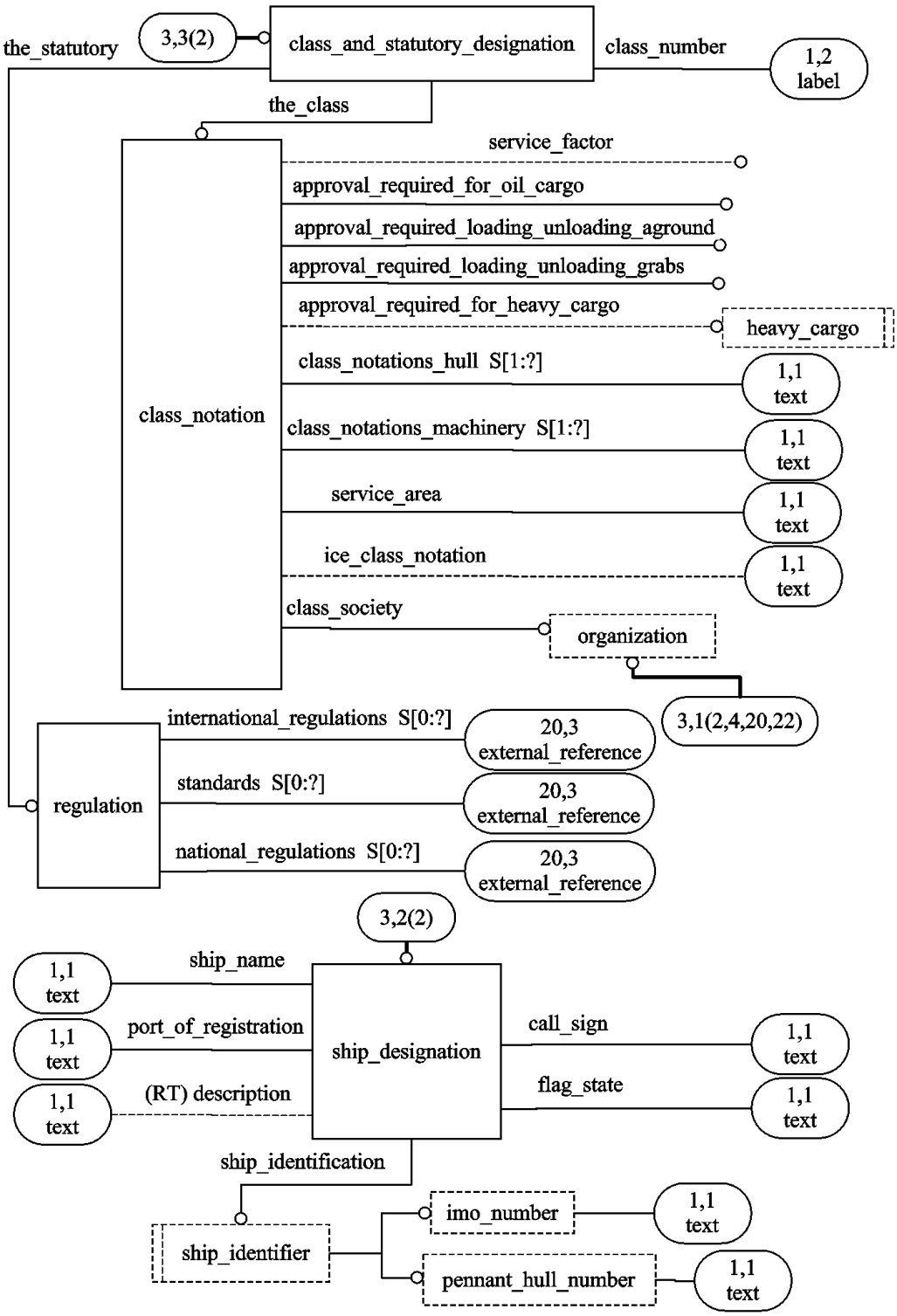

Figure G.3 — ARM diagram (3 of 23)

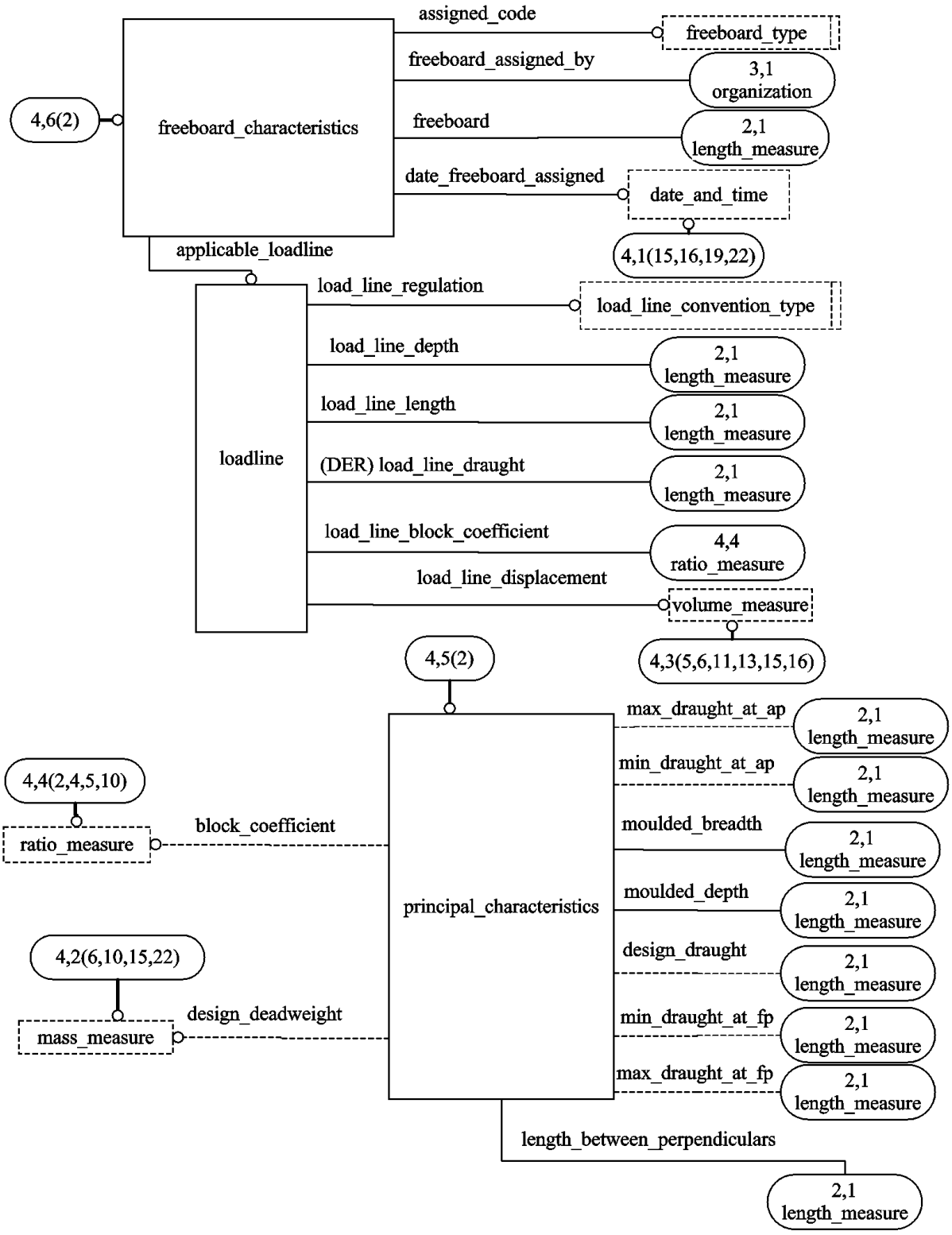

Figure G.4 — ARM diagram (4 of 23)

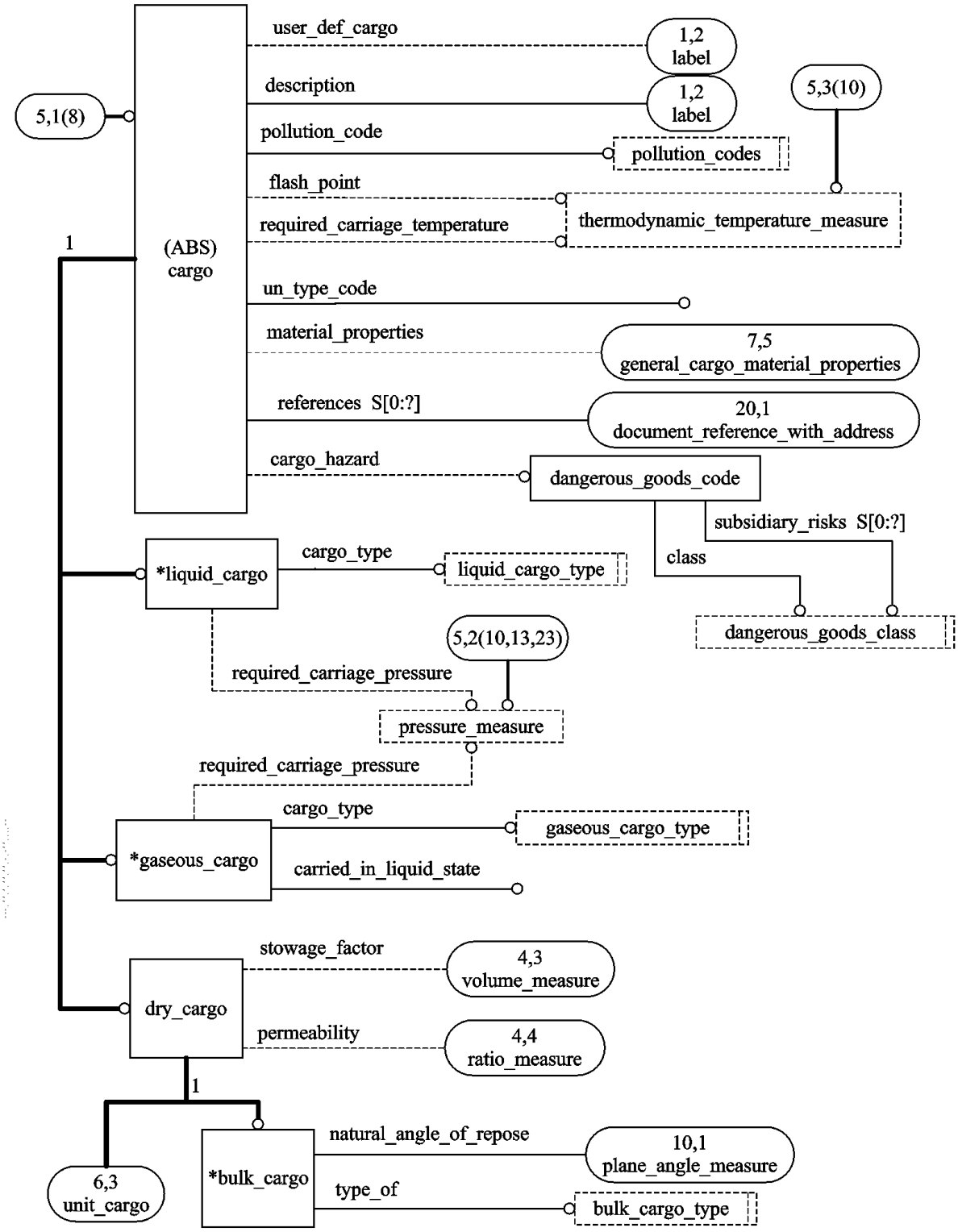

Figure G.5 — ARM diagram (5 of 23)

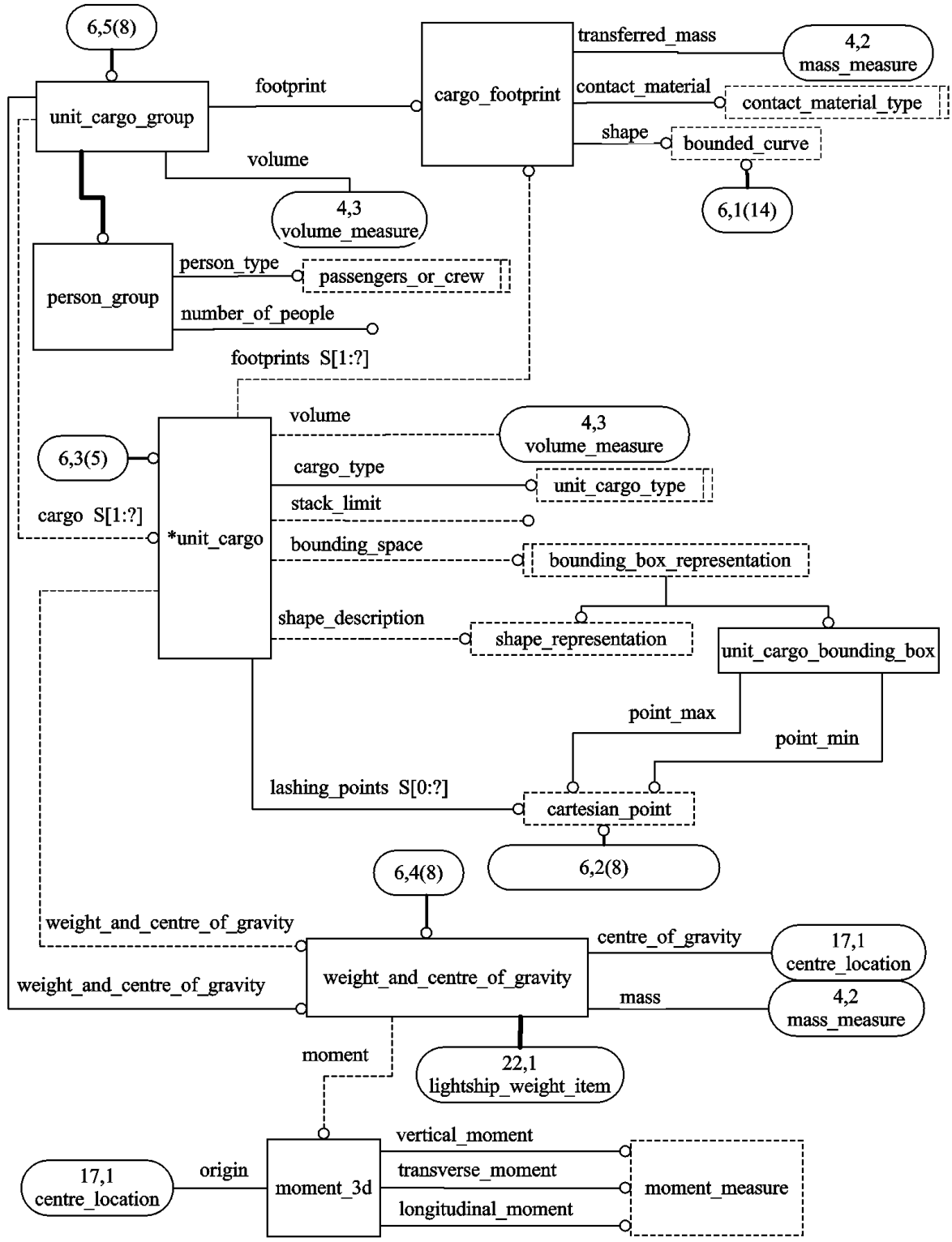

Figure G.6 — ARM diagram (6 of 23)

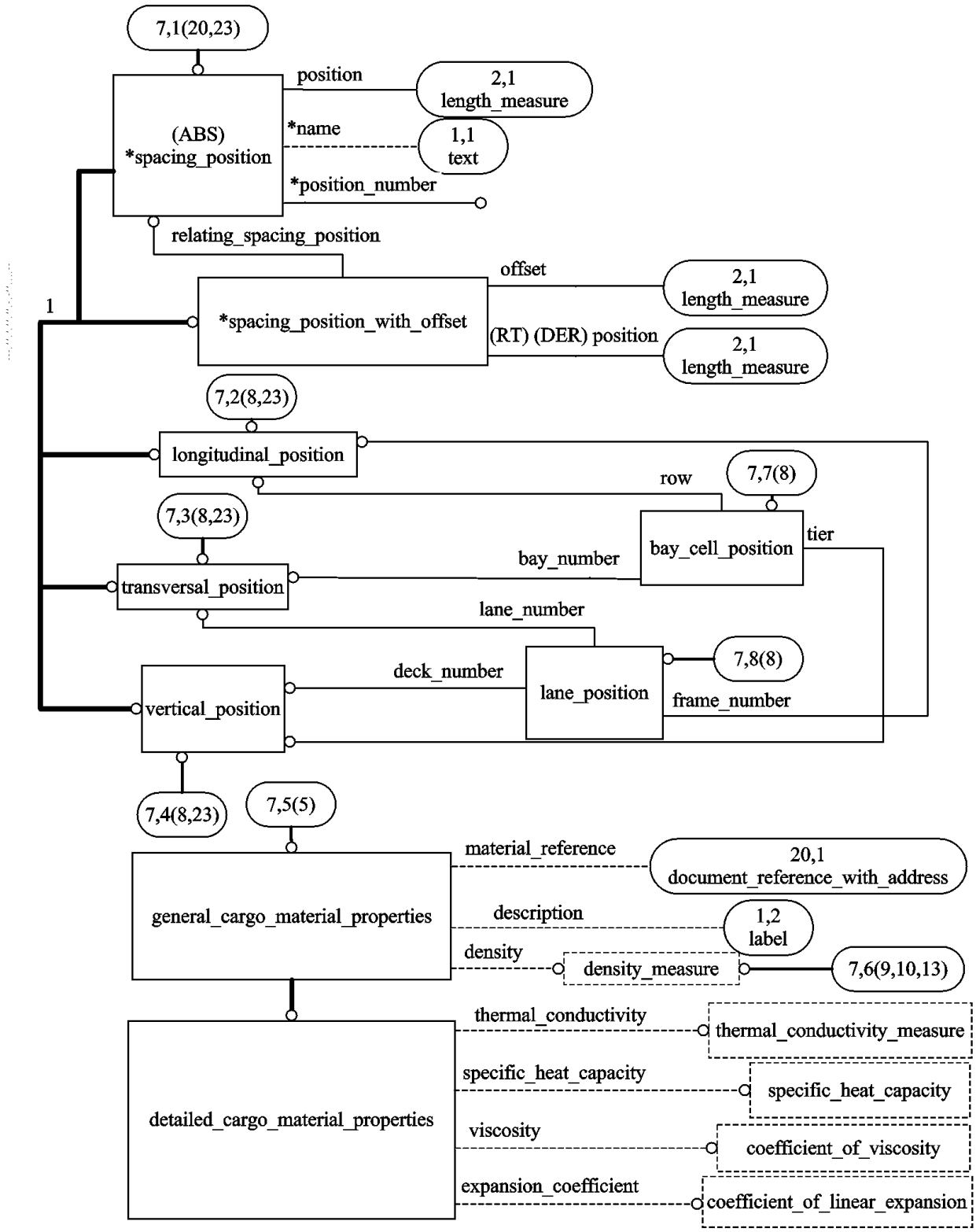

Figure G.7 — ARM diagram (7 of 23)

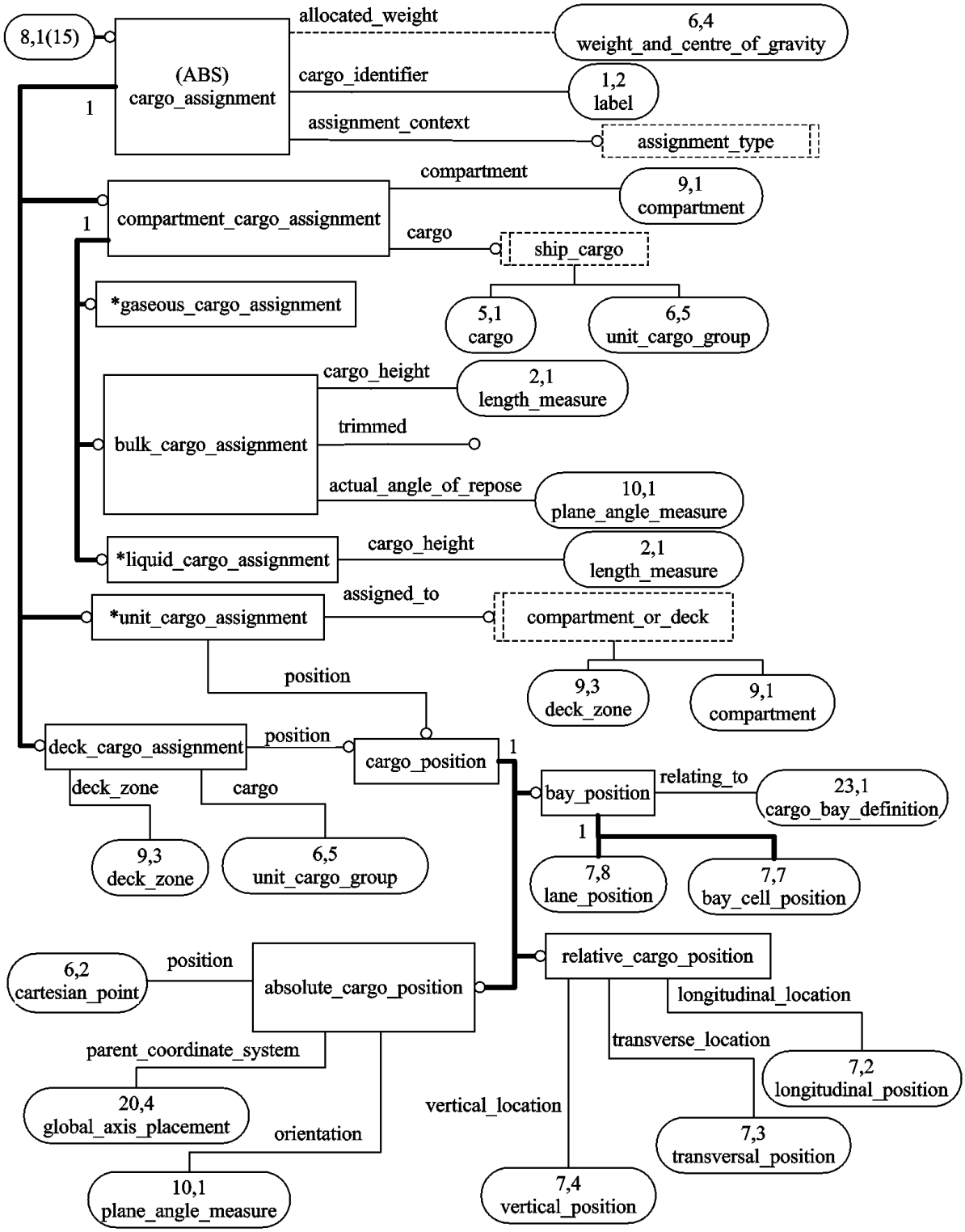

Figure G.8 — ARM diagram (8 of 23)

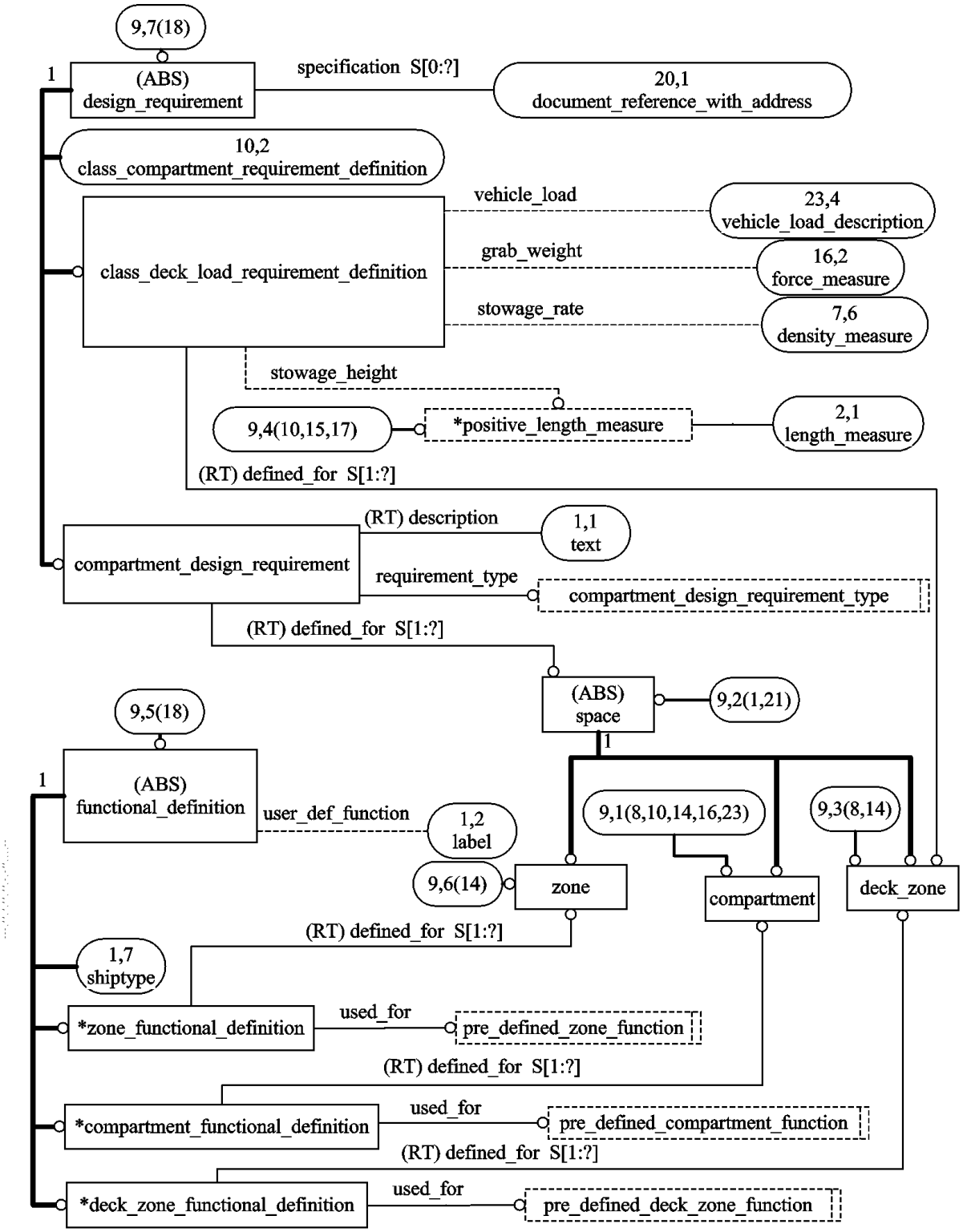

Figure G.9 — ARM diagram (9 of 23)

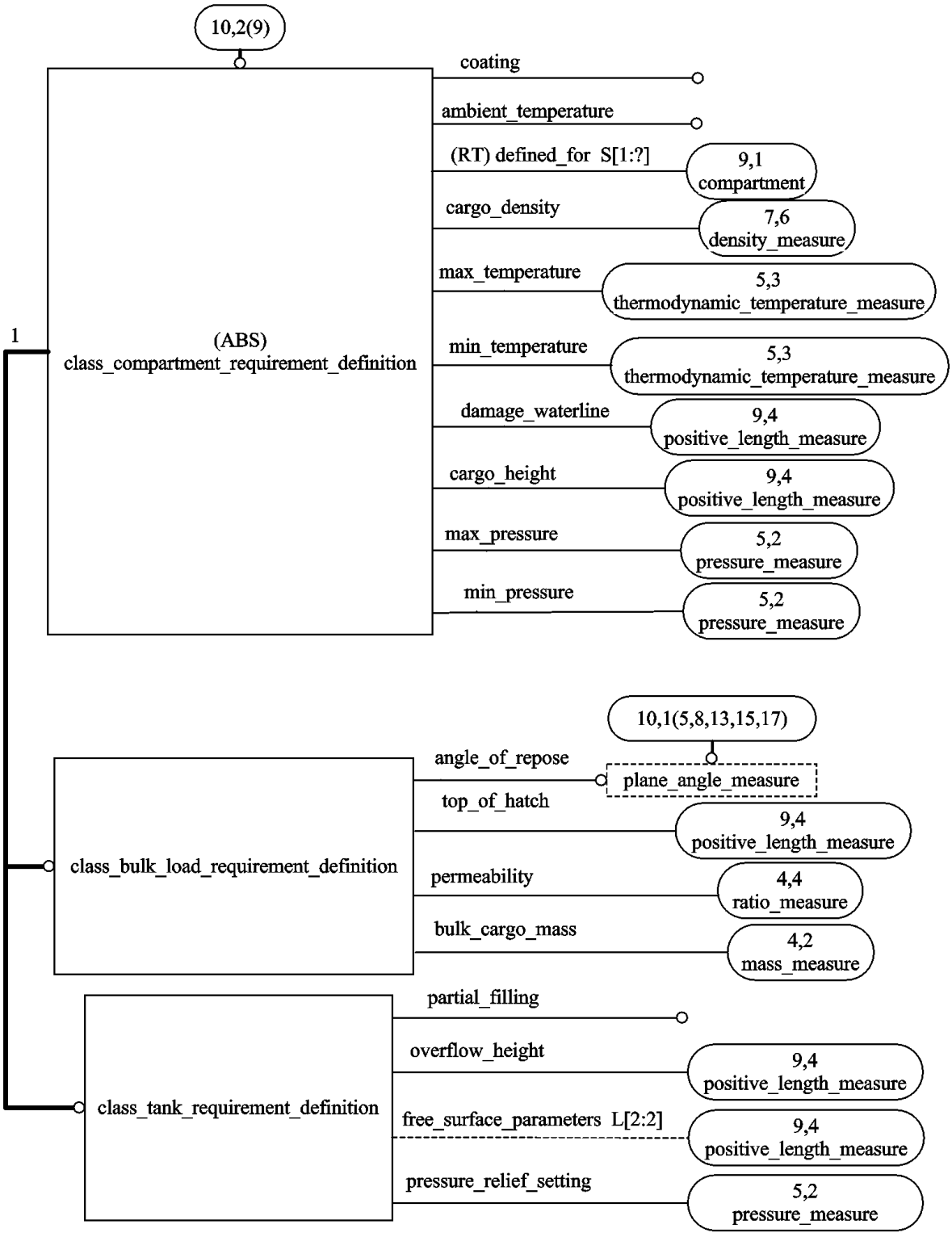

Figure G.10 — ARM diagram (10 of 23)

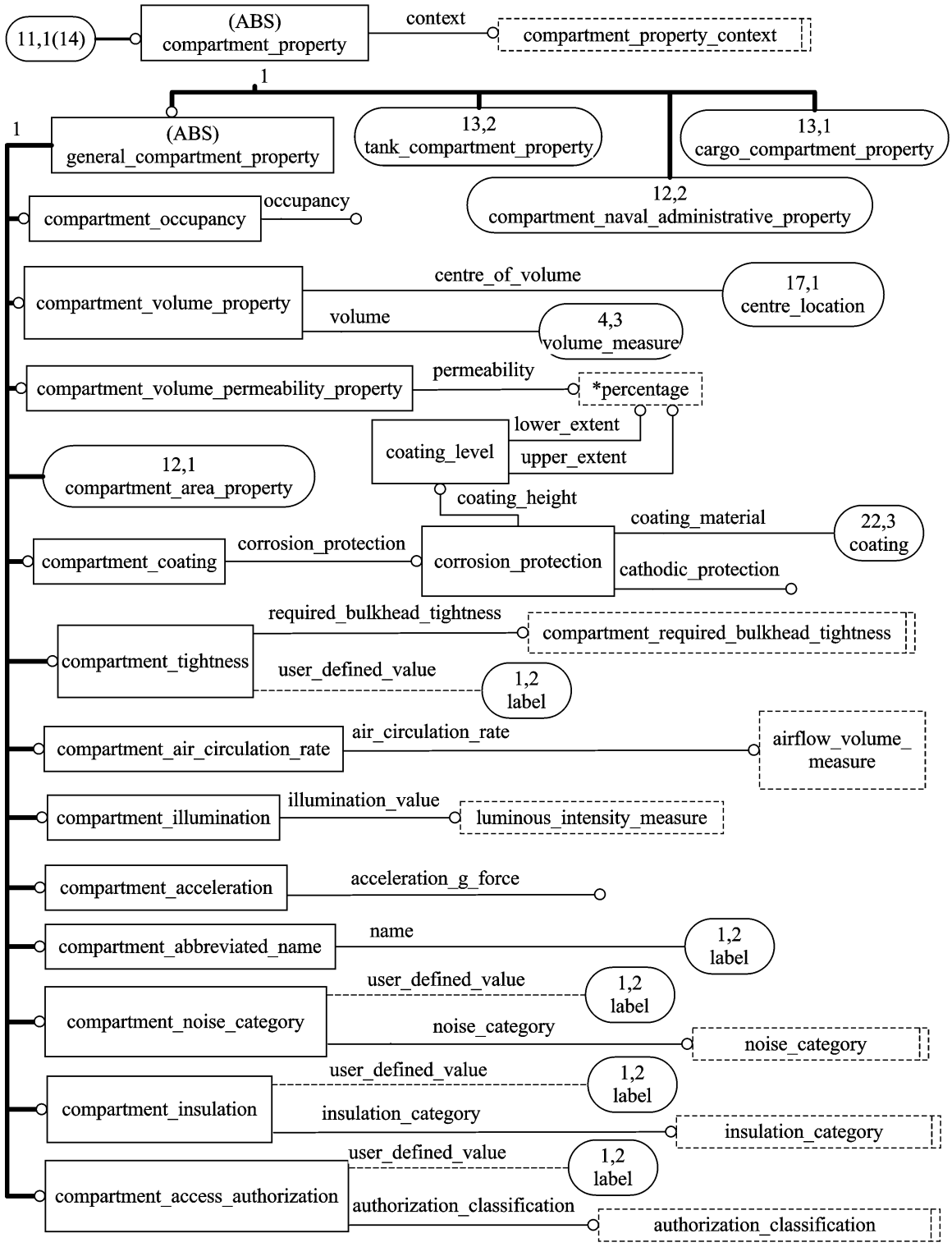

Figure G.11 — ARM diagram (11 of 23)

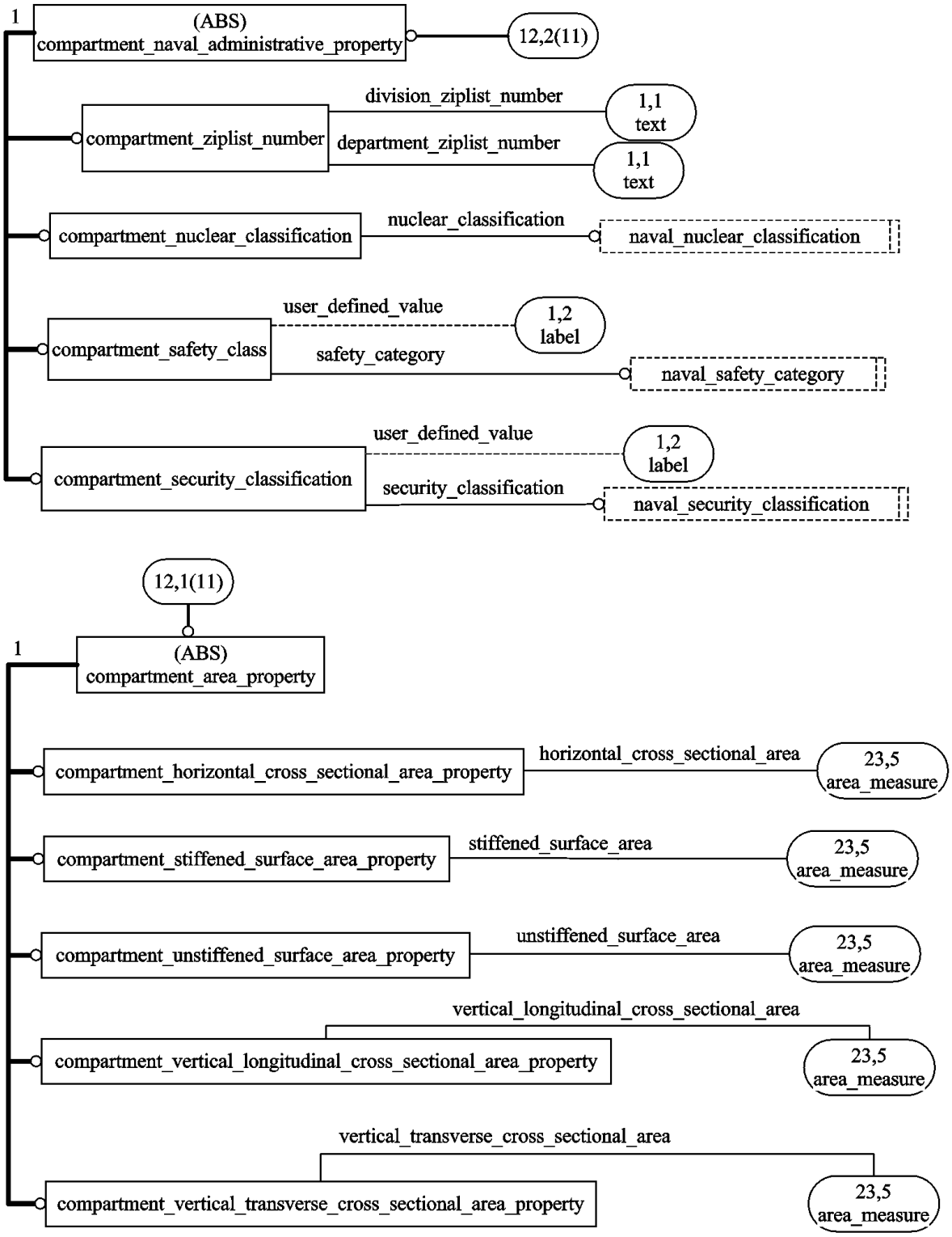

Figure G.12 — ARM diagram (12 of 23)

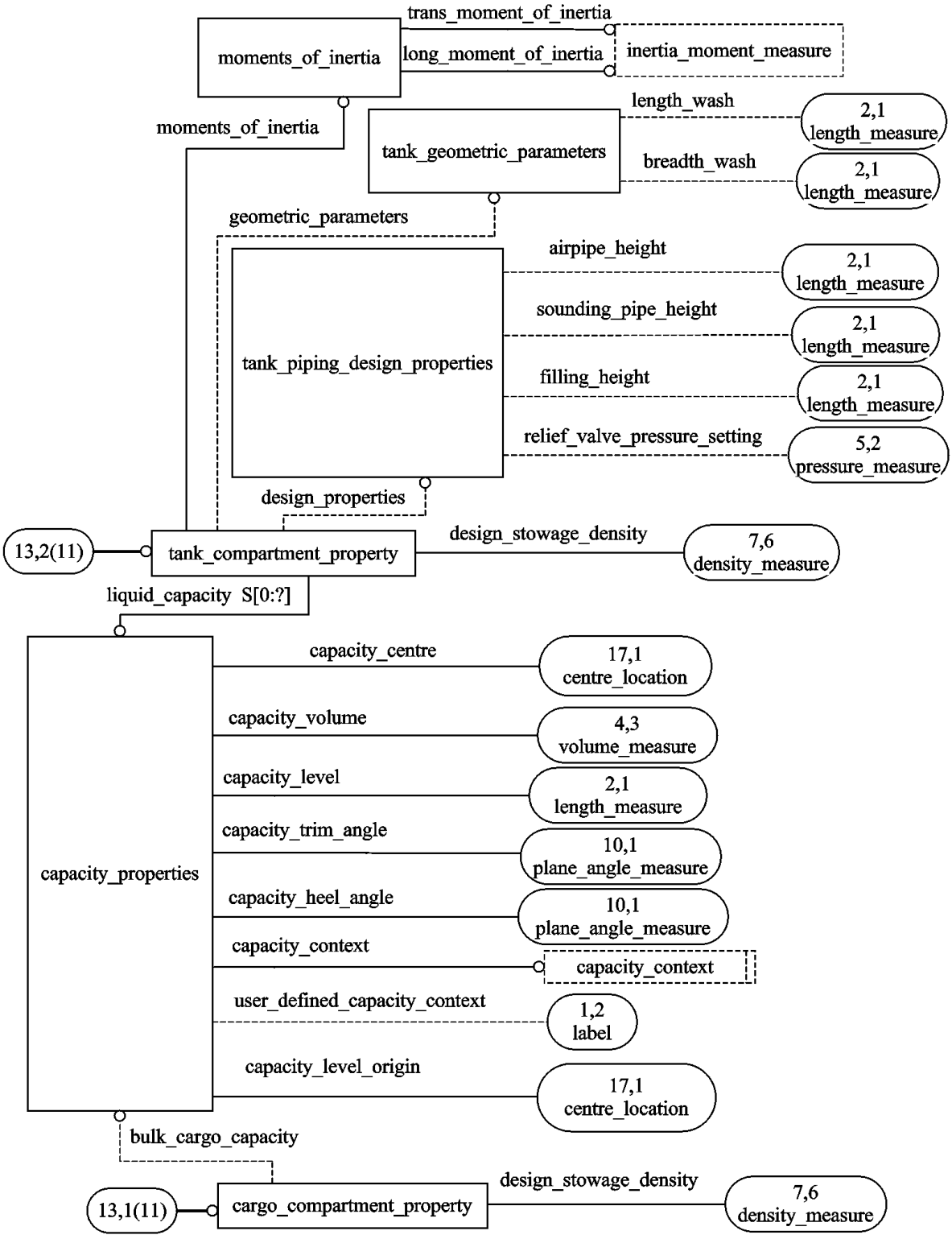

Figure G.13 — ARM diagram (13 of 23)

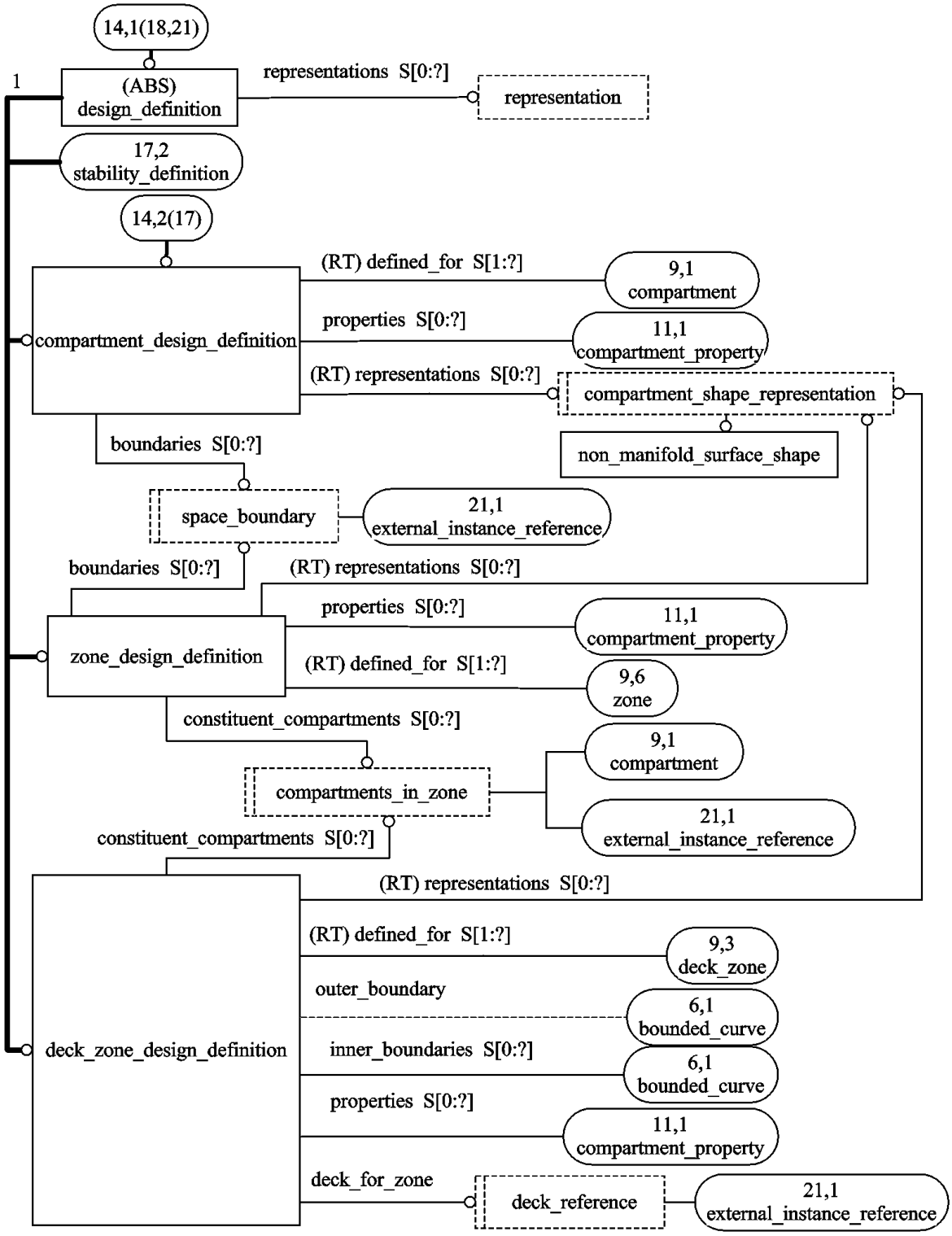

Figure G.14 — ARM diagram (14 of 23)

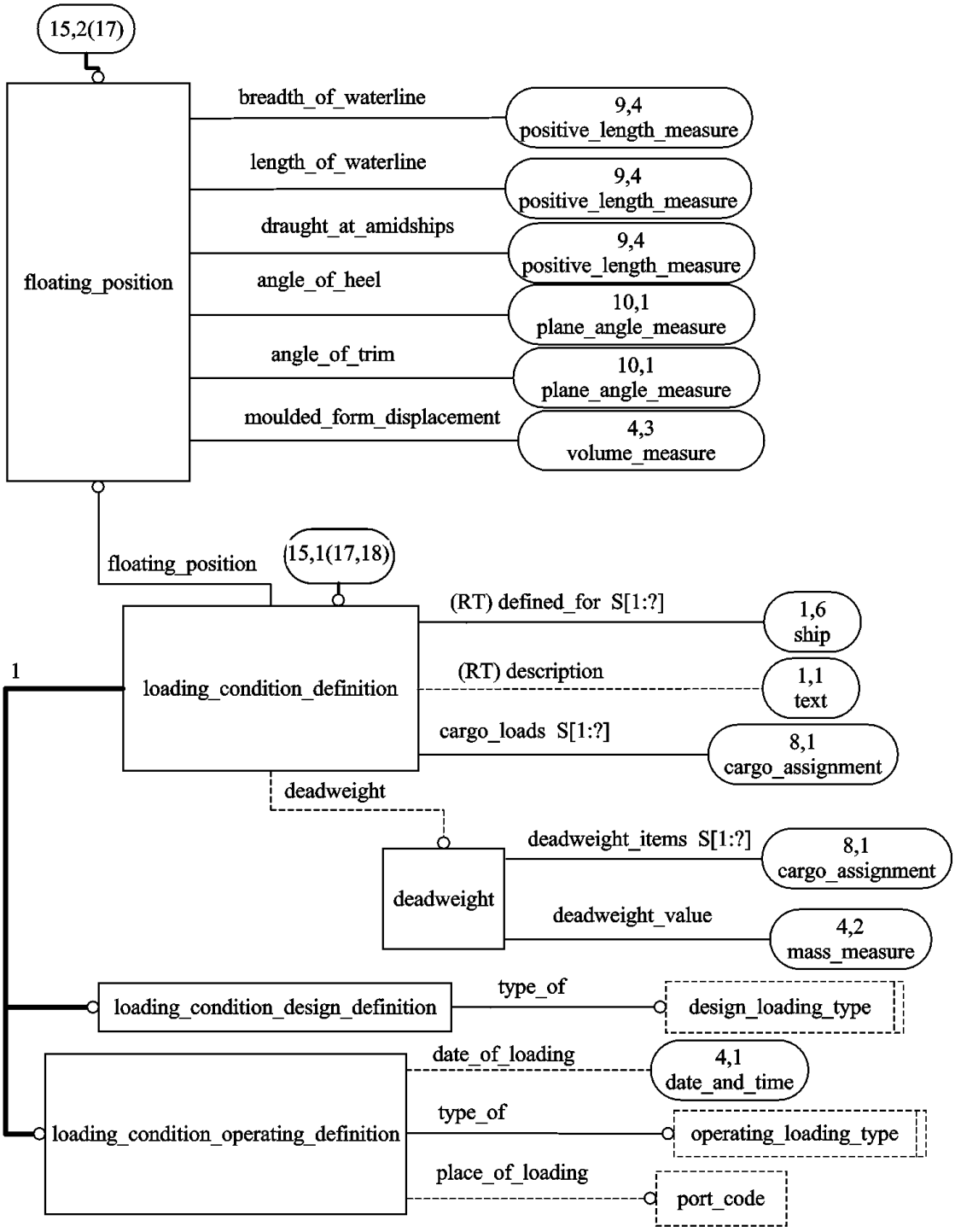

Figure G.15 — ARM diagram (15 of 23)

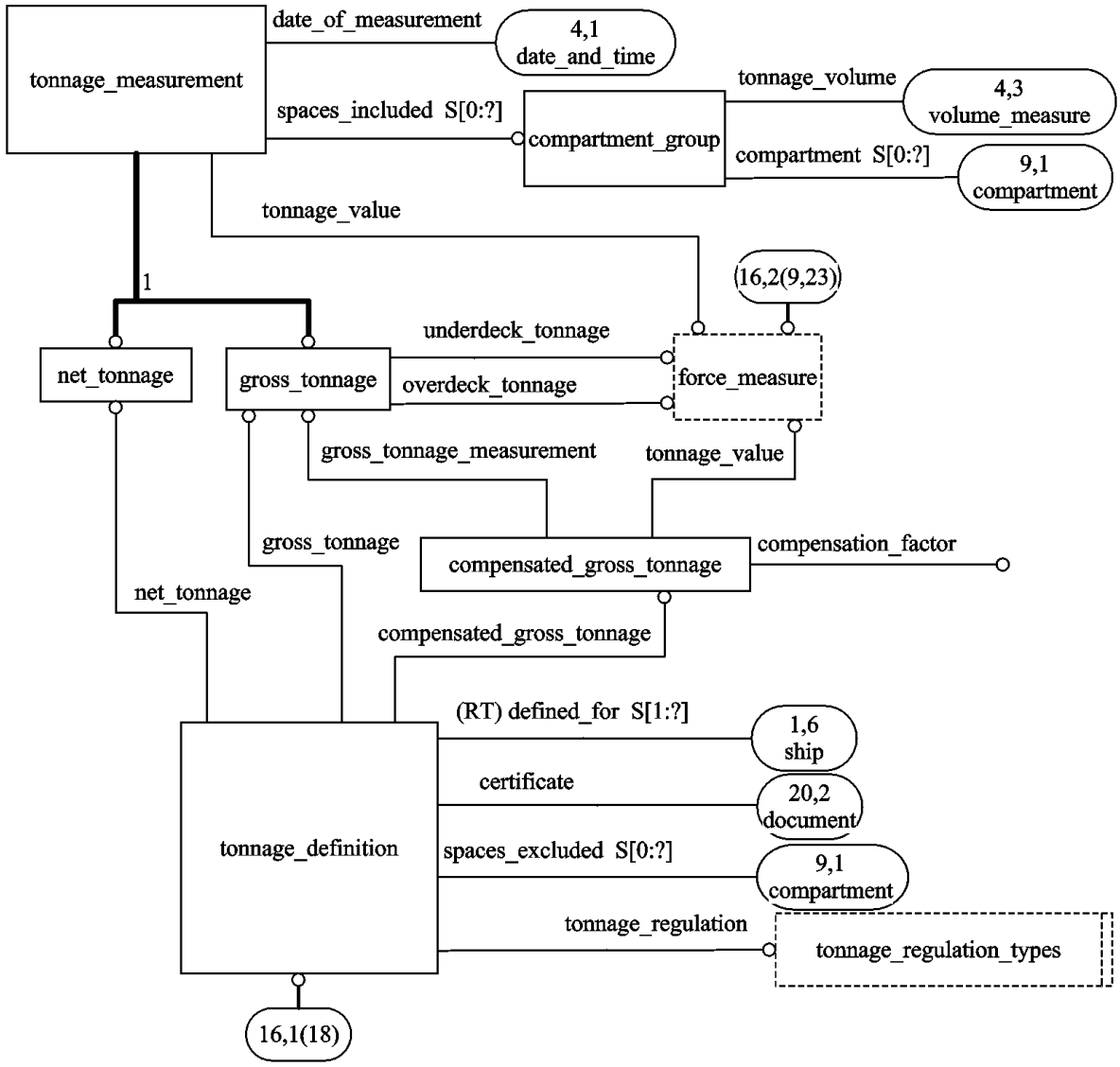

Figure G.16 — ARM diagram (16 of 23)

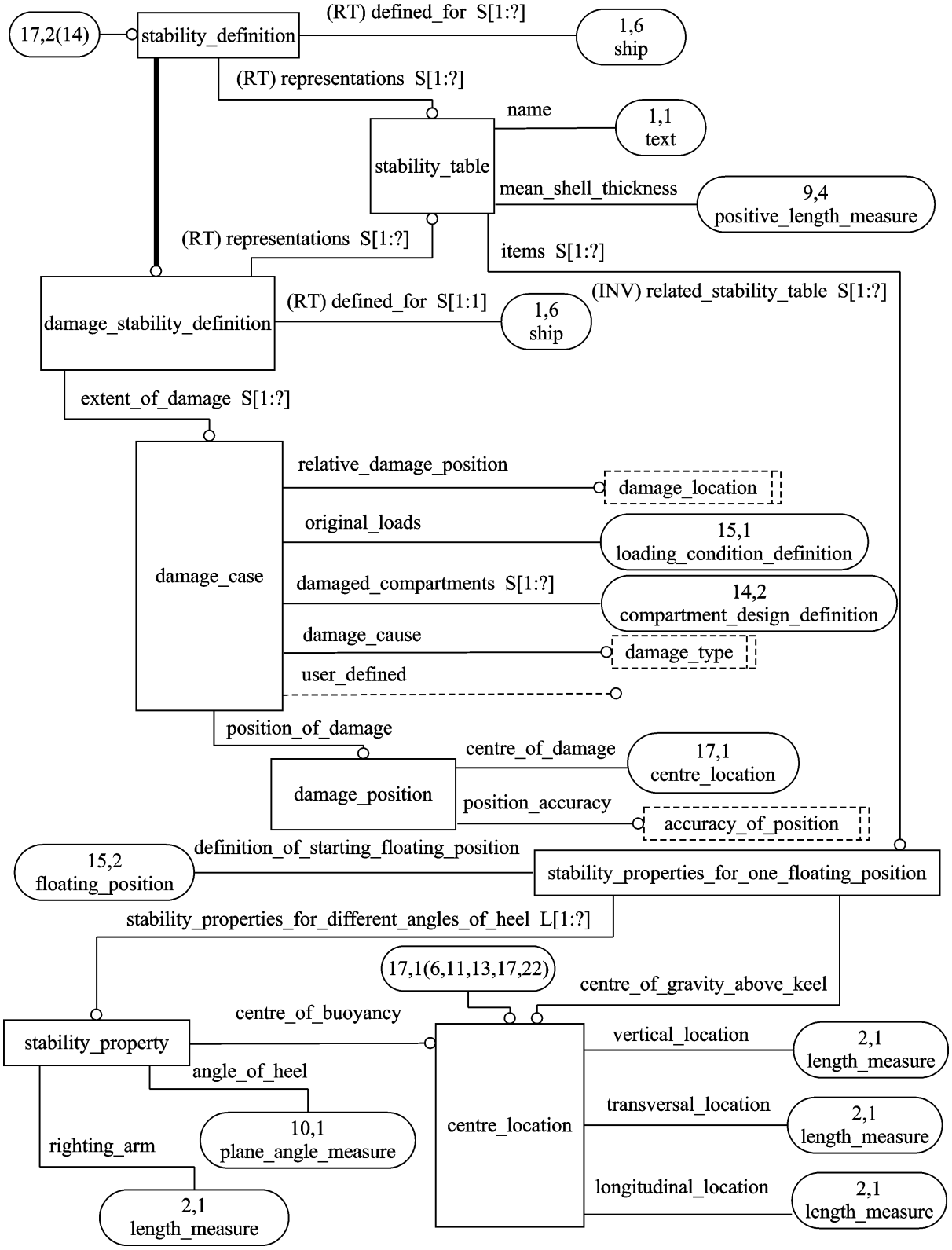

Figure G.17 — ARM diagram (17 of 23)

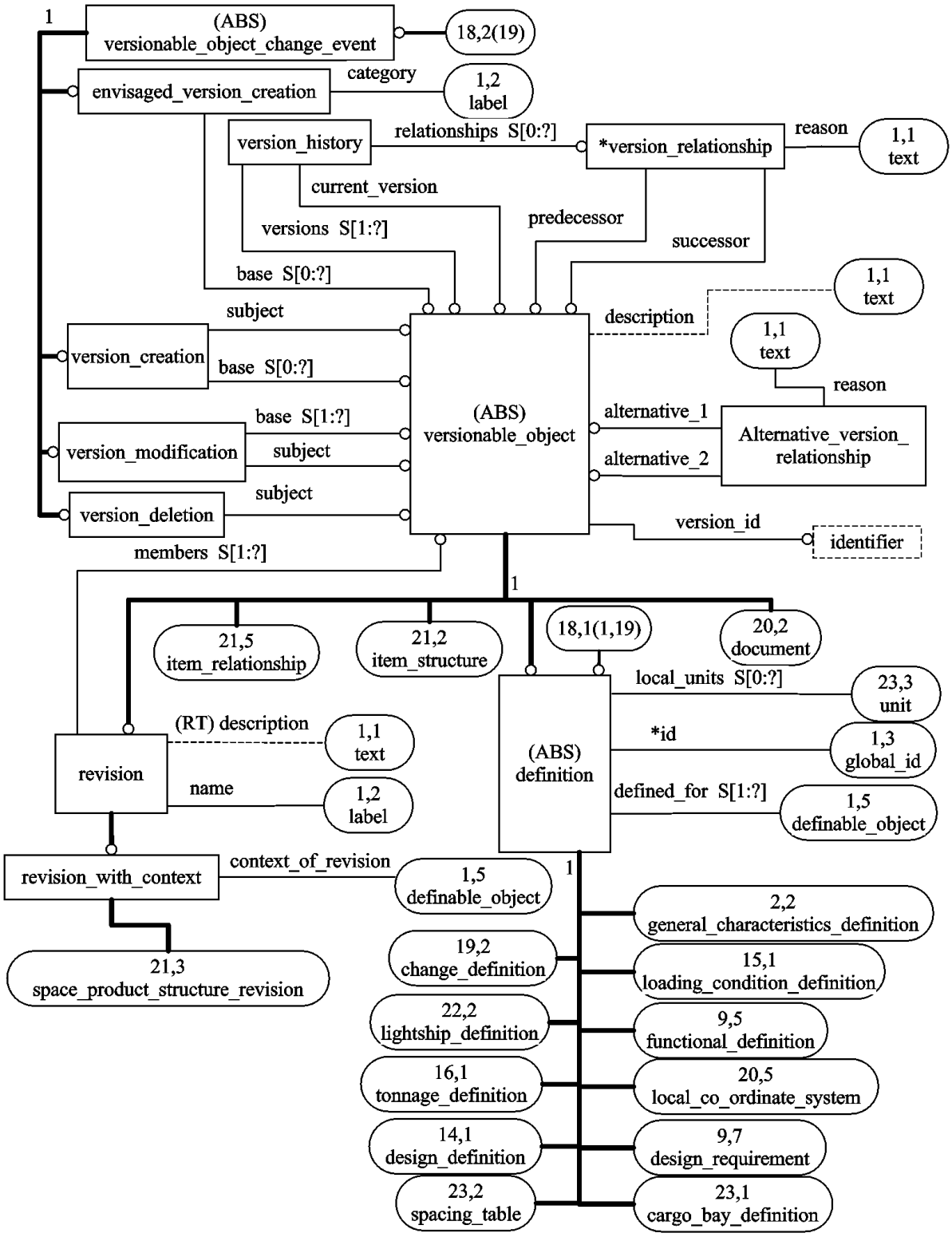

Figure G.18 — ARM diagram (18 of 23)

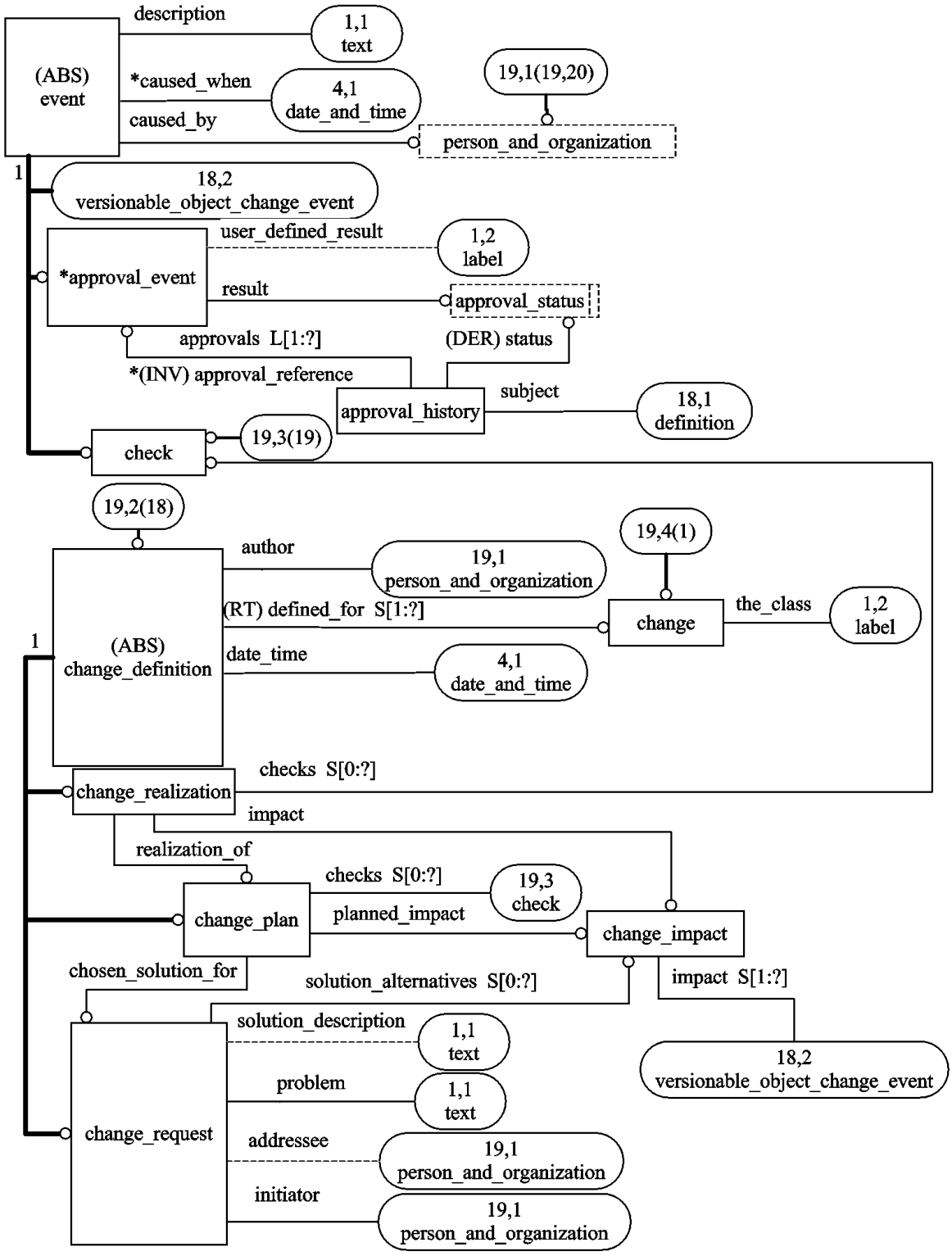

Figure G.19 — ARM diagram (19 of 23)

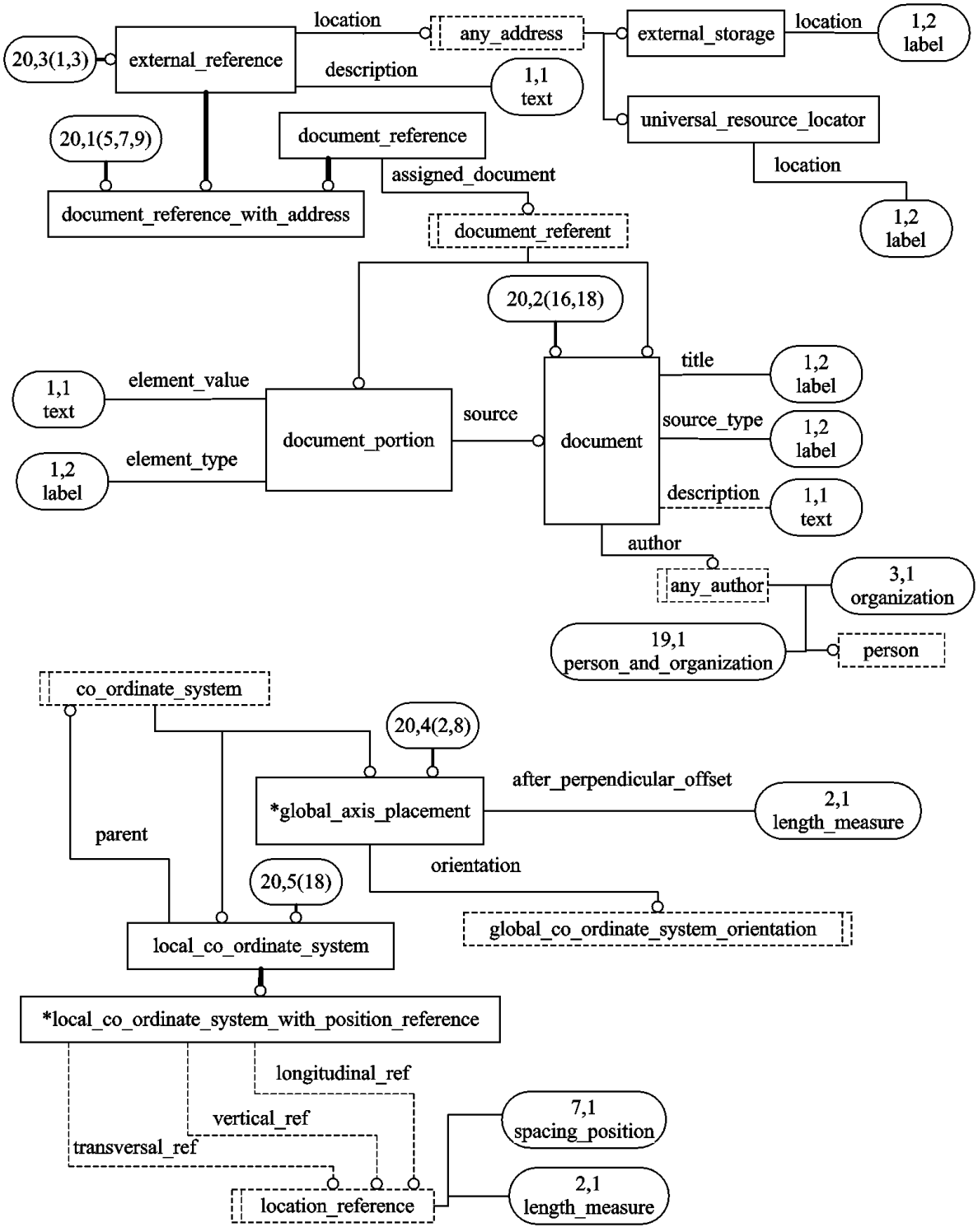

Figure G.20 — ARM diagram (20 of 23)

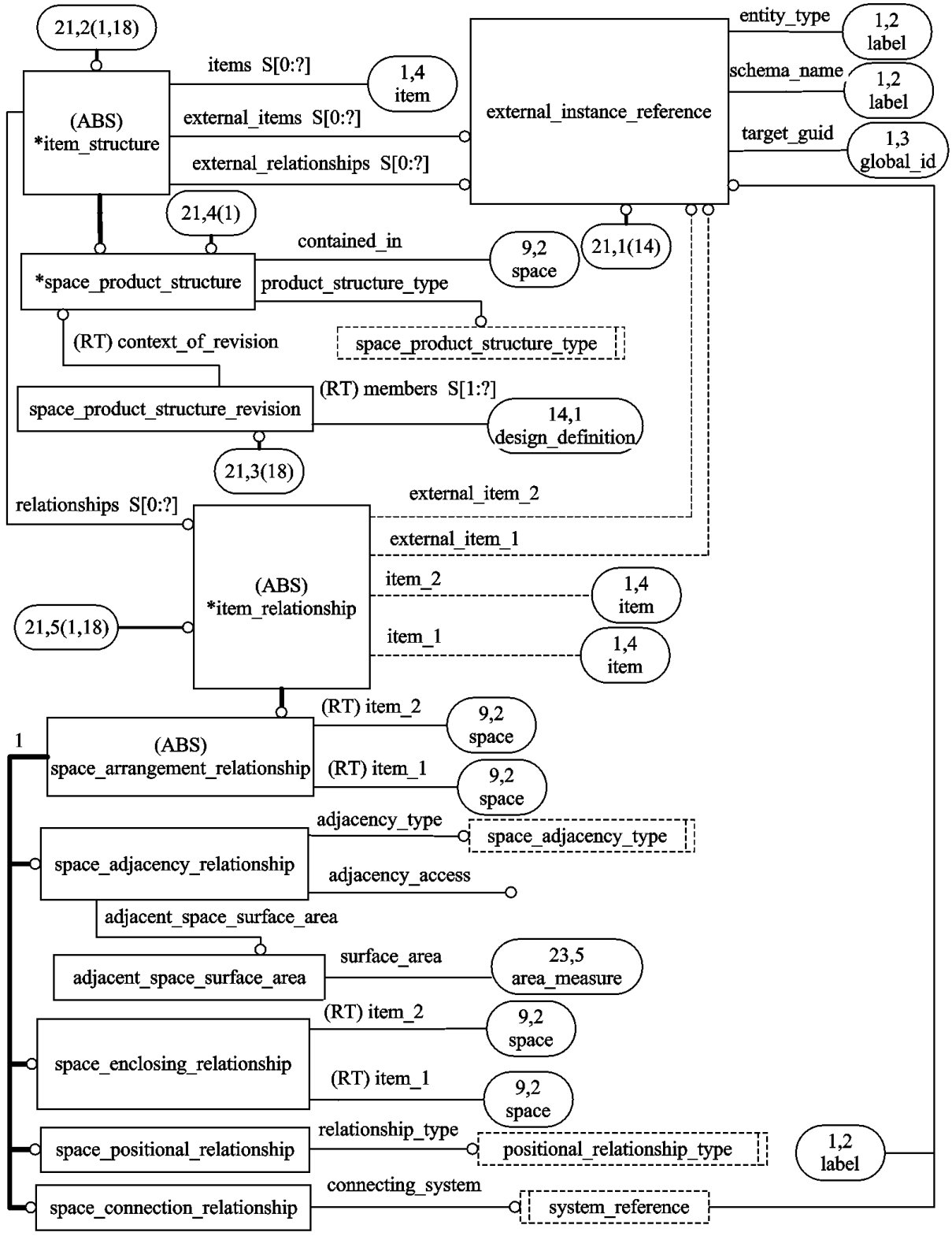

Figure G.21 — ARM diagram (21 of 23)

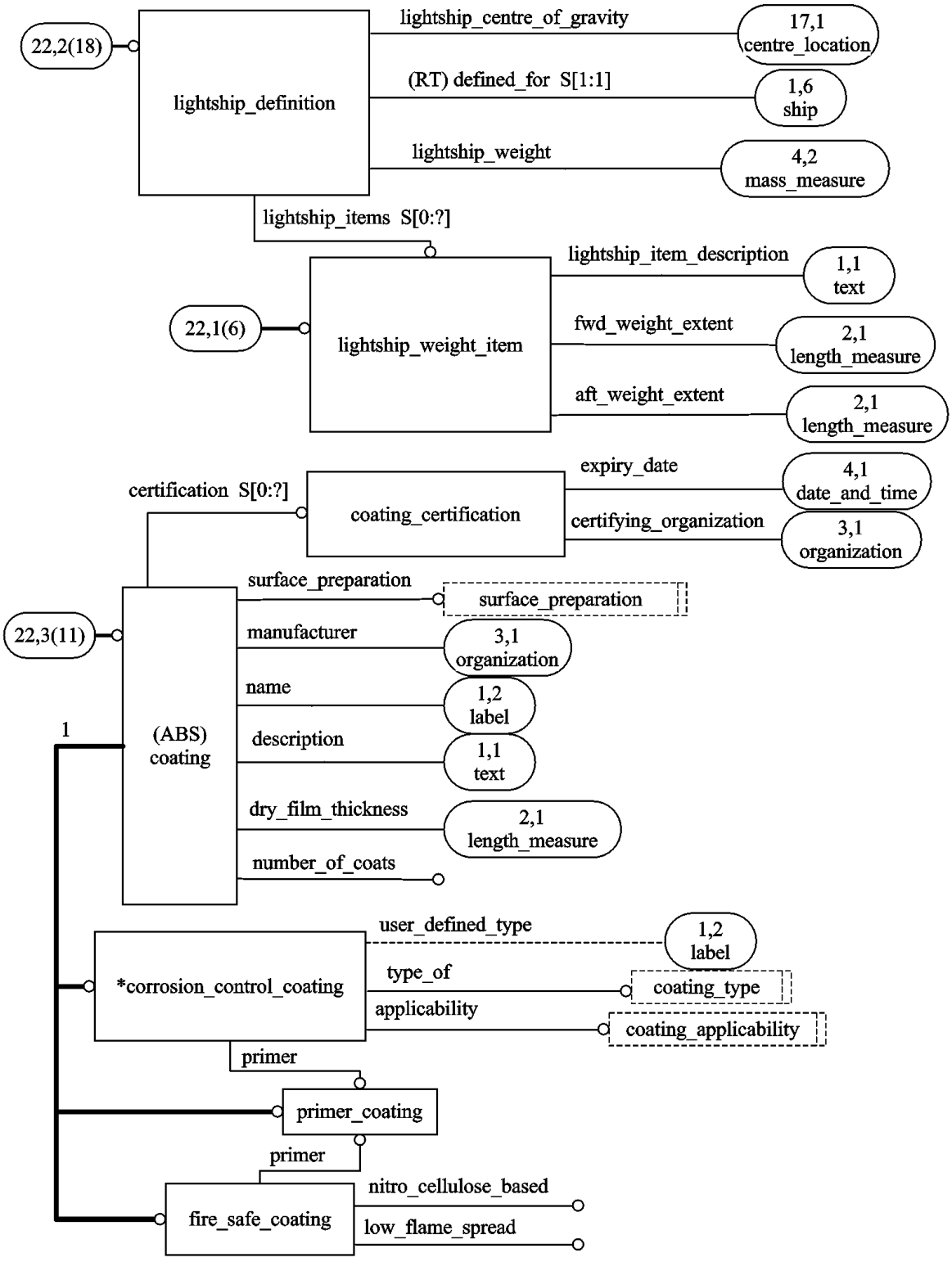

Figure G.22 — ARM diagram (22 of 23)

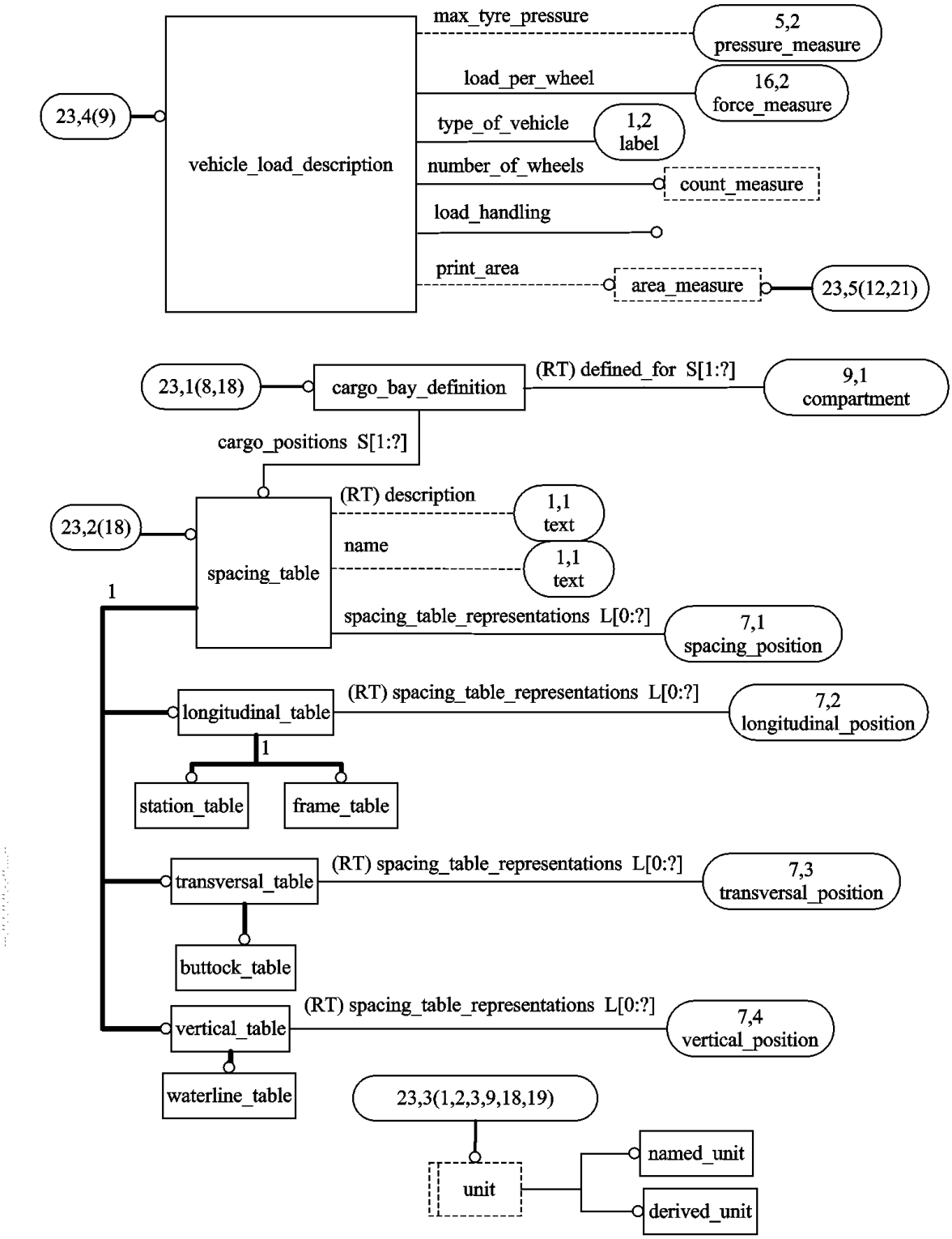

Figure G.23 — ARM diagram (23 of 23)

Annex H

(informative)

AIM EXPRESS-G

Figures H.1 through H.29 correspond to the AIM EXPRESS annotated listing given in annex A. The figures use the EXPRESS-G graphical notation for the EXPRESS language. EXPRESS-G is defined in annex D of ISO 10303-11.

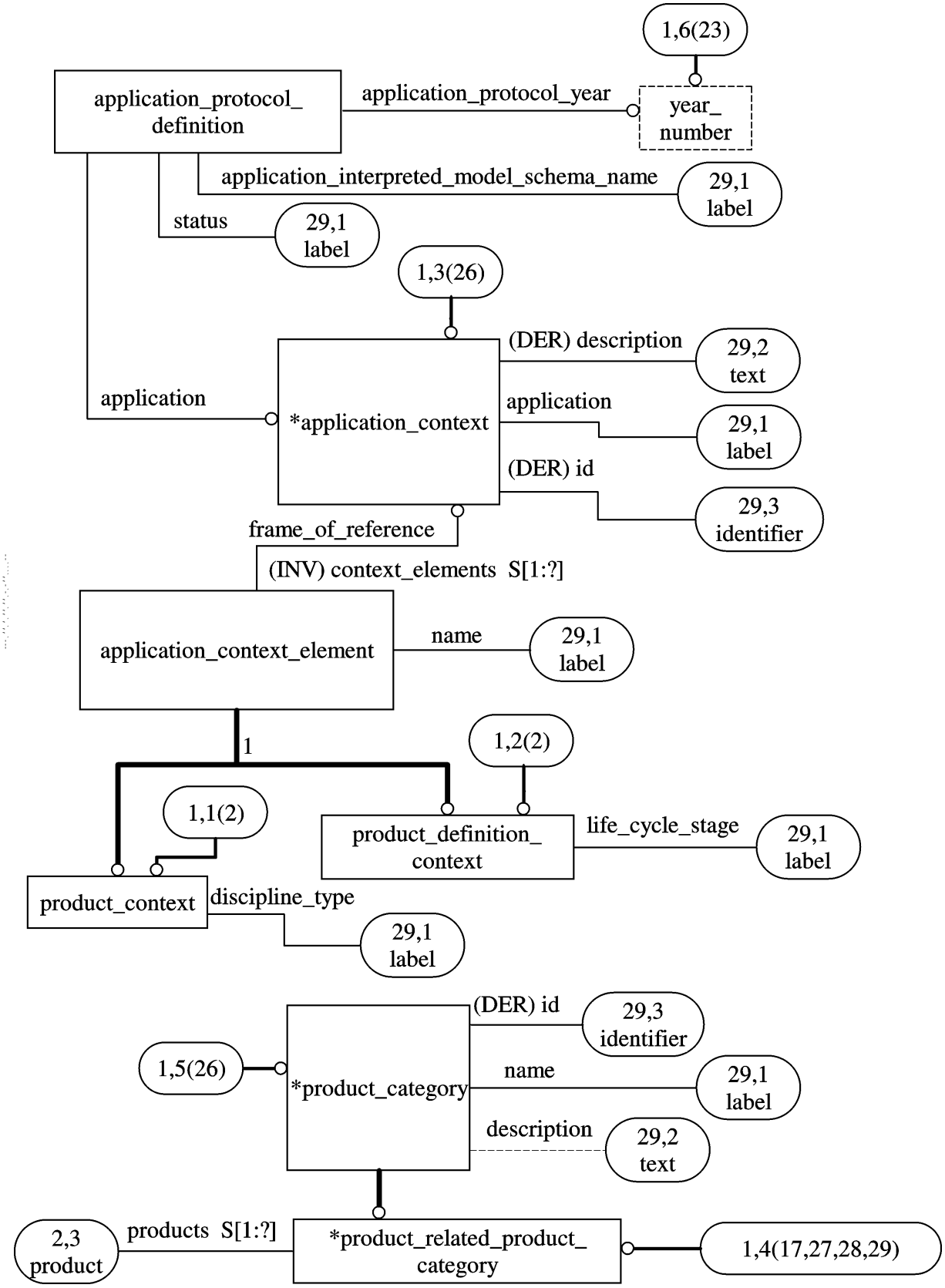

Figure H.1 — AIM EXPRESS-G diagram application context

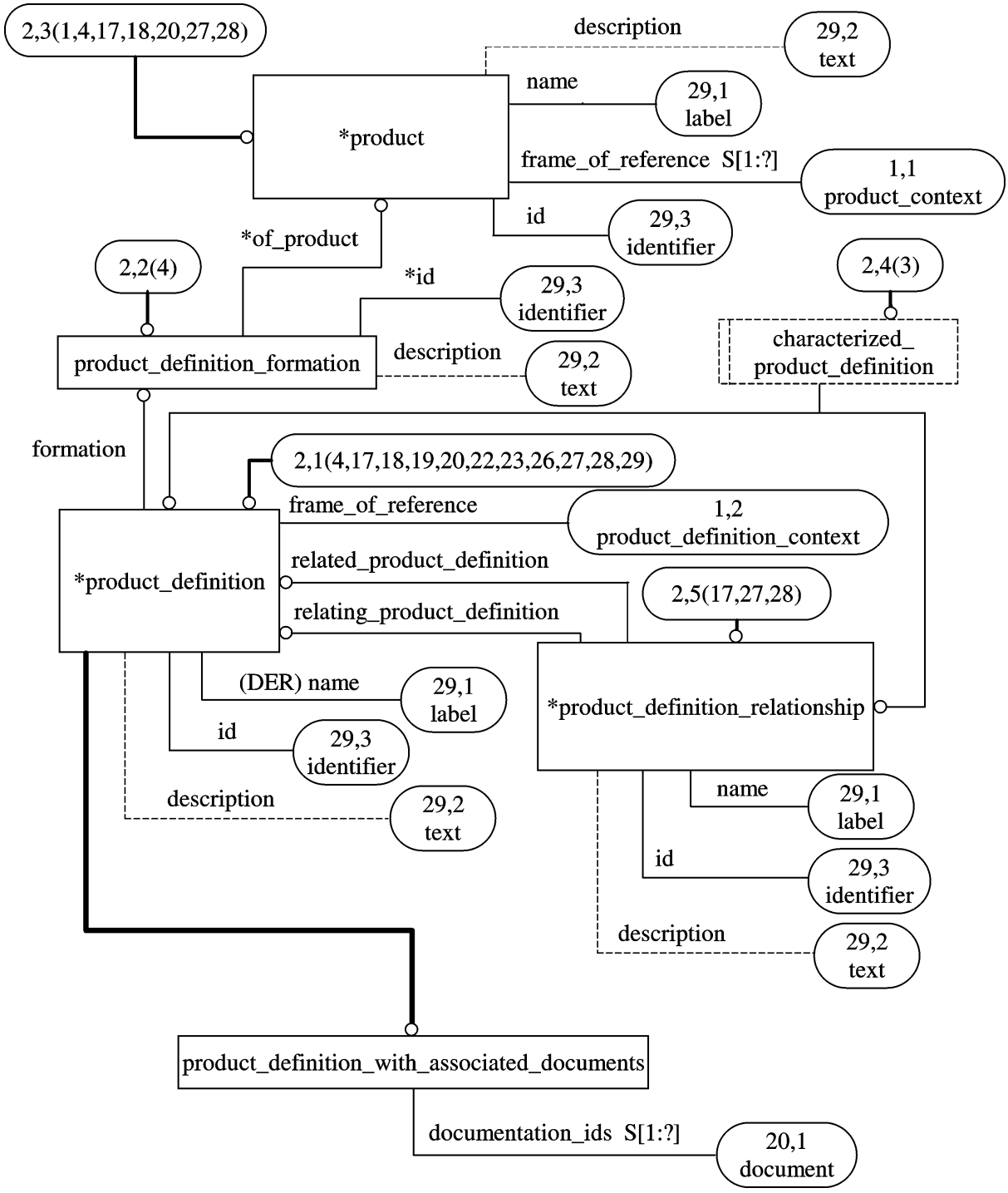

Figure H.2 — AIM EXPRESS-G diagram product definition

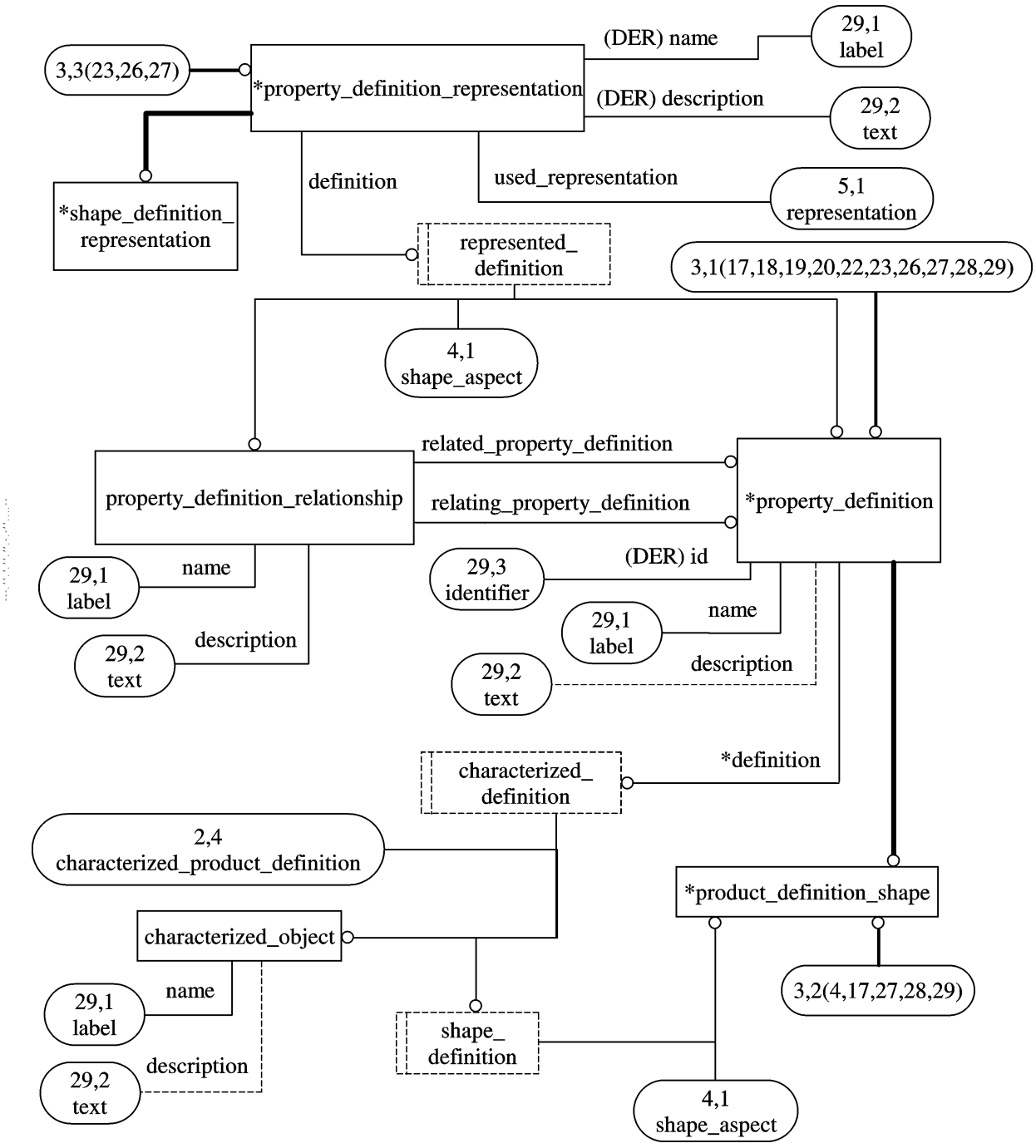

Figure H.3 — AIM EXPRESS-G diagram property definition

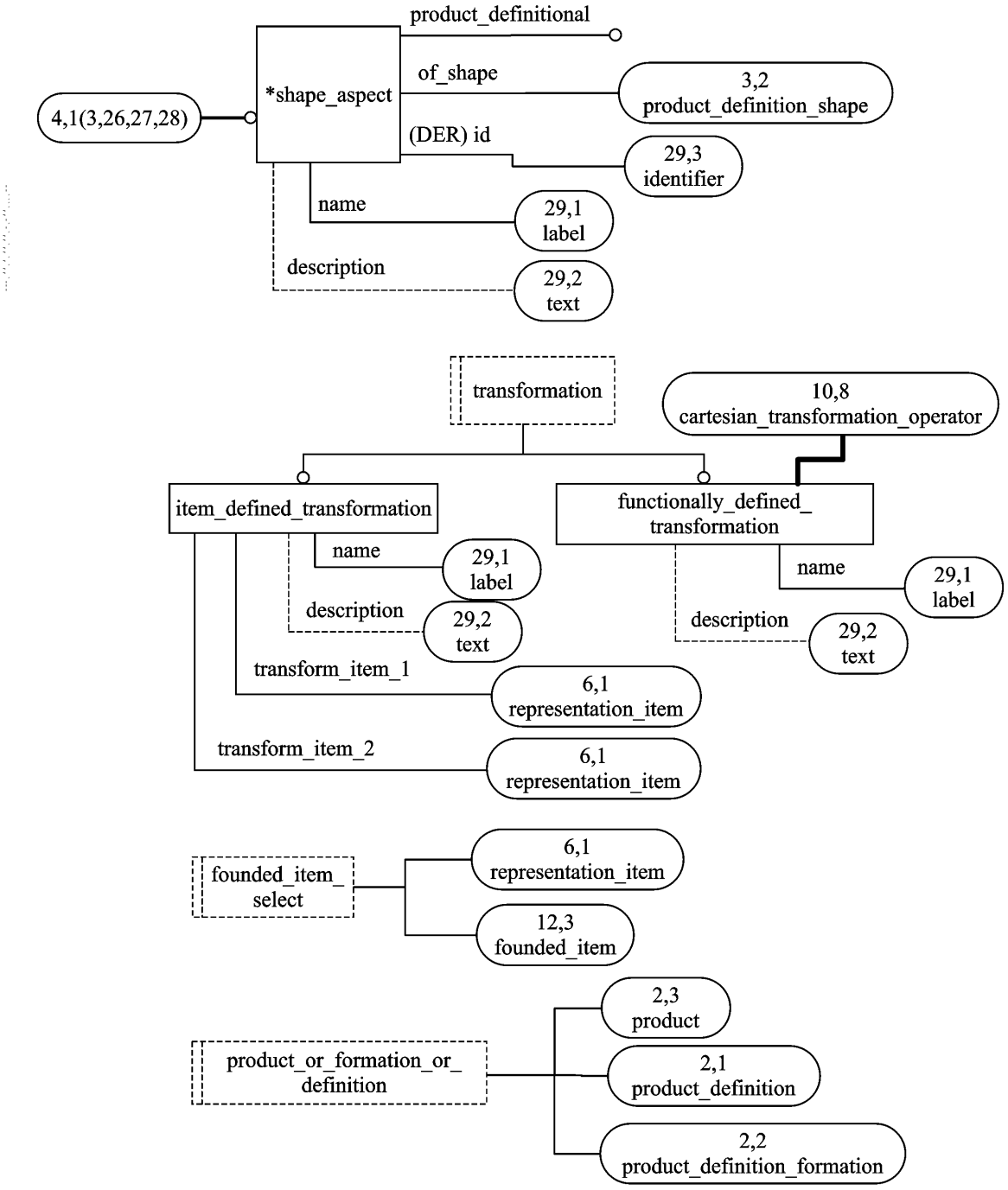

Figure H.4 — AIM EXPRESS-G diagram shape aspect

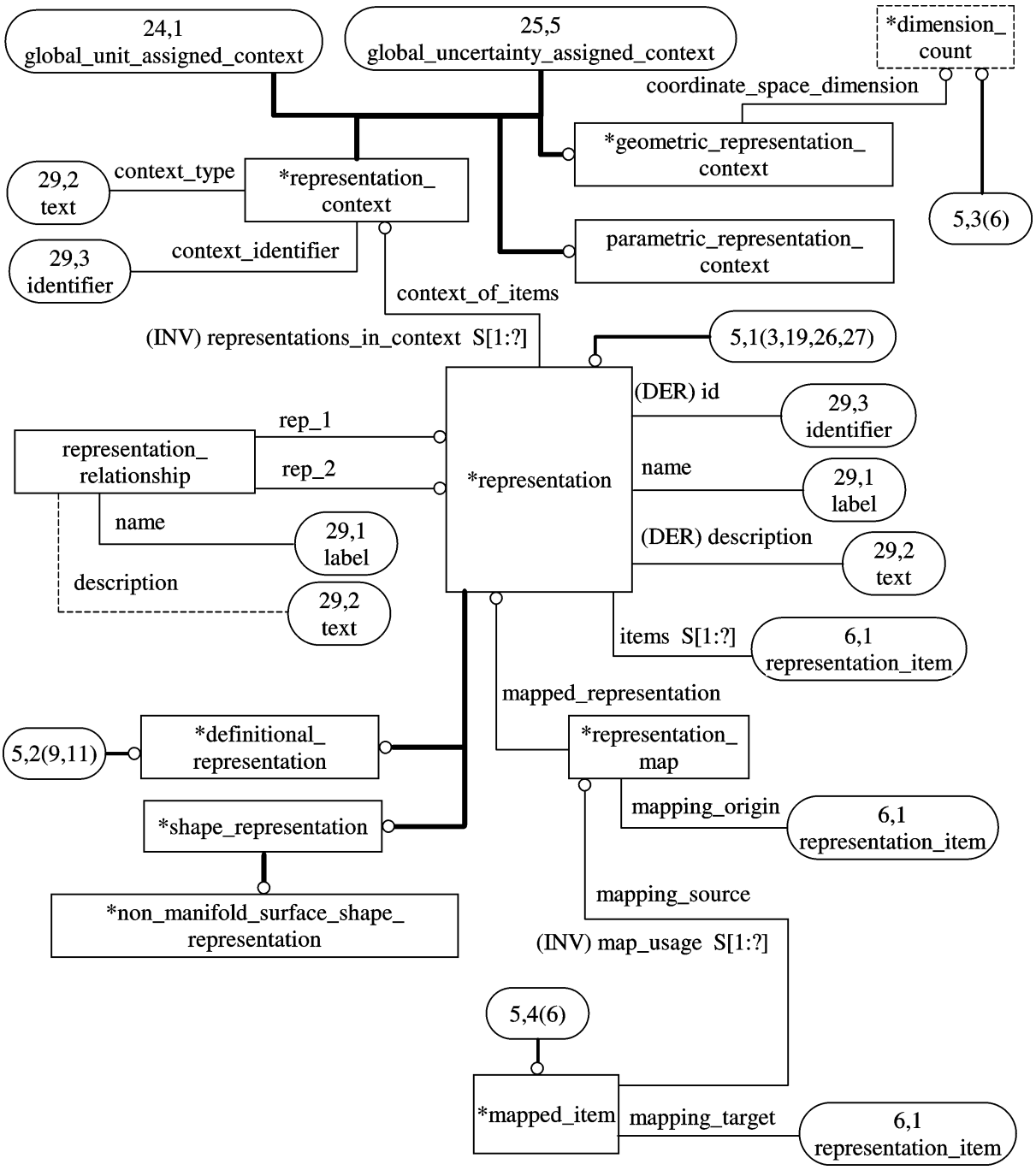

Figure H.5 — AIM EXPRESS-G diagram representation

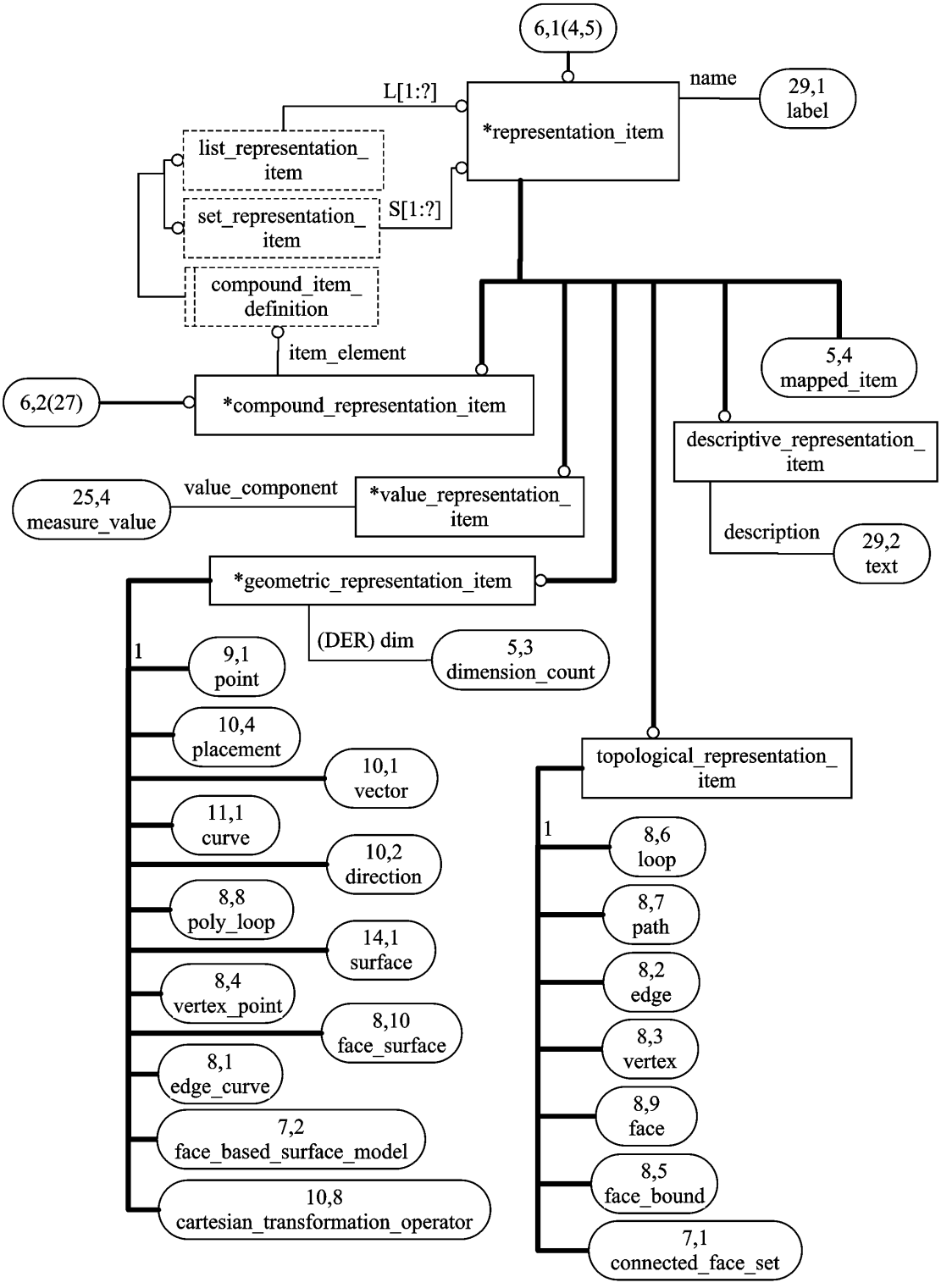

Figure H.6 — AIM EXPRESS-G diagram geometry and topology

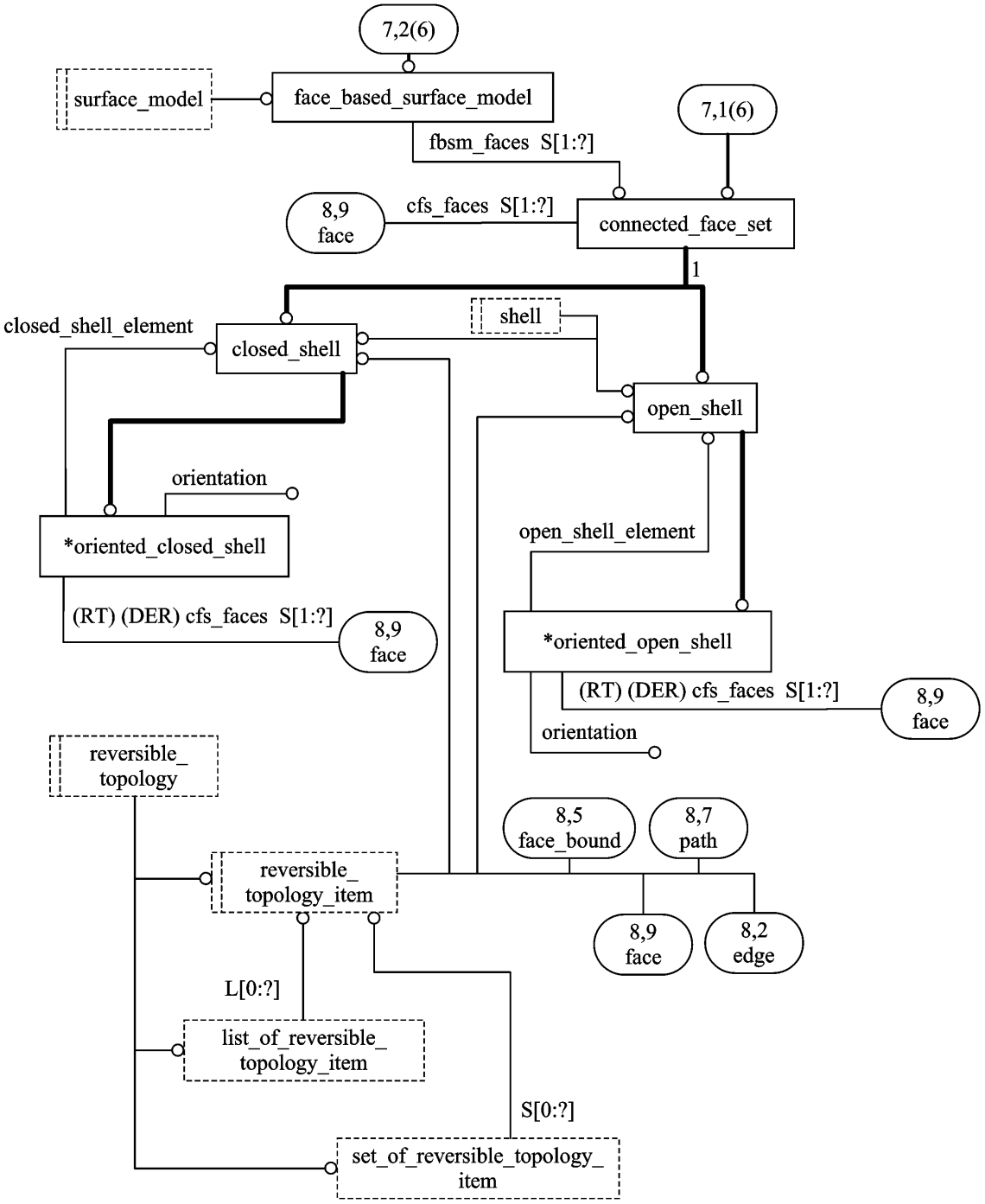

Figure H.7 — AIM EXPRESS-G diagram face based surface model

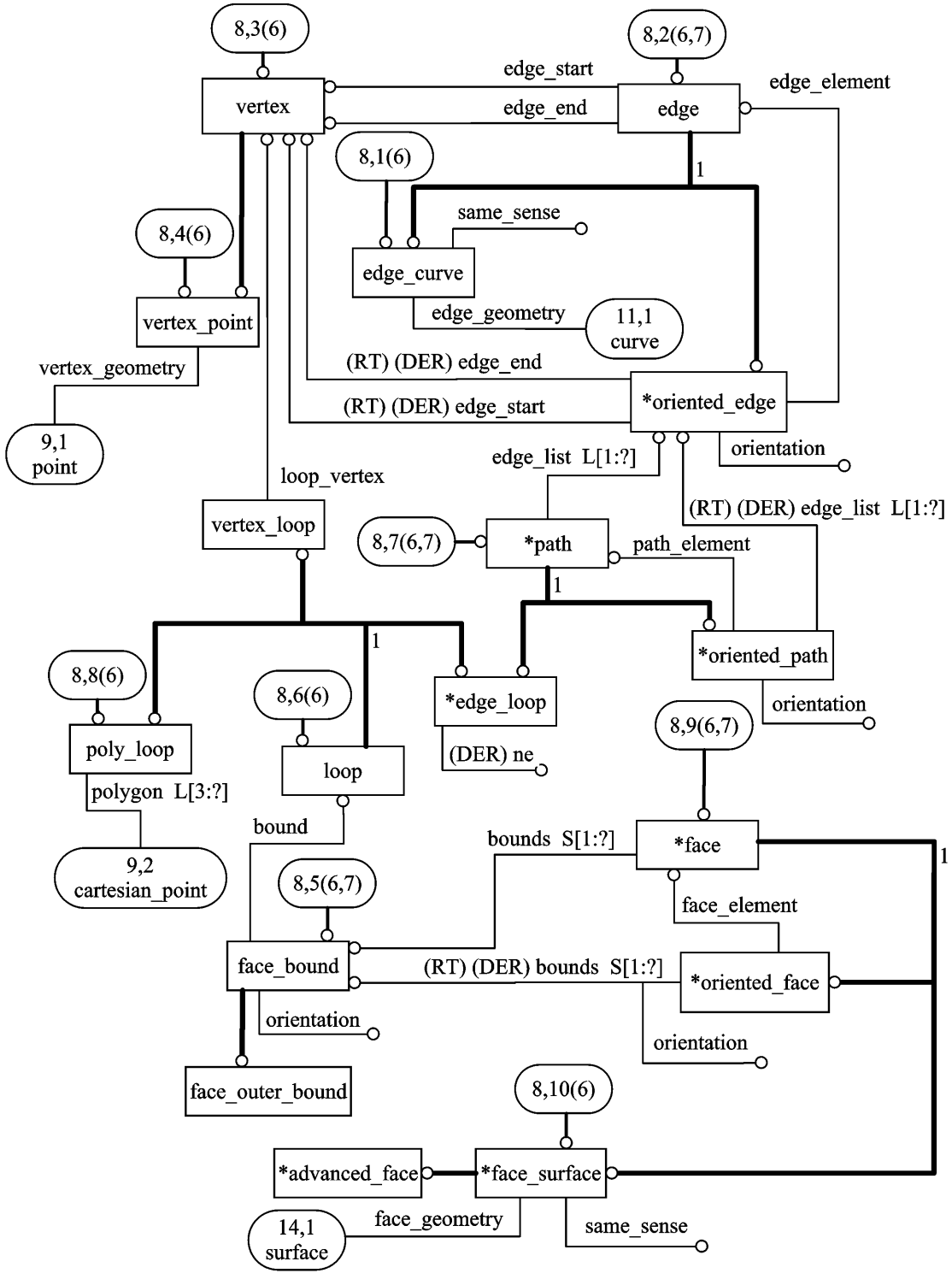

Figure H.8 — AIM EXPRESS-G diagram topology

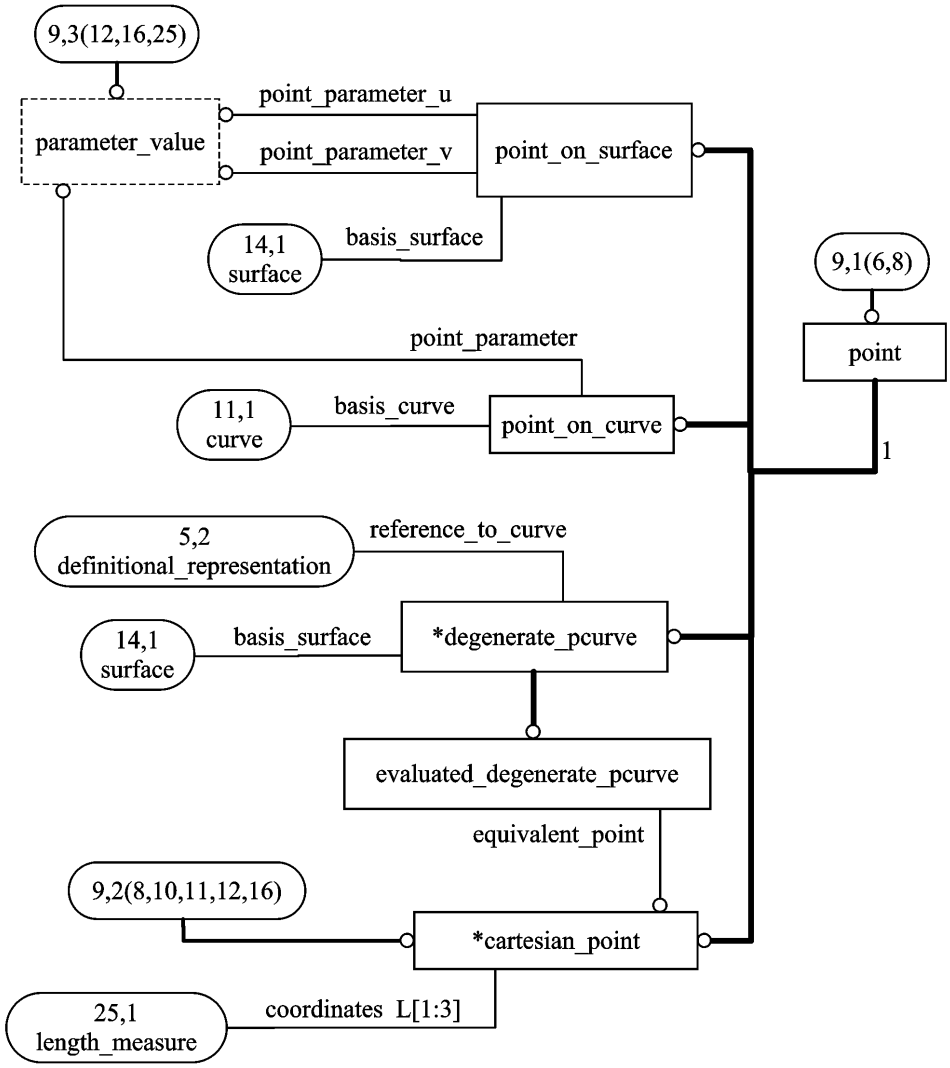

Figure H.9 — AIM EXPRESS-G diagram point

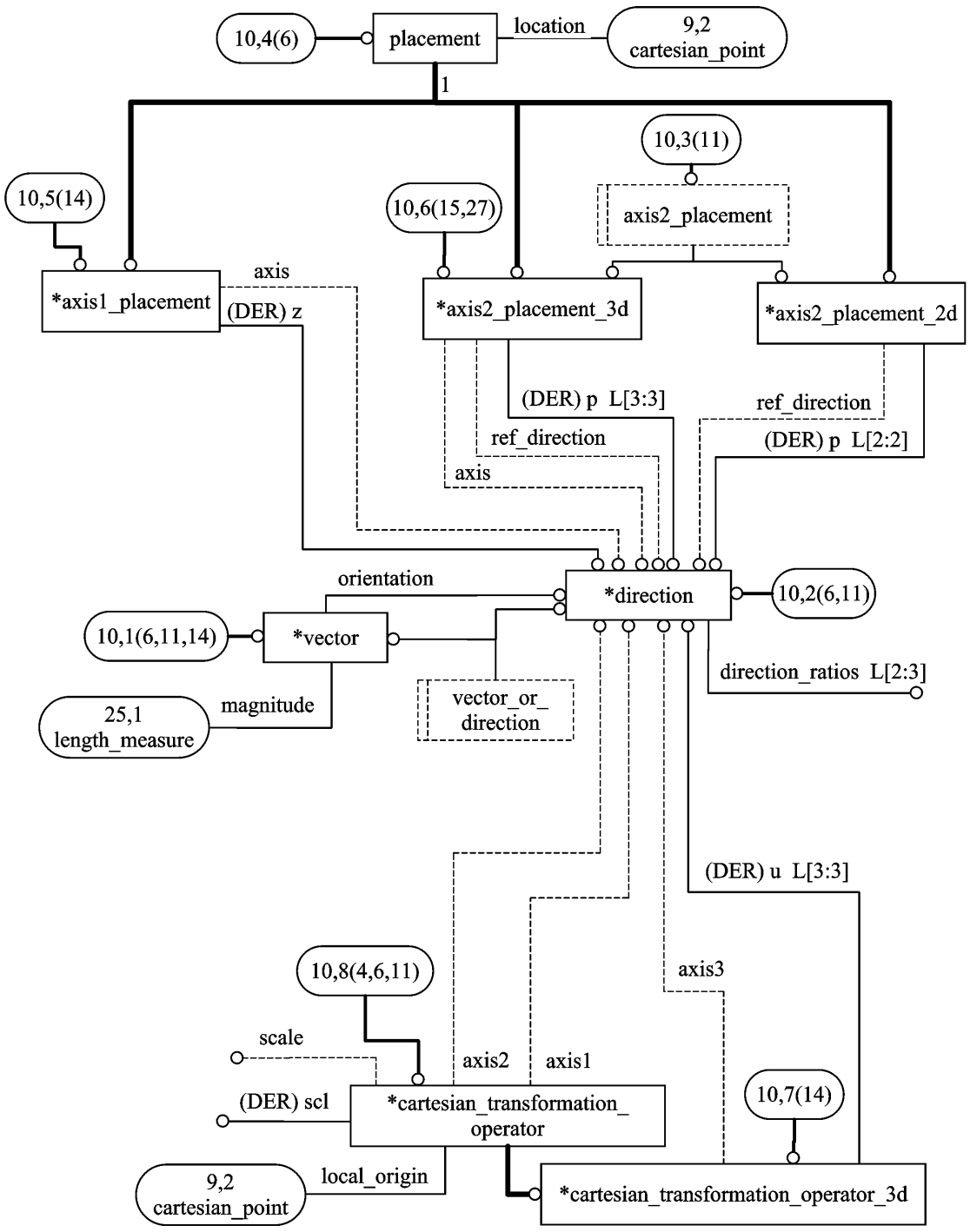

Figure H.10 — AIM EXPRESS-G diagram geometric orientation

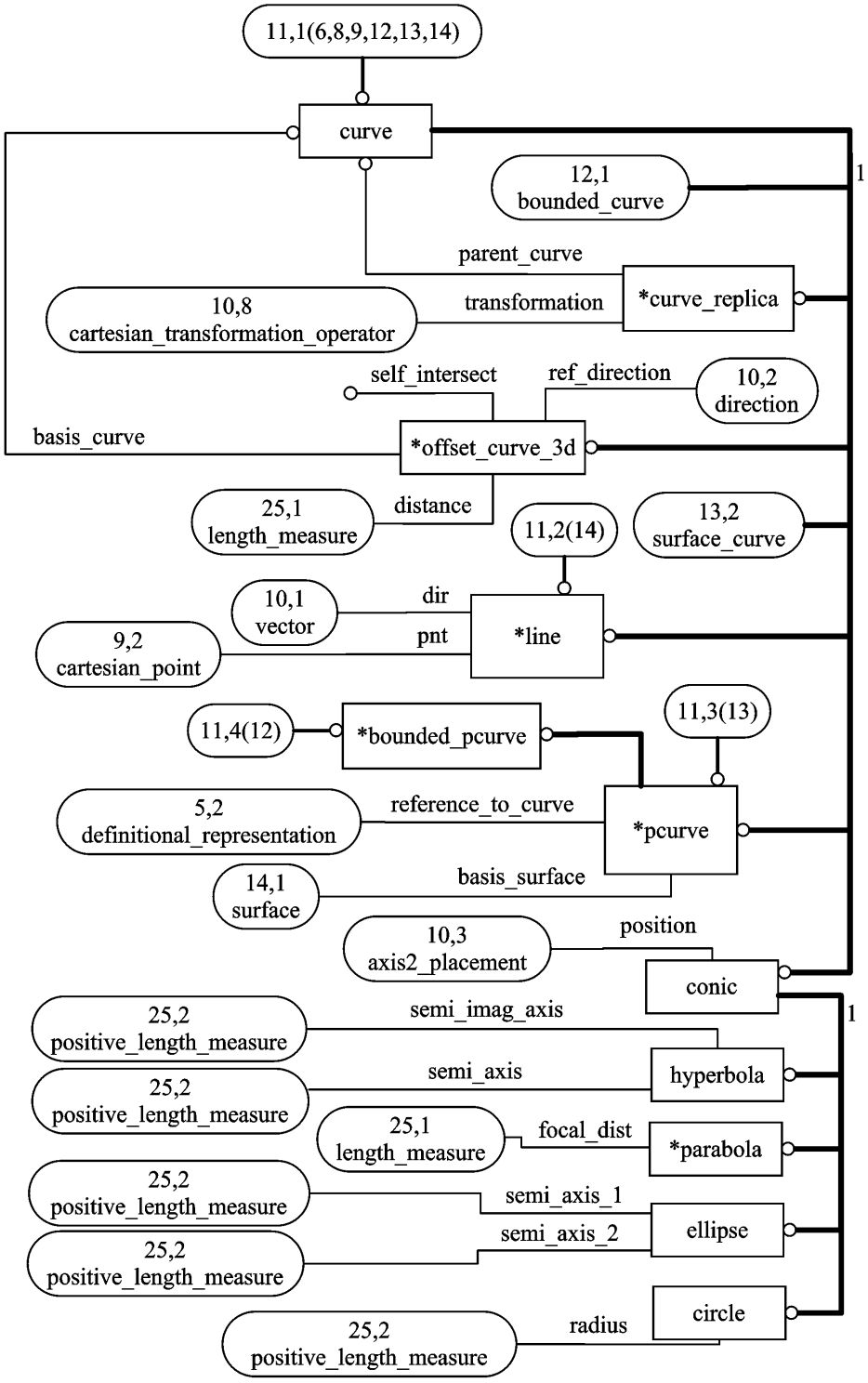

Figure H.11 — AIM EXPRESS-G diagram curve

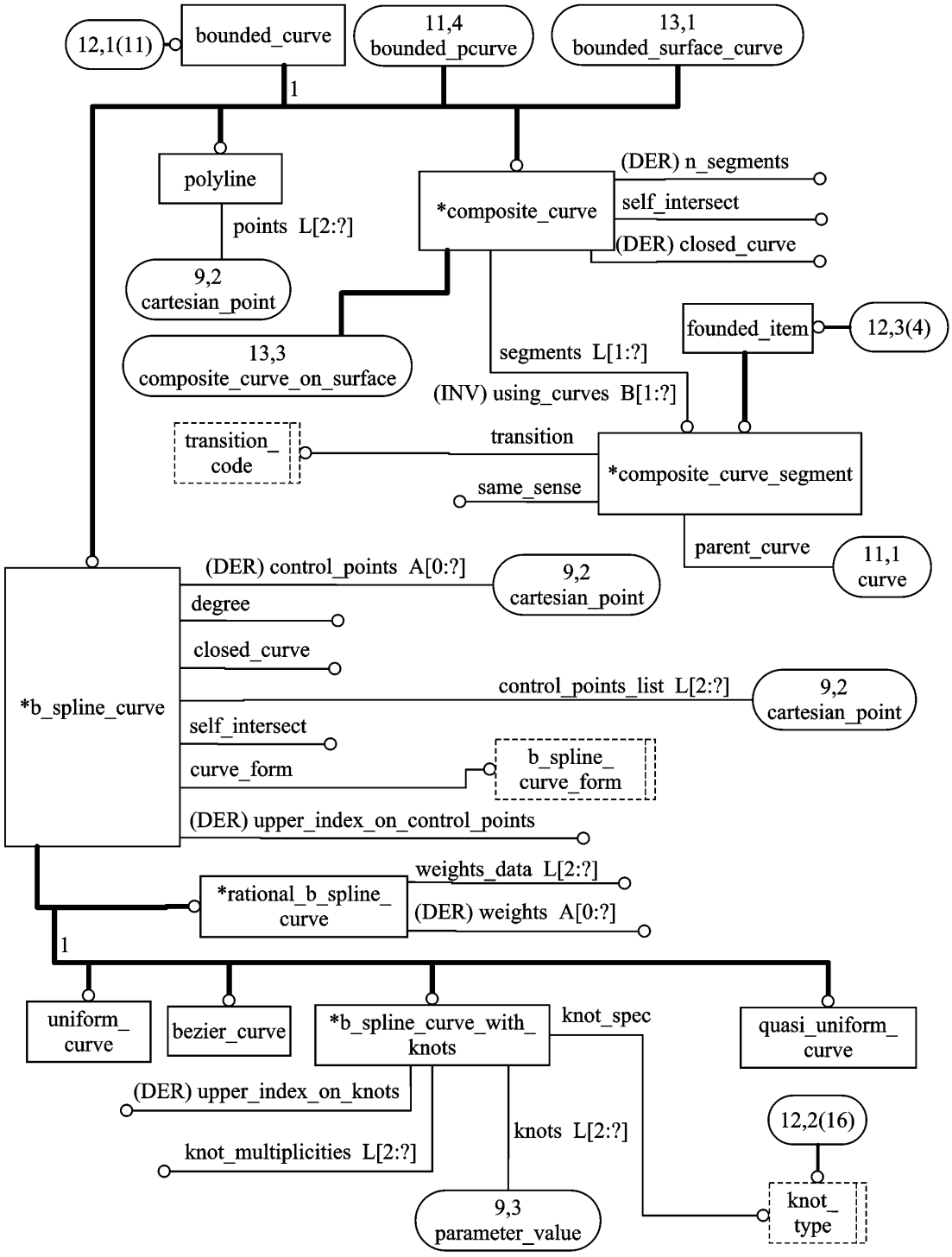

Figure H.12 — AIM EXPRESS-G diagram bounded curve

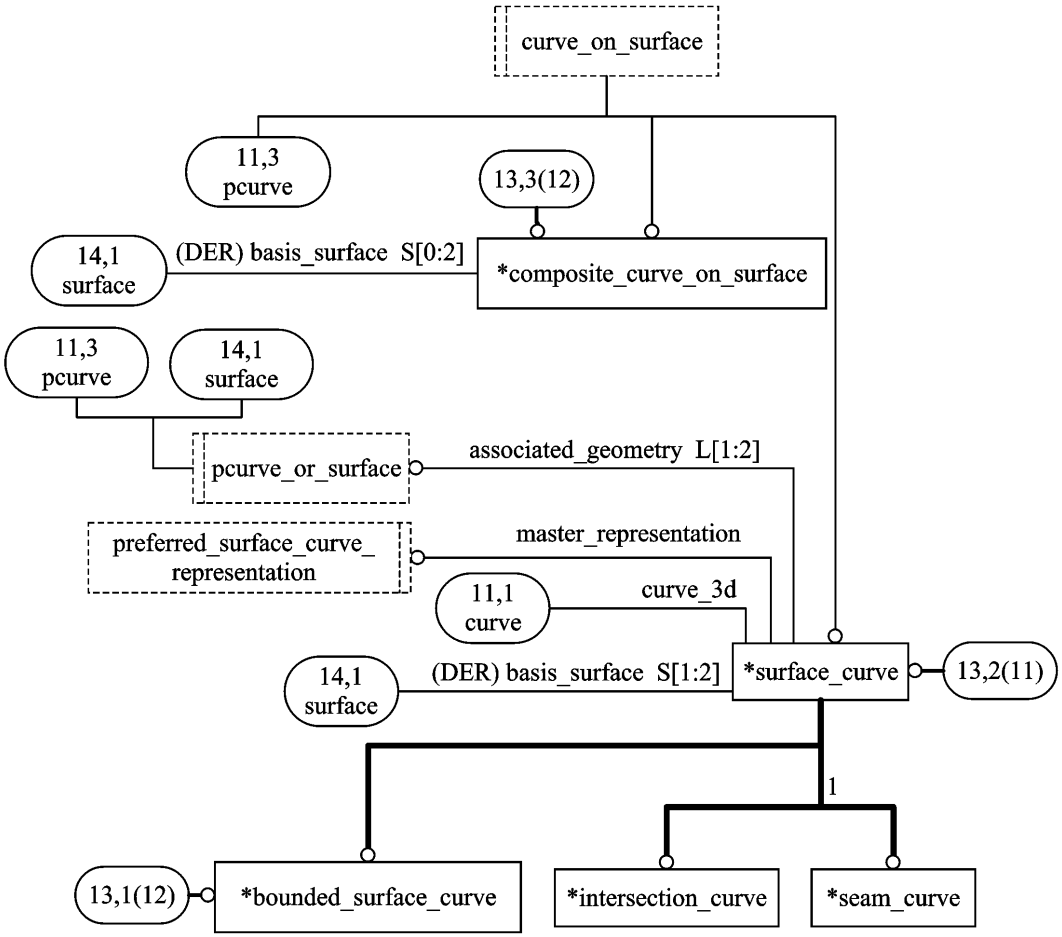

Figure H.13 — AIM EXPRESS-G diagram surface curve

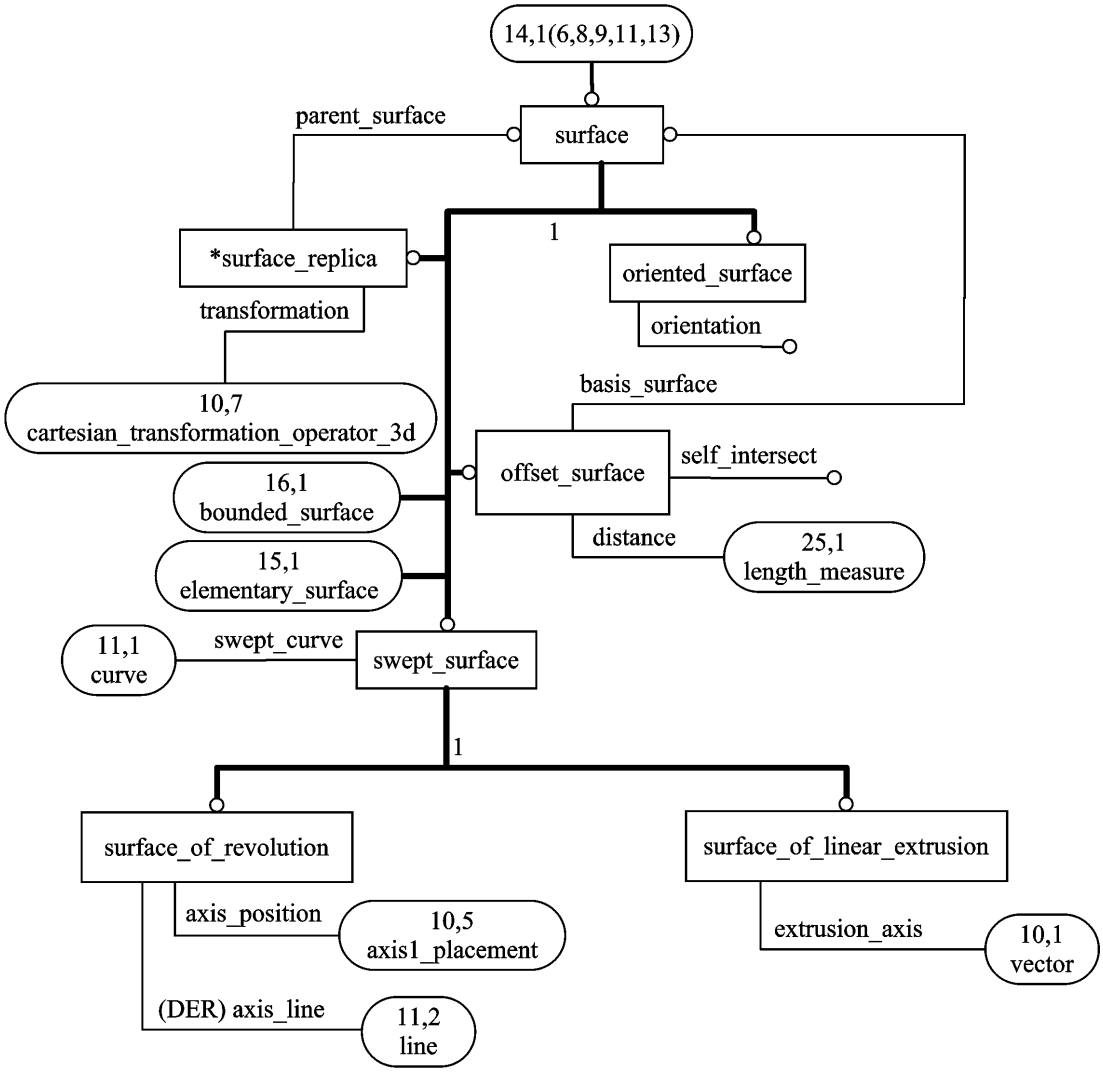

Figure H.14 — AIM EXPRESS-G diagram surface

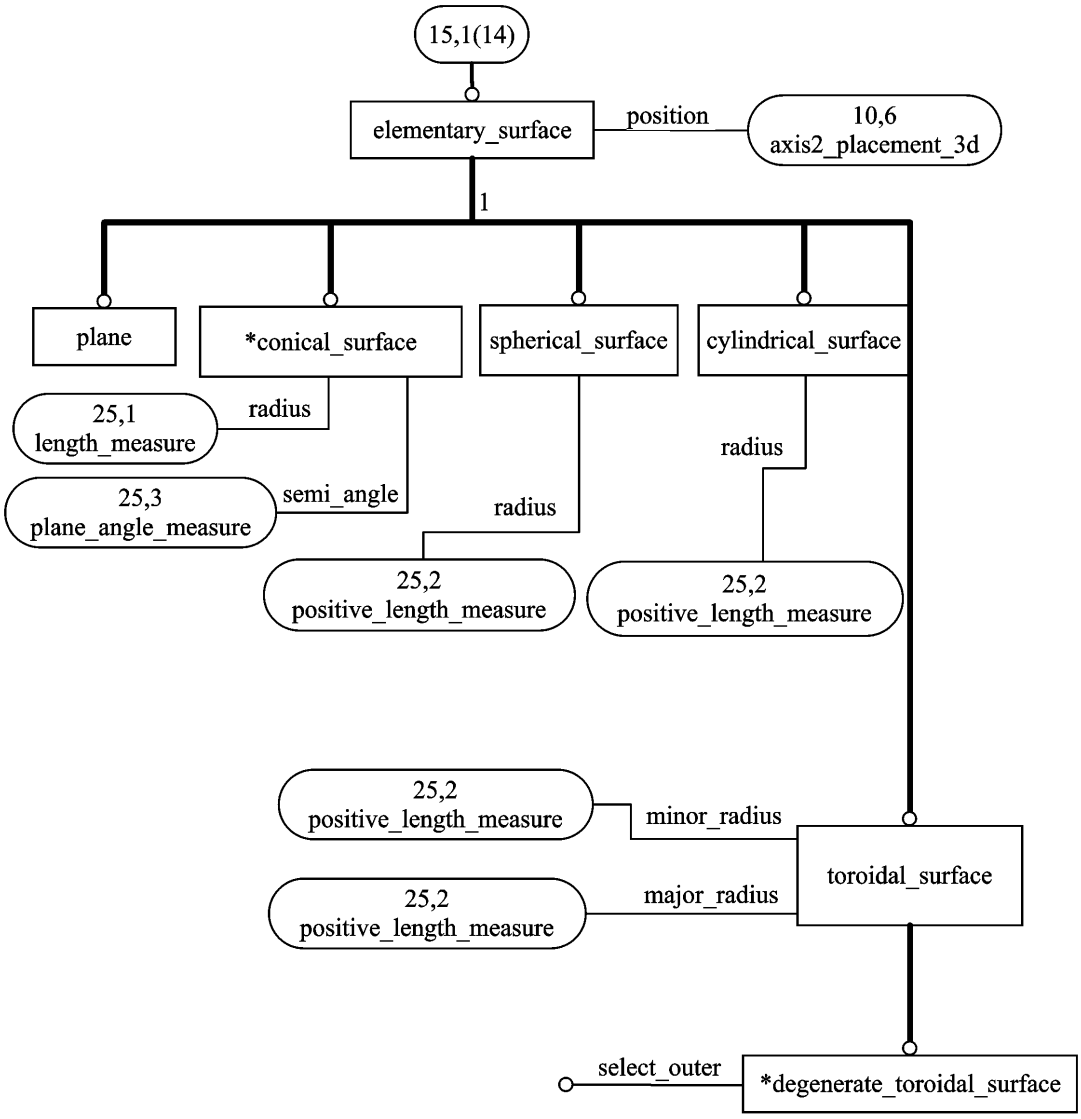

Figure H.15 — AIM EXPRESS-G diagram elementary surface

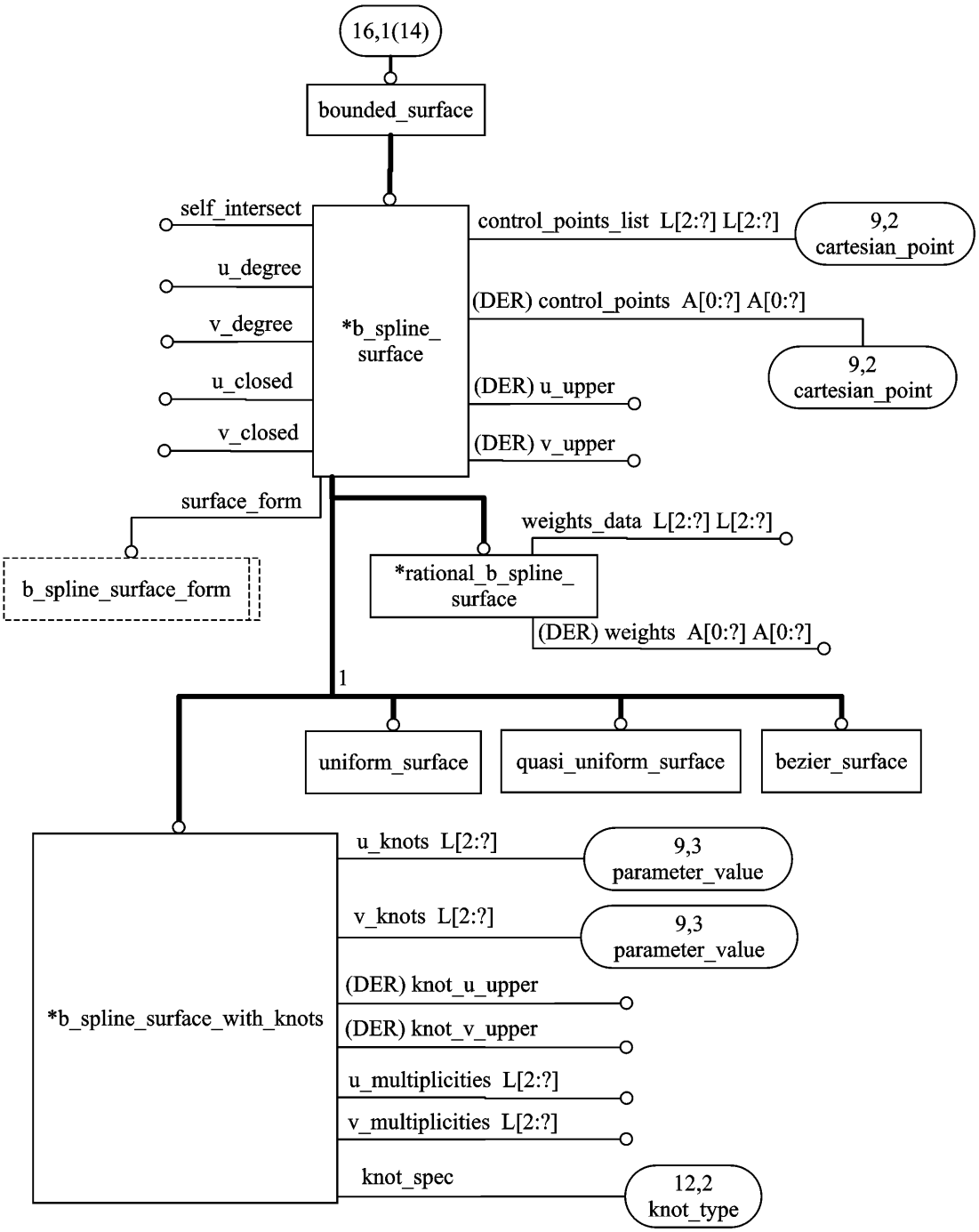

Figure H.16 — AIM EXPRESS-G diagram bounded surface

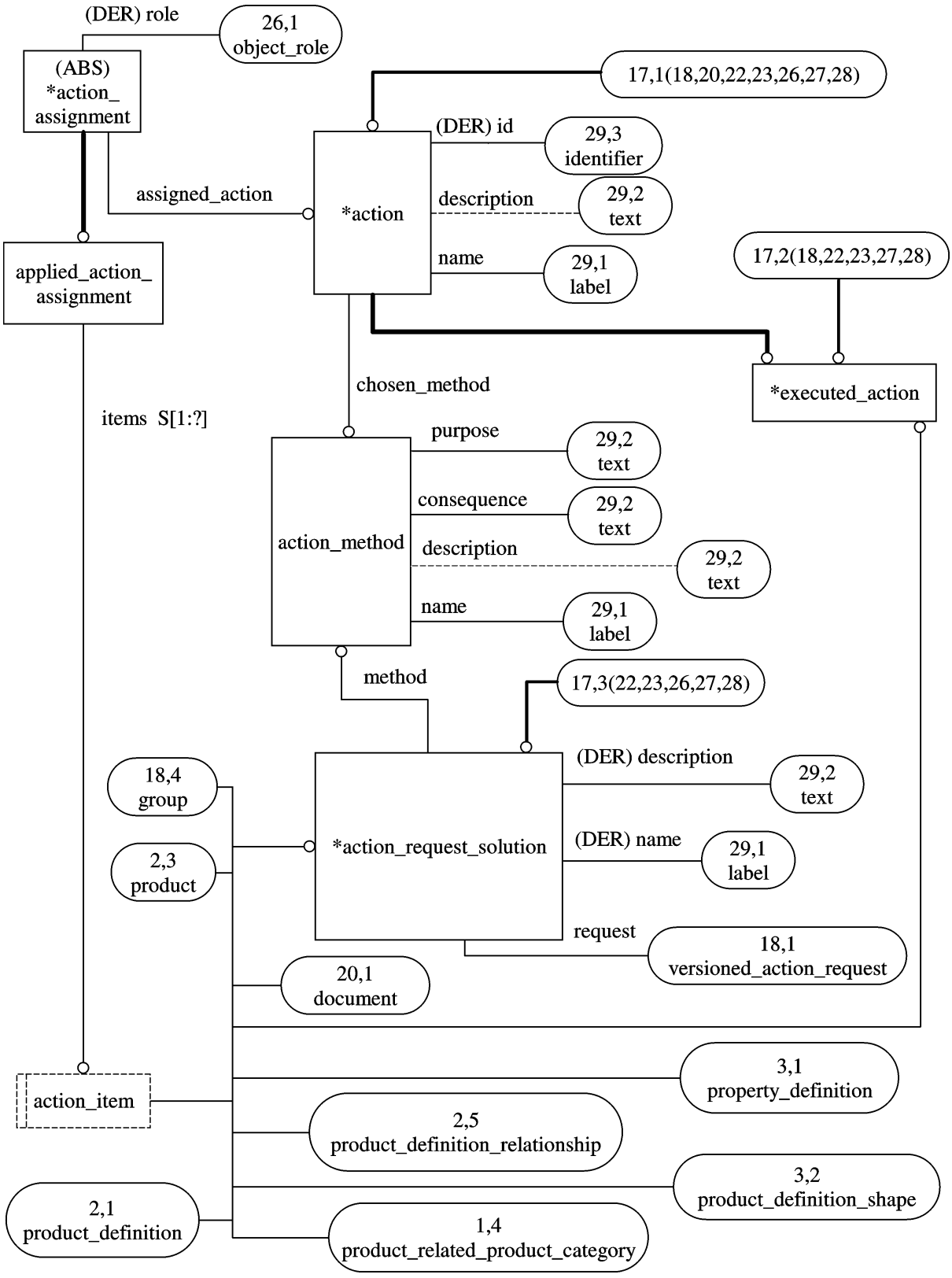

Figure H.17 — AIM EXPRESS-G diagram action

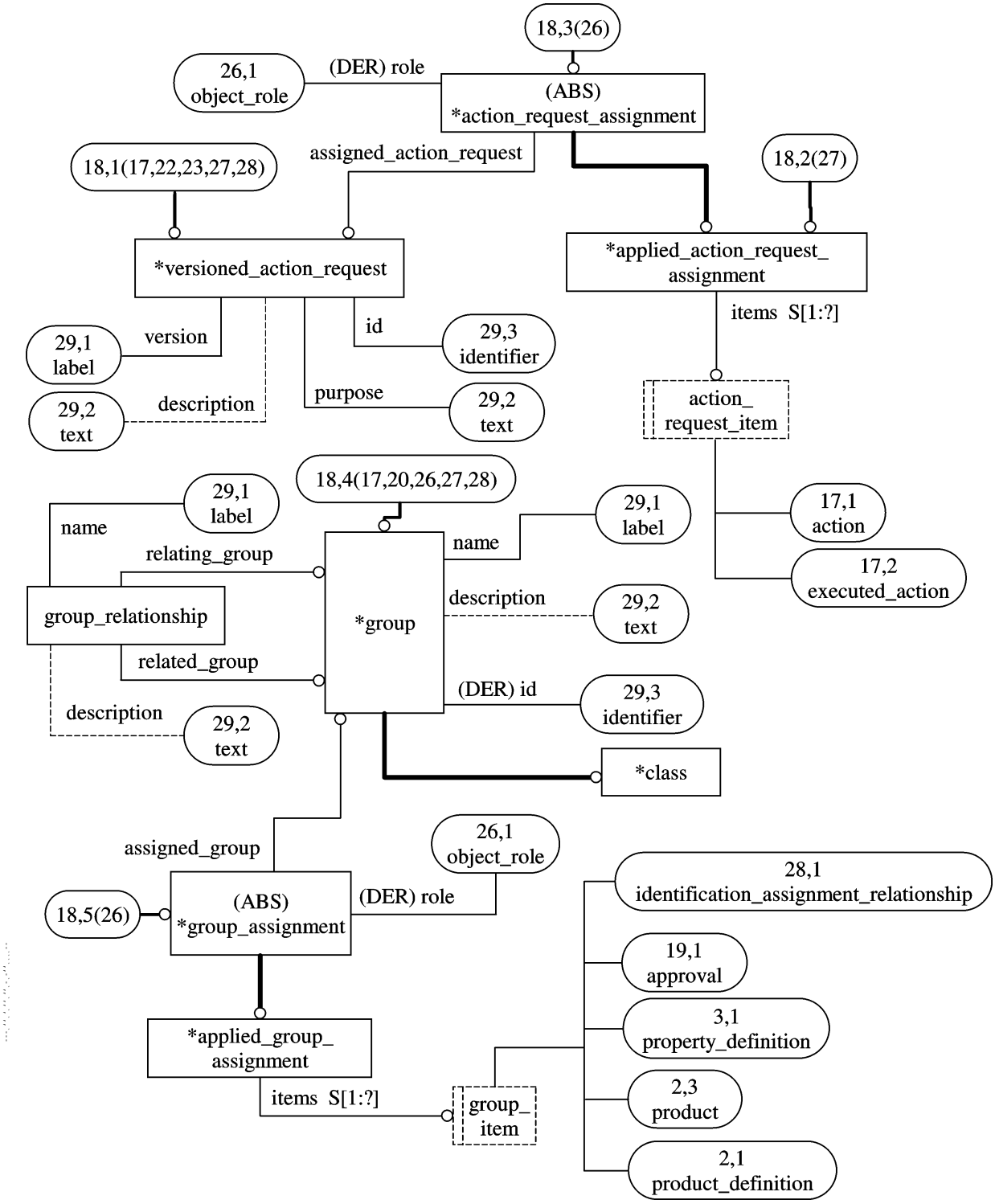

Figure H.18 — AIM EXPRESS-G diagram group

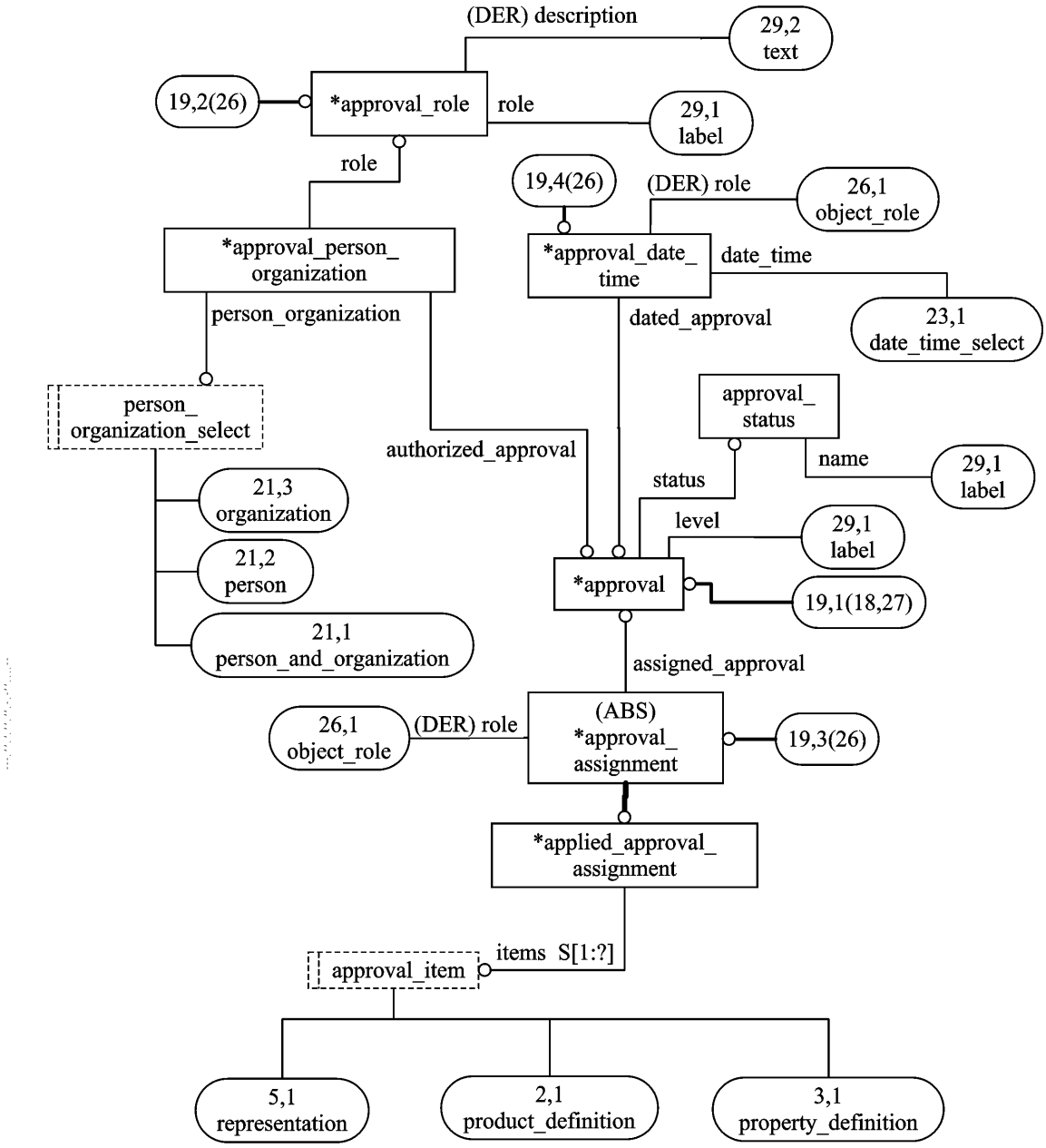

Figure H.19 — AIM EXPRESS-G diagram approval

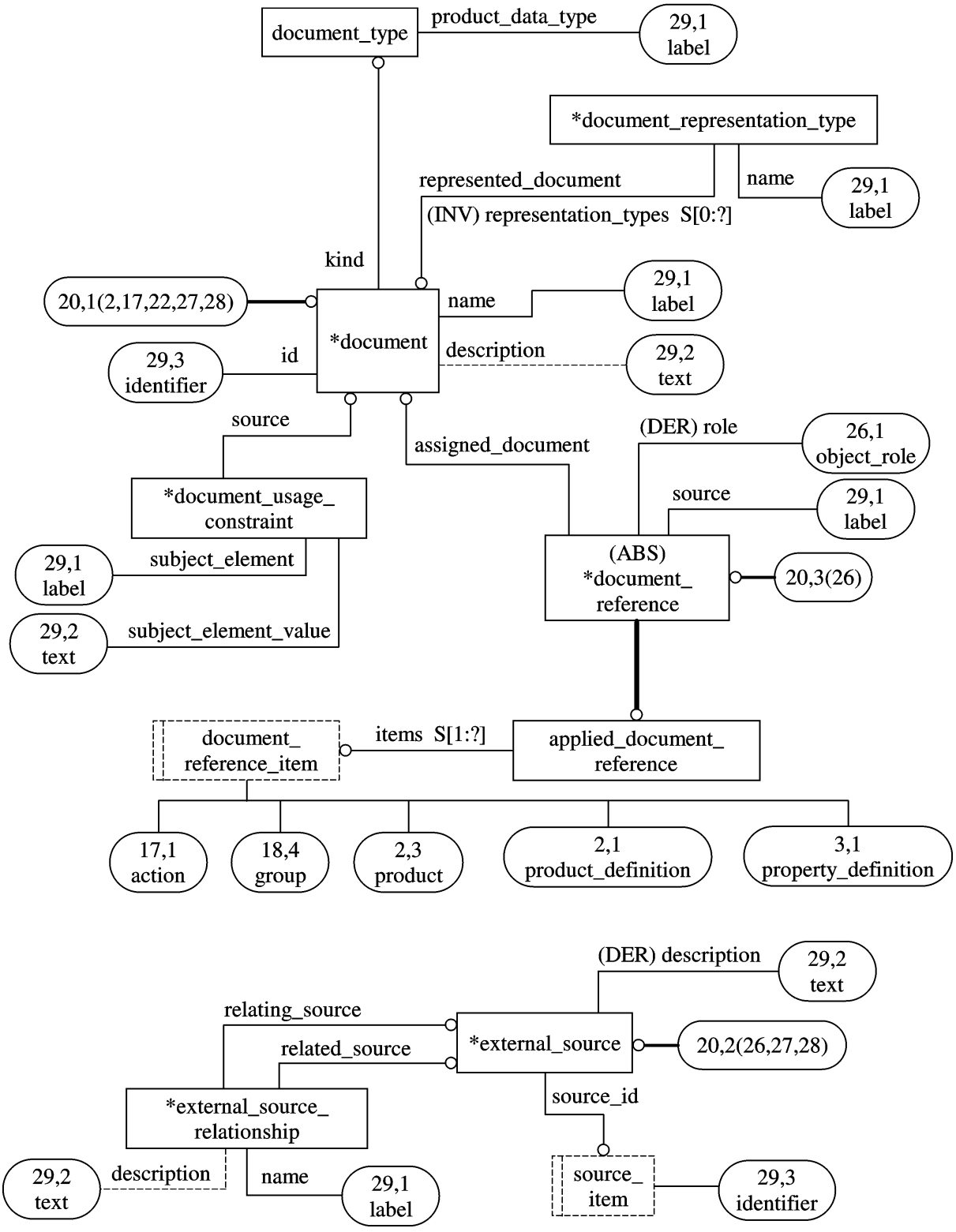

Figure H.20 — AIM EXPRESS-G diagram document

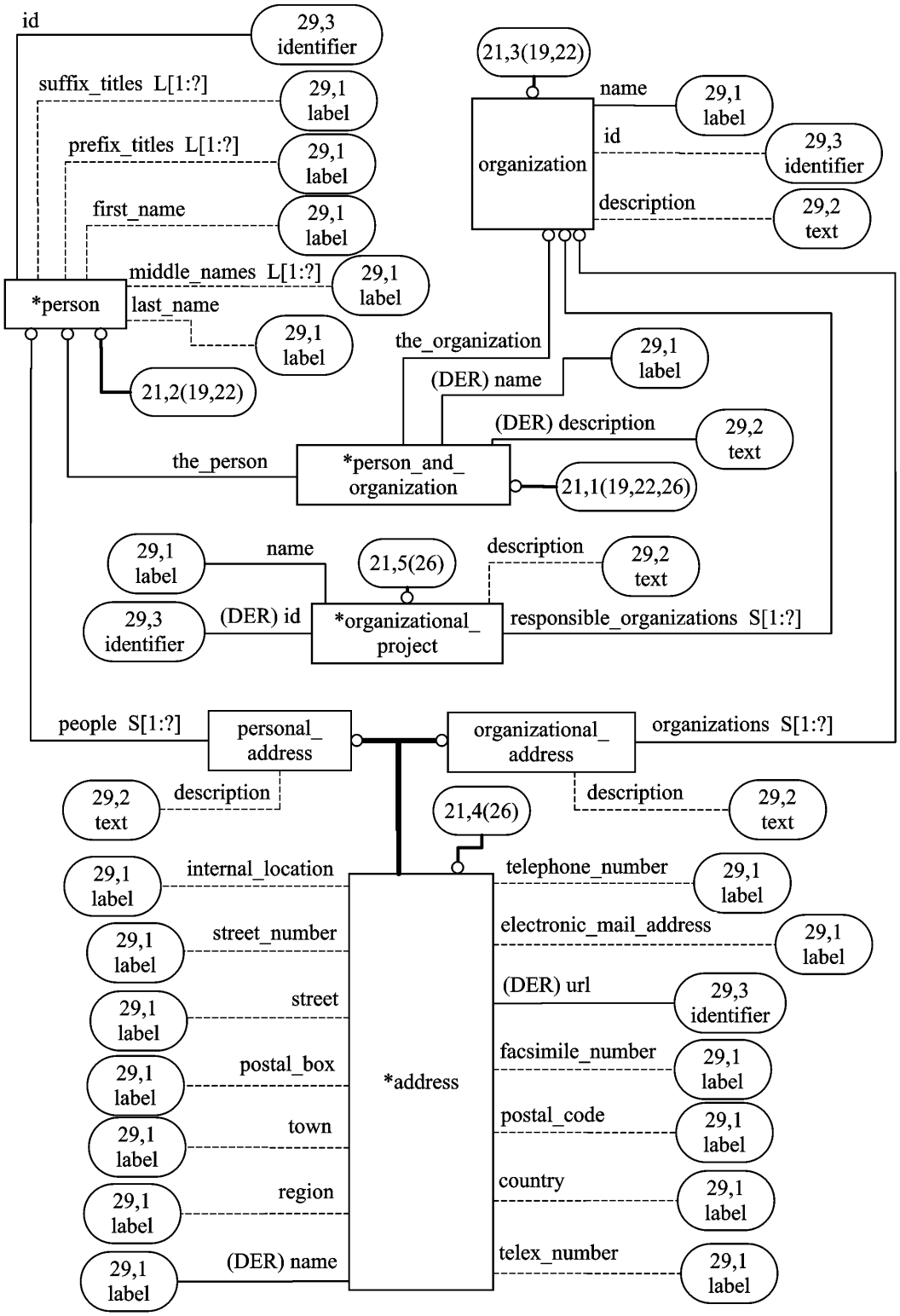

Figure H.21 — AIM EXPRESS-G diagram person and organization

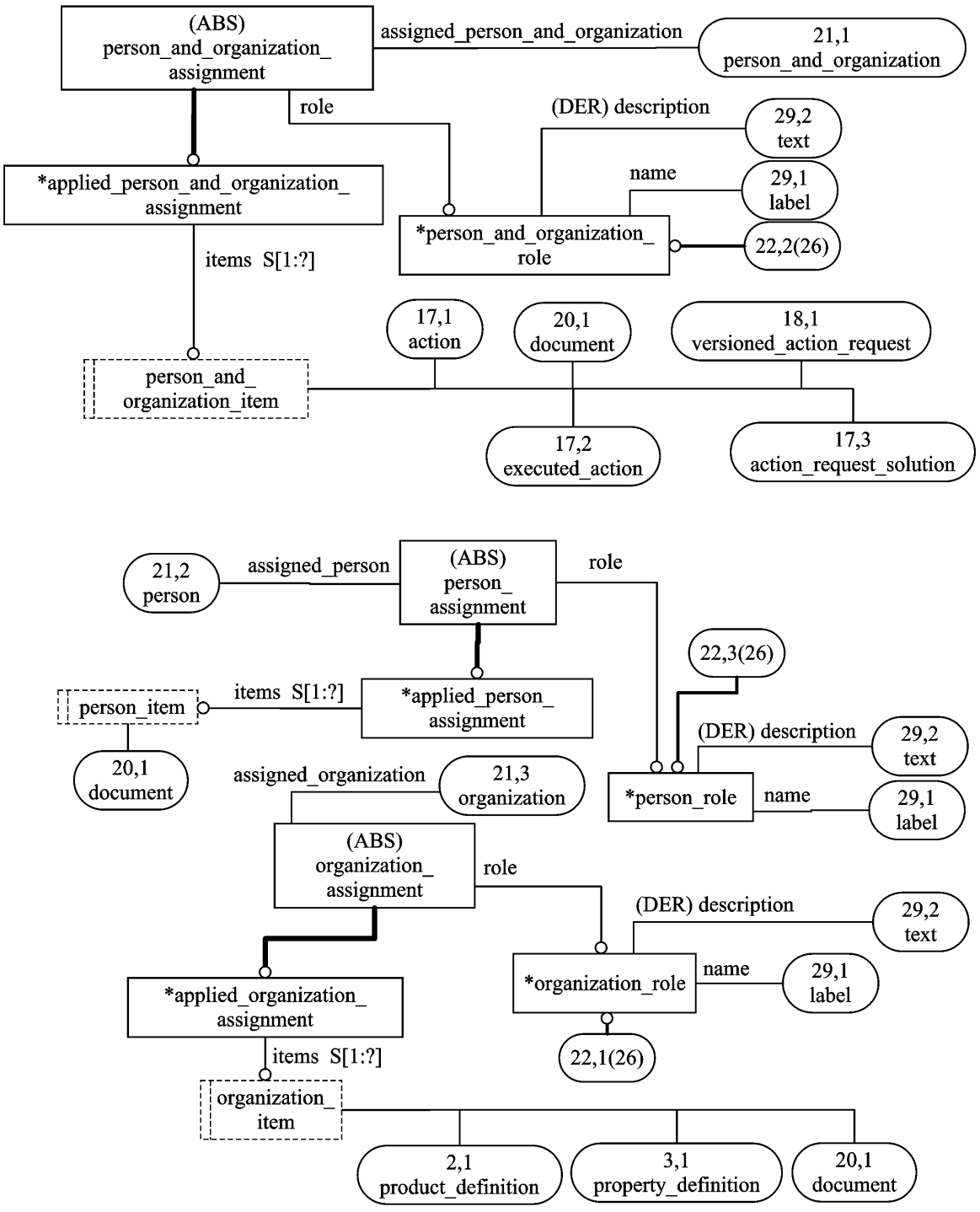

Figure H.22 — AIM EXPRESS-G diagram person and organization assignment

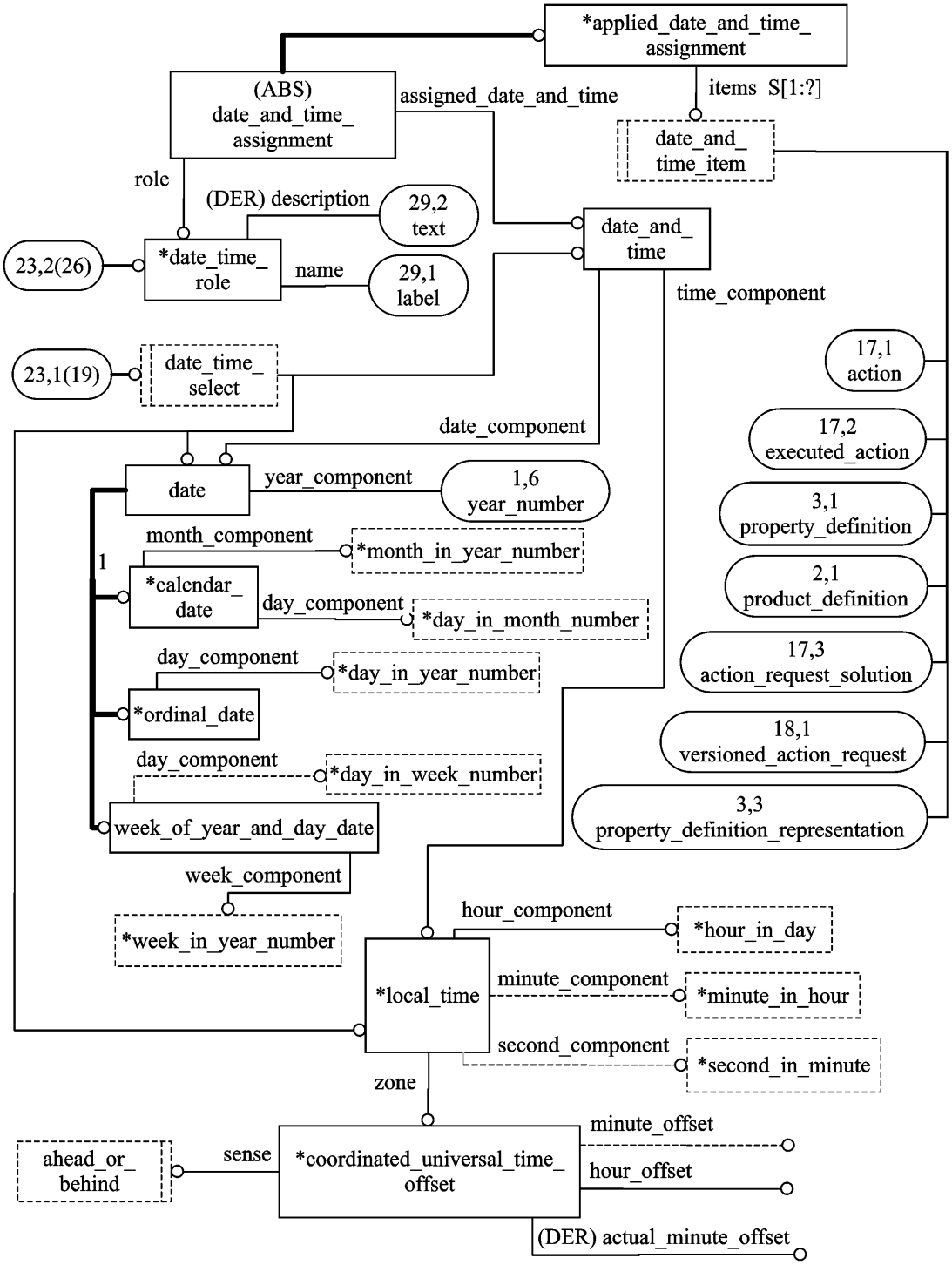

Figure H.23 — AIM EXPRESS-G diagram date and time

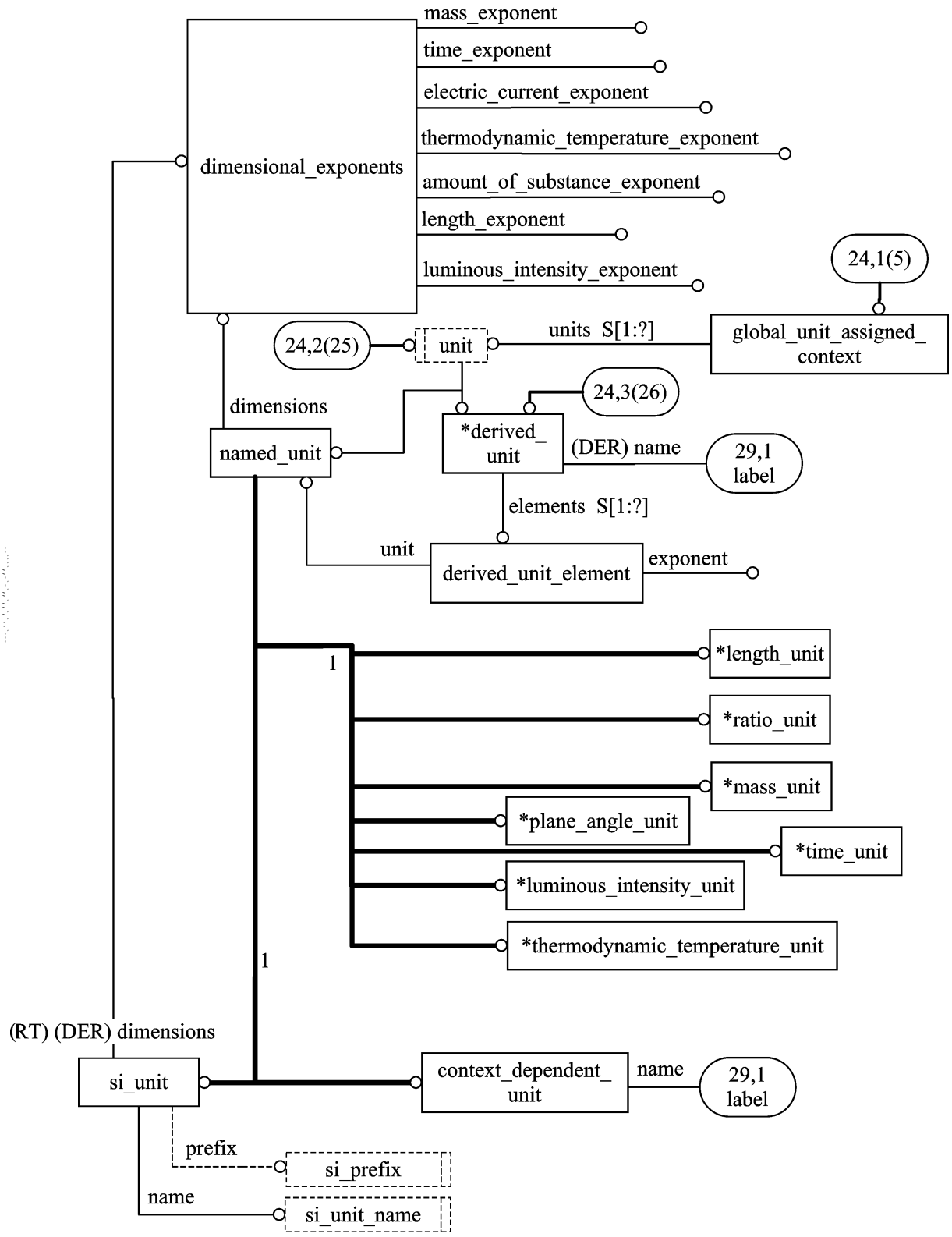

Figure H.24 — AIM EXPRESS-G diagram units

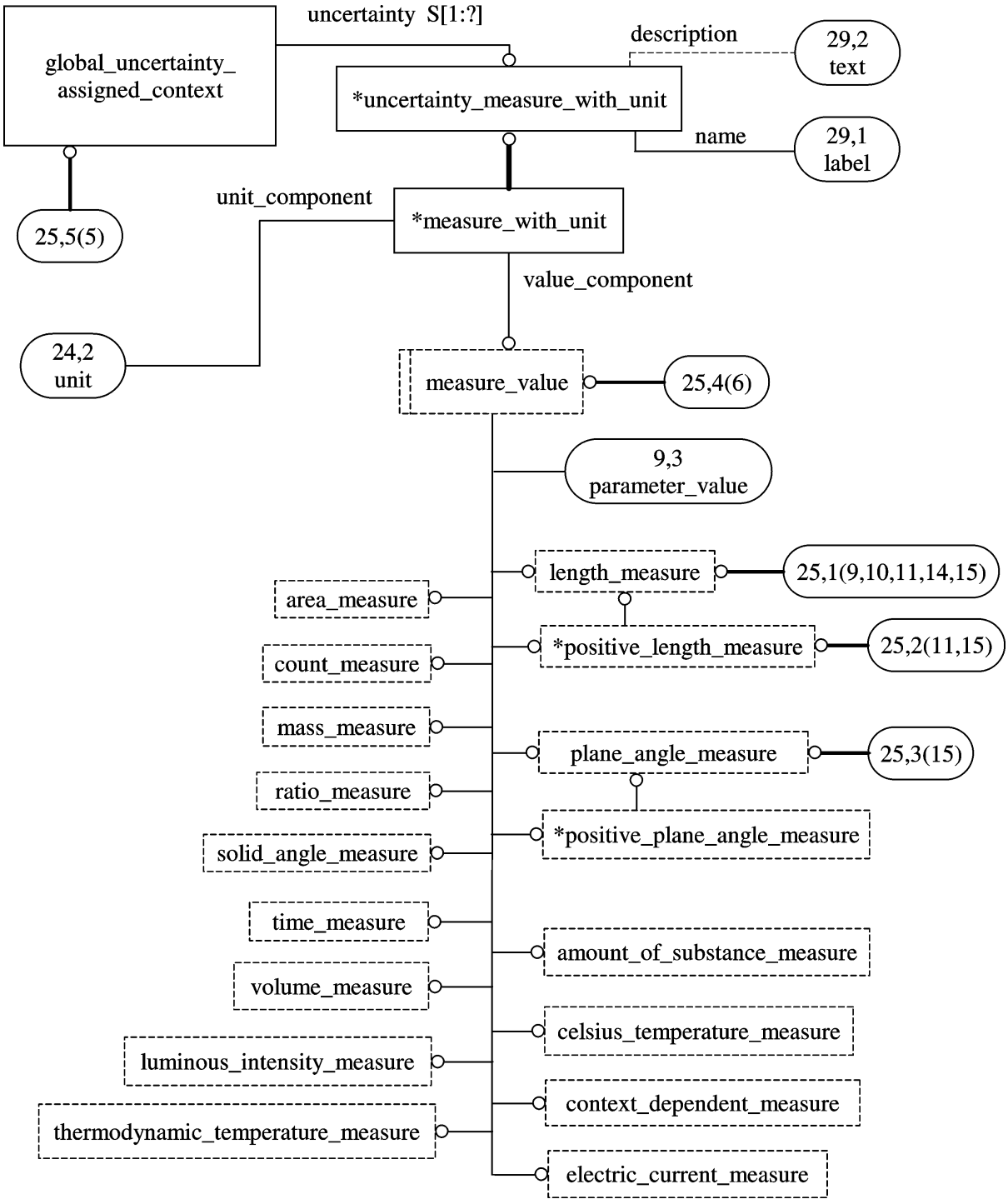

Figure H.25 — AIM EXPRESS-G diagram measures

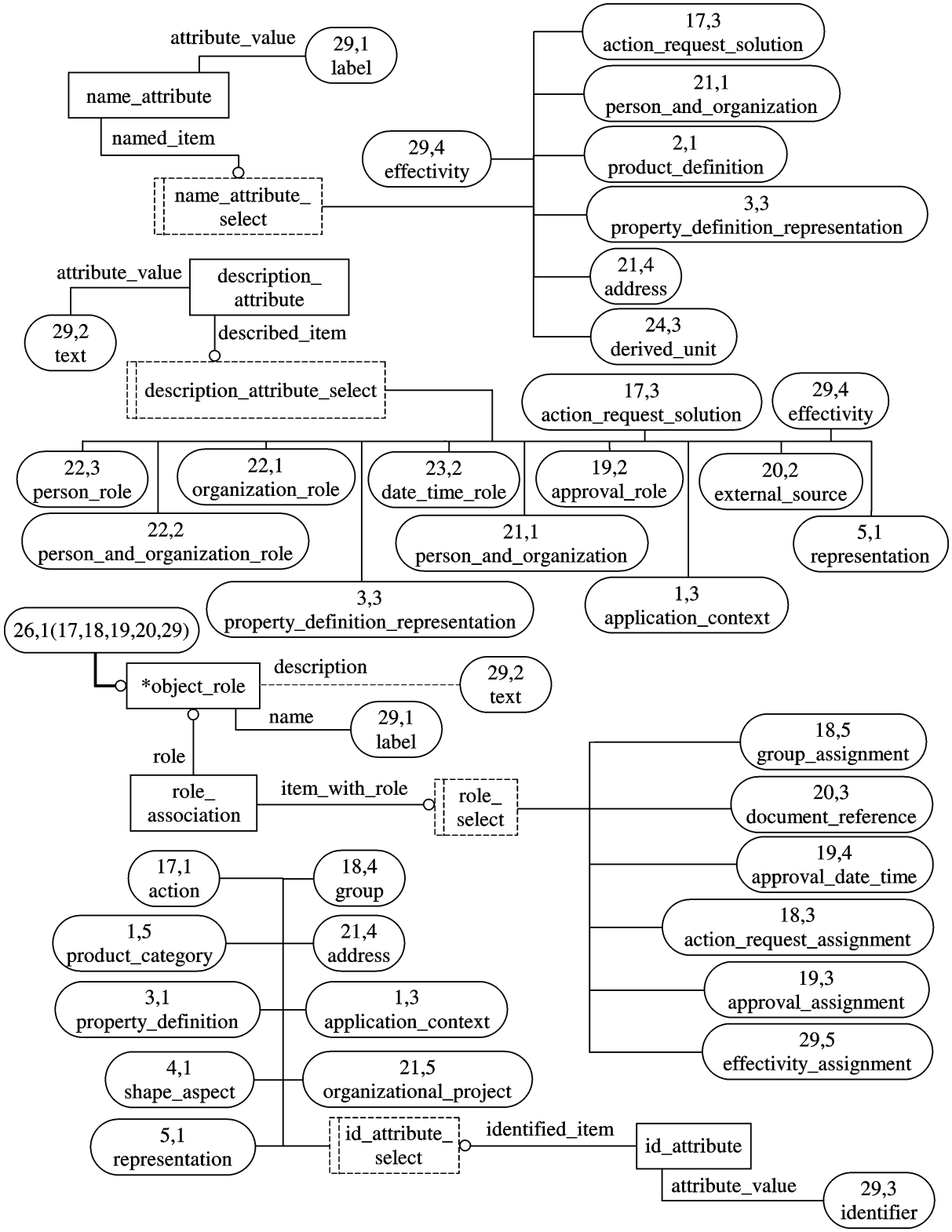

Figure H.26 — AIM EXPRESS-G diagram associations and attributes

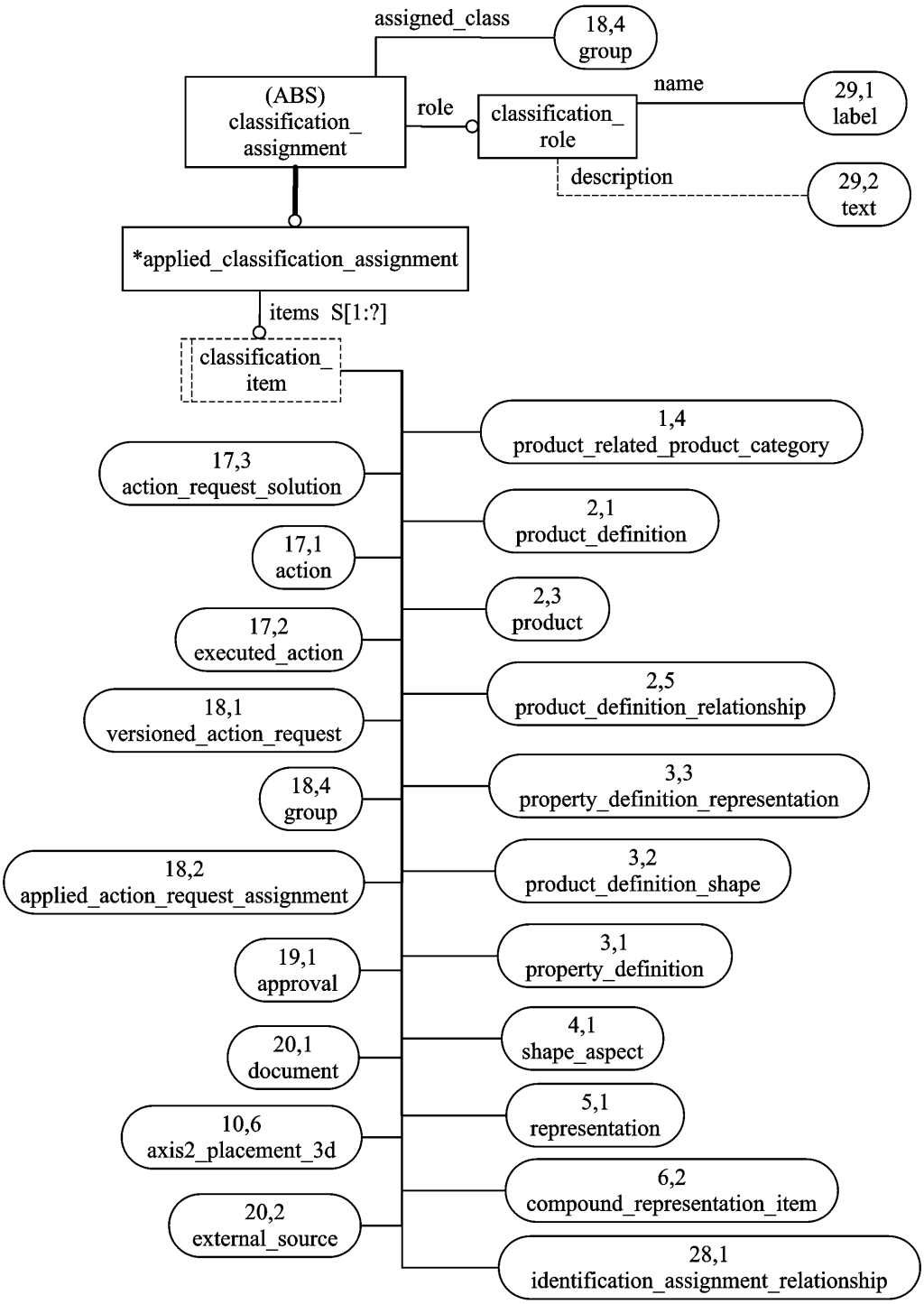

Figure H.27 — AIM EXPRESS-G diagram classification assignment

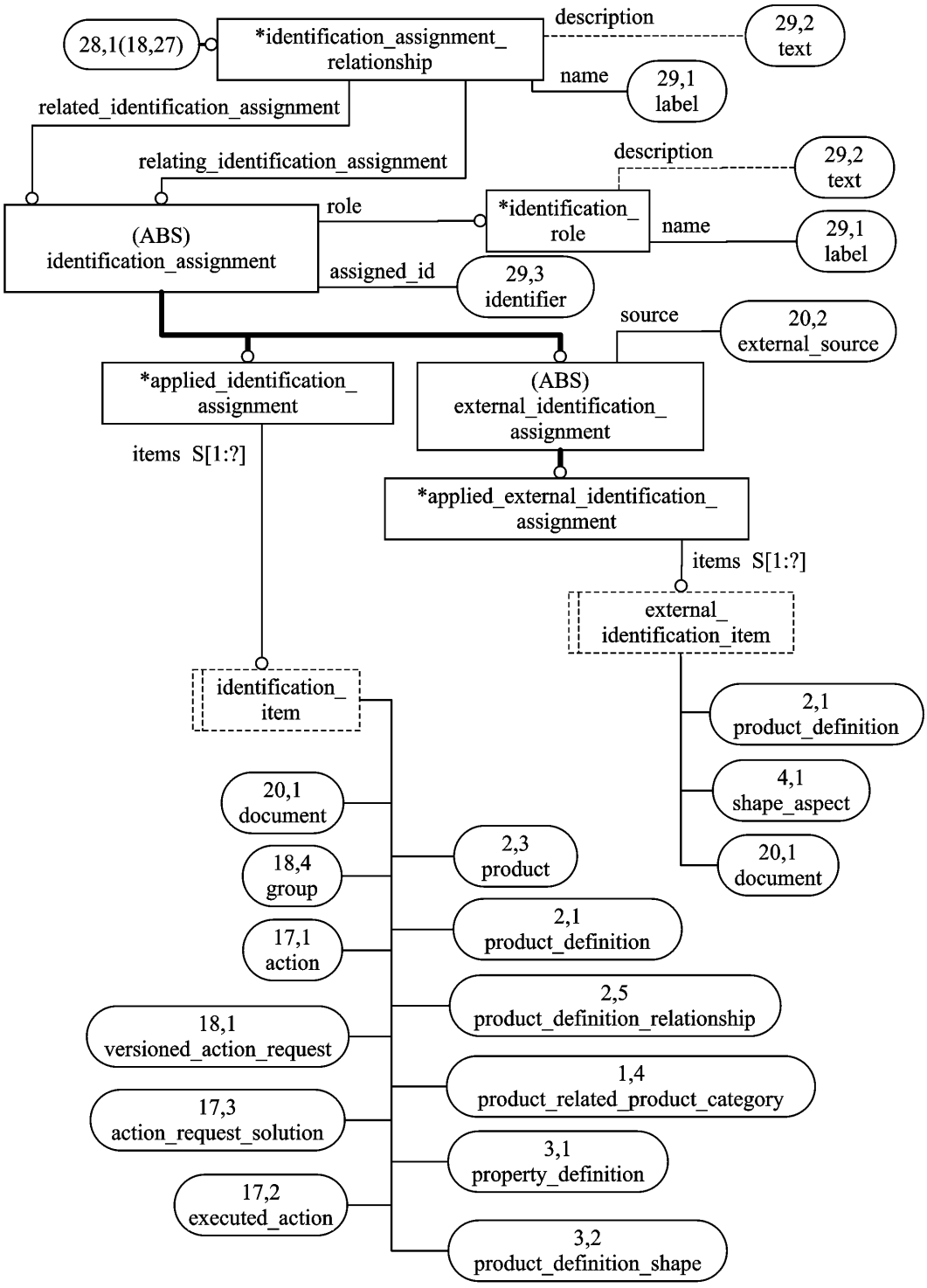

Figure H.28 — AIM EXPRESS-G diagram identification assignment

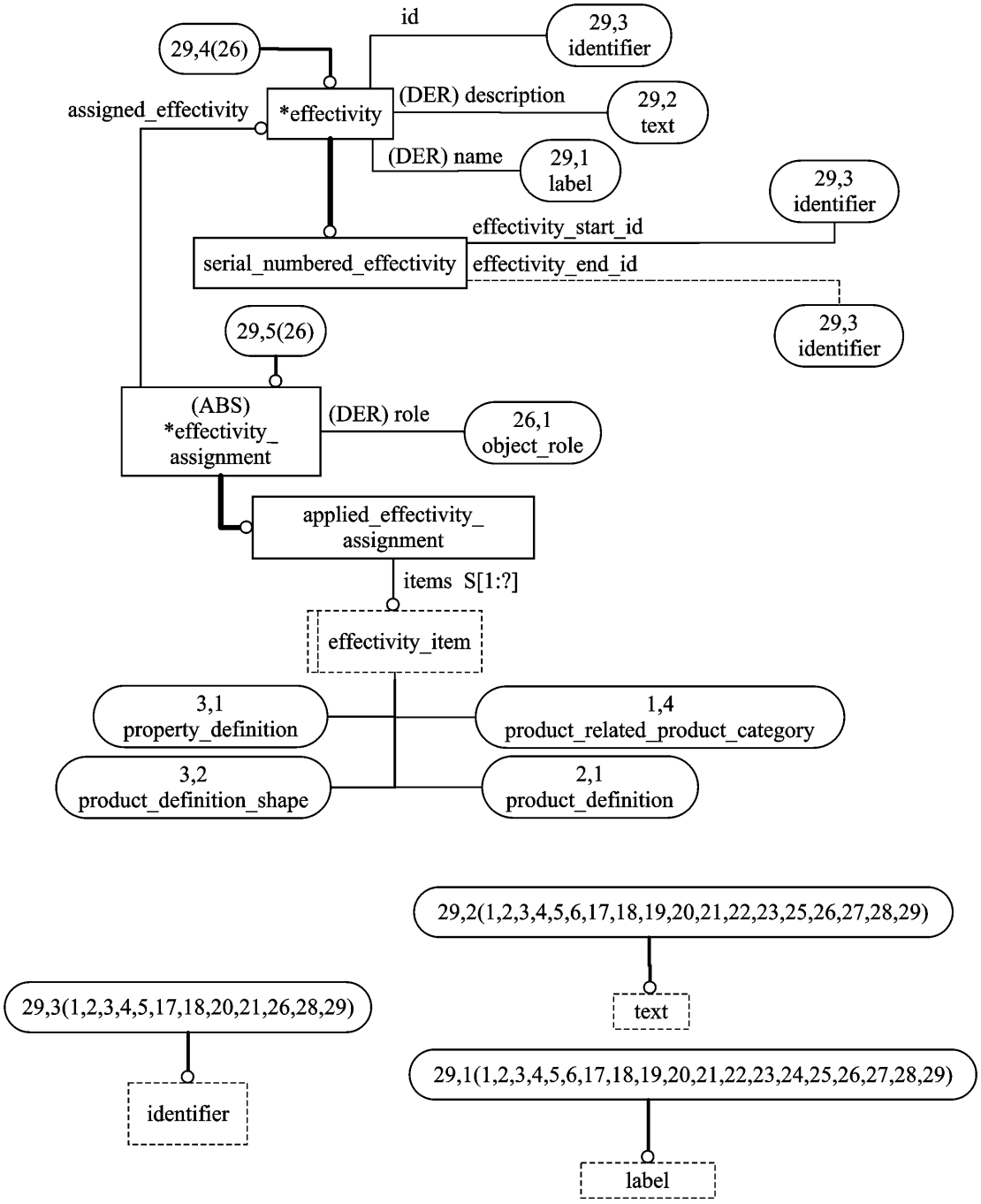

Figure H.29 — AIM EXPRESS-G diagram effectivity assignment and defined types

Annex J

(informative)

Computer-interpretable listings

This annex provides a listing of the complete EXPRESS schema specified in annex A of this part of ISO 10303 without comments or other explanatory text. It also provides a listing of the EXPRESS entity names and corresponding short names as specified in annex B of this part of ISO 10303. The content of this annex is available in computer-interpretable form and can be found at the following URLs:

— Short names: http://www.tc184-sc4.org/Short_names/

— EXPRESS: http://www.tc184-sc4.org/EXPRESS/

If there is difficulty accessing these sites contact ISO Central Secretariat or contact the ISO TC184/SC4 Secretariat directly at: sc4sec@tc184-sc4.org.

NOTE The information provided in computer-interpretable form at the above URLs is informative. The information that is contained in the body of this part of ISO 10303 is normative.

Annex K

(informative)

Technical discussions

K.1 Fundamental concepts and assumptions

K.1.1 Measures and units

Some attributes of application objects in this part of ISO 10303 specify relationships to types of measures. The appropriate unit corresponding to each measure is defined globally in the units attribute of the Ship object (See 4.2.141.3) , or locally in the local_units attribute of the Definition object (See 4.2.80.3). Units are instantiable in either SI units or other systems of units.

Table K.1 lists the units that are used in the application interpreted model of this part of ISO 10303, corresponding to each ARM measure type.

Table K.1 — ARM measures and corresponding AIM measures and units

application object attribute reference	application interpreted model measure and unit
Airflow_volume_measure	value_representation_item with value_representation_item.value_component of context_dependent_measure and derived_unit in set of Units referenced from global_unit_assigned_context.units. The derived_unit.name shall be airflow_volume_unit. The derived_unit.elements shall consist of two derived_unit_elements: 1) derived_unit_element.unit = length_unit and derived_unit_element.exponent = 3; 2) derived_unit_element.unit = time_unit and derived_unit_element.exponent = –1.
Area_measure	value_representation_item with value_representation_item.value_-component of area_measure and derived_unit in set of Units referenced from global_unit_assigned_context.units. The derived_unit.name shall be area_unit. The derived_unit.elements shall consist of one derived_unit_element: 1) derived_unit_element.unit = length_unit and derived_unit_element.exponent = 2.
Length_measure	value_representation_item with value_representation_item.value_component of length_measure and length_unit in set of Units referenced from global_unit_assigned_context.units

Table K.1 — ARM measures and corresponding AIM measures and units (continued)

Positive_length_measure	value_representation_item with value_representation_item.value_component of positive_length_measure and length_unit in set of Units referenced from global_unit_assigned_context.units
Luminous_intensity_measure	value_representation_item with value_representation_item.value_component of luminous_intensity_measure and luminous_intensity_unit in set of Units referenced from global_unit_assigned_context.units
Mass_measure	value_representation_item with value_representation_item.value_component of mass_measure and mass_unit in set of Units referenced from global_unit_assigned_context.units
Plane_angle_measure	value_representation_item with value_representation_item.value_component of plane_angle_measure and plane_angle_unit in set of Units referenced from global_unit_assigned_context.units
Ratio_measure	value_representation_item with value_representation_item.value_component of ratio_measure and ratio_unit in set of Units referenced from global_unit_assigned_context.units
Specific_heat_capacity	value_representation_item with value_representation_item.value_component of context_dependent_measure and derived_unit in set of Units referenced from global_unit_assigned_context.units. The derived_unit.name shall be Specific_heat_capacity_unit. The derived_unit.elements shall consist of three derived_unit_elements: 1) derived_unit_element.unit = length_unit and derived_unit_element.exponent = 2; 2) derived_unit_element.unit = time_unit and derived_unit_element.exponent = –2; 3) derived_unit_element.unit = thermodynamic_temperature_unit and derived_unit_element.exponent = –1.

Table K.1 — ARM measures and corresponding AIM measures and units (continued)

Thermal_conductivity_measure	value_representation_item with value_representation_item.value_component of context_dependent_measure and derived_unit in set of Units referenced from global_unit_assigned_context.units. The derived_unit.name shall be thermal_conductivity_unit. The derived_unit.elements shall consist of four derived_unit_elements: 1) derived_unit_element.unit = mass_unit and derived_unit_element.exponent = 1; 2) derived_unit_element.unit = length_unit and derived_unit_element.exponent = 1; 3) derived_unit_element.unit = time_unit and derived_unit_element.exponent = –3; 4) derived_unit_element.unit = thermodynamic_temperature_unit and derived_unit_element.exponent = –1.
Thermodynamic_temperature_measure	value_representation_item with value_representation_item.value_component of thermodynamic_temperature_measure and thermodynamic_temperature_unit in set of Units referenced from global_unit_assigned_context.units
Coefficient_of_thermal_expansion	value_representation_item with value_representation_item.value_component of context_dependent_measure and derived_unit in set of Units referenced from global_unit_assigned_context.units. The derived_unit.name shall be coefficient_of_thermal_expansion_unit. The derived_unit.elements shall consist of three derived_unit_elements: 1) derived_unit_element.unit = length_unit and derived_unit_element.exponent = 1; 1) derived_unit_element.unit = length_unit and derived_unit_element.exponent = -1; 3) derived_unit_element.unit = thermodynamic_temperature_unit and derived_unit_element.exponent = -1.

Table K.1 — ARM measures and corresponding AIM measures and units (continued)

Coefficient_of_viscosity	value_representation_item with value_representation_item.value_component of context_dependent_measure and derived_unit in set of Units referenced from global_unit_assigned_context.units. The derived_unit.name shall be coefficient_of_viscosity_unit. The derived_unit.elements shall consist of three derived_unit_elements: 1) derived_unit_element.unit = mass_unit and derived_unit_element.exponent = 1; 2) derived_unit_element.unit = length_unit and derived_unit_element.exponent = -1; 3) derived_unit_element.unit = time_unit and derived_unit_element.exponent = –1.
Volume_measure	value_representation_item with value_representation_item.value_-component of volume_measure and derived_unit in set of Units referenced from global_unit_assigned_context.units. The derived_unit.name shall be volume_unit. The derived_unit.elements shall consist of one derived_unit_element: 1) derived_unit_element.unit = length_unit and derived_unit_element.exponent = 3.
Density_measure	value_representation_item with value_representation_item.value_component of context_dependent_measure and derived_unit in set of Units referenced from global_unit_assigned_context.units. The derived_unit.name shall be density_unit. The derived_unit.elements shall consist of two derived_unit_elements: 1) derived_unit_element.unit = mass_unit and derived_unit_element.exponent = 1; 2) derived_unit_element.unit = length_unit and derived_unit_element.exponent = –3.

Table K.1 — ARM measures and corresponding AIM measures and units (continued)

Force_measure	value_representation_item with value_representation_item.value_component of context_dependent_measure and derived_unit in set of Units referenced from global_unit_assigned_context.units. The derived_unit.name shall be force_unit. The derived_unit.elements shall consist of three derived_unit_elements: 1) derived_unit_element.unit = mass_unit and derived_unit_element.exponent = 1; 2) derived_unit_element.unit = length_unit and derived_unit_element.exponent = 1; 3) derived_unit_element.unit = time_unit and derived_unit_element.exponent = -2.
Inertia_moment_measure	value_representation_item with value_representation_item.value_component of context_dependent_measure and derived_unit in set of Units referenced from global_unit_assigned_context.units. The derived_unit.name shall be inertia_moment_unit. The derived_unit.elements shall consist of one derived_unit_element: 1) derived_unit_element.unit = lenth_unit and derived_unit_element.exponent = 4.
Moment_measure	value_representation_item with value_representation_item.value_component of context_dependent_measure and derived_unit in set of Units referenced from global_unit_assigned_context.units. The derived_unit.name shall be moment_unit. The derived_unit.elements shall consist of three derived_unit_elements: 1) derived_unit_element.unit = mass_unit and derived_unit_element.exponent = 1; 2) derived_unit_element.unit = length_unit and derived_unit_element.exponent = 2; 3) derived_unit_element.unit = time_unit and derived_unit_element.exponent = -2.

Table K.1 — ARM measures and corresponding AIM measures and units (concluded)

Pressure_measure	value_representation_item with value_representation_item.value_component of context_dependent_measure and derived_unit in set of Units referenced from global_unit_assigned_context.units. The derived_unit.name shall be pressure_unit. The derived_unit.elements shall consist of three derived_unit_elements: 1) derived_unit_element.unit = mass_unit and derived_unit_element.exponent = 1; 2) derived_unit_element.unit = length_unit and derived_unit_element.exponent = -1; 3) derived_unit_element.unit = time_unit and derived_unit_element.exponent = -2.
Speed_measure	value_representation_item with value_representation_item.value_component of context_dependent_measure and derived_unit in set of Units referenced from global_unit_assigned_context.units. The derived_unit.name shall be speed_unit. The derived_unit.elements shall consist of two derived_unit_elements: 1) derived_unit_element.unit = length_unit and derived_unit_element.exponent = 1; 2) derived_unit_element.unit = time_unit and derived_unit_element.exponent = -1.

K.2 The ship product model

In view of the complexity of a ship it is necessary to subdivide the product model for the ship into distinct functional areas, allowing for partition into a fixed number of application protocols. The current version of the ship product model is shown schematically in Figure K.1.

The key elements of the ship product model are:

— arrangement;

— moulded forms;

— mechanical systems (machinery, propulsion, cargo handling);

— structures;

— distribution systems (piping, HVAC, electrical, hydraulics/pneumatics);

— outfitting and furnishings;

— communication;

— combat systems;

— navigation;

— operation.

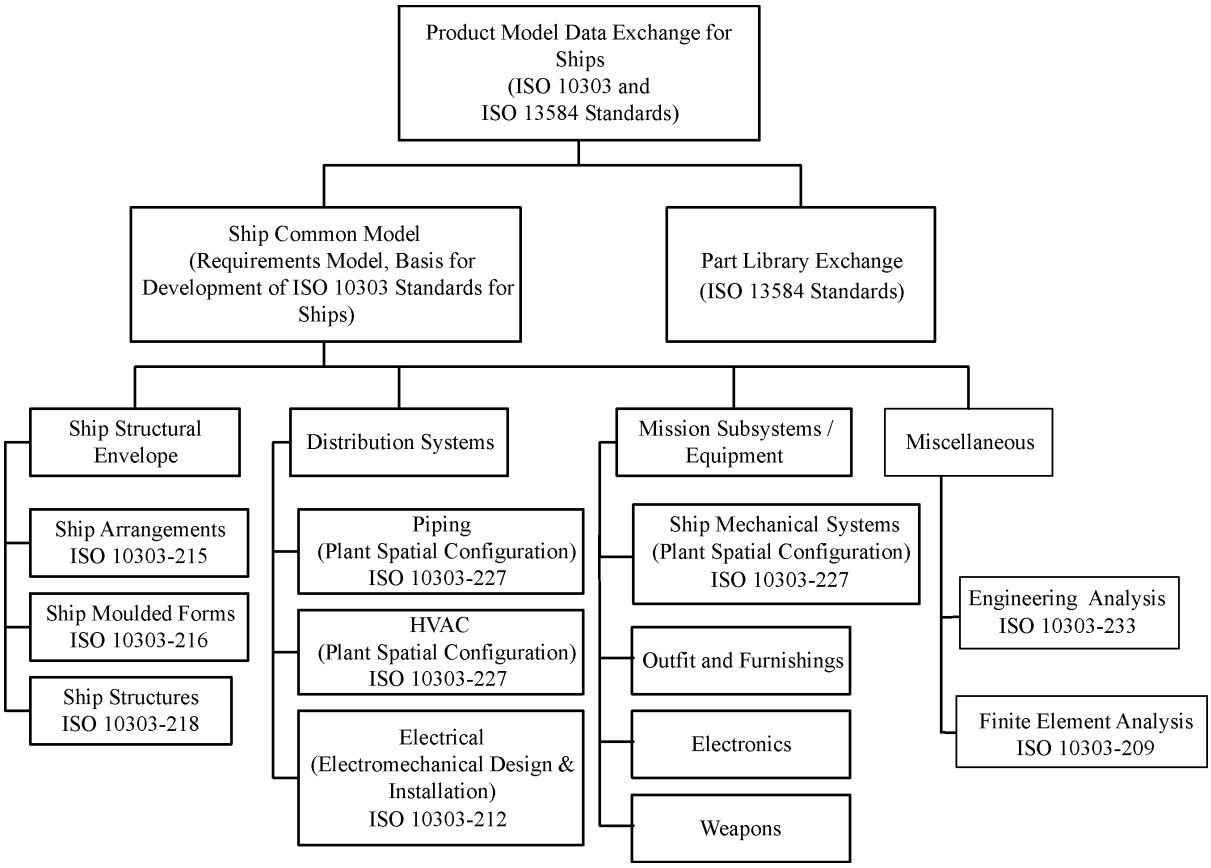

Figure K.1 — Ship product model

Each functional area of the ship product model is described by one or more different application protocols. The currently available shipbuilding application protocols under development within ISO are:

— ISO 10303-215: Ship arrangement;

— ISO 10303-216: Ship moulded forms;

— ISO 10303-218: Ship structures;

— ISO 10303-227: Piping systems and Mechanical Systems (Plant Spatial Configuration);

— ISO 10303-212: Electrical systems

The subdivision of the entire ship product model allows a distributed modelling work. It is also possible to start the modelling work with the functional areas reflecting the early design stages of the life cycle of the ship and validate these models before starting the modelling work at areas in the later lifecycle stages.

Each shipbuilding AP covers only a part of the ship product model. The consequence of this is that if the APs should work together as an entire product model for the ship then an overall mechanism, around which all the shipbuilding APs can be integrated, has to be defined. Only then the product model can be implemented in a product data management system, where all data are hold outside of application systems in the neutral data format of the ship product model. This mechanism for integrating different APs is called the Ship Common Model, which is described in the clause K.2.

K.3 The ship common model

The Ship Common Model (SCM) defines a common framework and modelling basis for all shipbuilding APs to ensure interoperability between these APs.

The Ship Common Model is a set of Building Blocks which are used in the ship product model context. The SCM provides a modelling framework, a set of domain independent and reusable product structure models that are required for more than one Application Protocol, as well as a set of commonly used constructs or utilities such as those used for configuration control and management concepts. The goal of the SCM is to contribute to the integration and overall consistency of the Application Reference Models of the different ship APs.

K.3.1 SCM framework

The modelling framework, which is part of the Ship Common Model, provides the realization of the general concepts of how to relate things, how to define their properties and how to represent them.

This framework introduces and resides in the following Building Blocks:

— definitions;

— generic_product_structures;

— representation_resources.

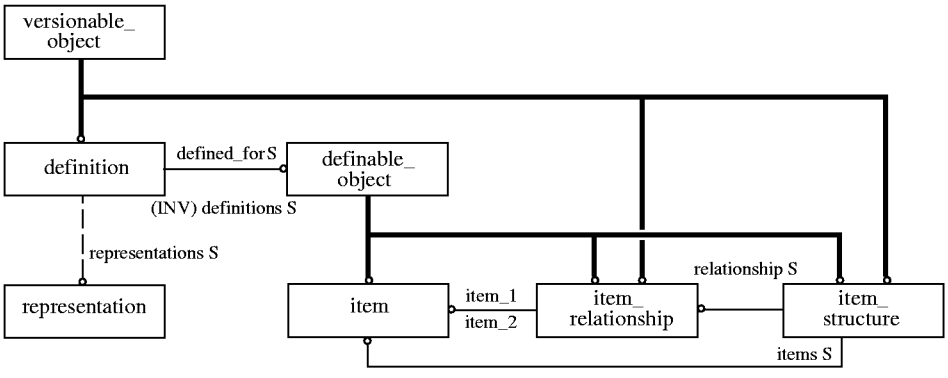

Figure K.2 — SCM framework

Effectively, this high level approach forces the product model to be split up across the main constructs of the framework namely the definable_objects, definitions and representations whilst being linked via a number of generic relationships. One of the benefits of this approach is that it enables a better management of information such as to organize the data according to different viewpoints and in the representation of life-cycle dependent requirements. Figure K.2 shows the primary constructs and relationships of the SCM framework elements.

Fundamental to the framework is the concept of a definable_object. The definable_object is a discrete, identifiable thing used in one or more activities associated with product. It serves as the most general object from which more specialized objects are derived. A definable_object can be an item, item_relationship or item_structure. Definitions describe definable_objects and are, as a result, the descriptive information-bearing entities of the model. A definition may be further classified as a Design definition, Functional definition or Manufacturing definition, etc. A definable_object may have many different versions of definition. The definition can be changed for a definable_object during the life cycle.

Typical items in the shipbuilding APs are:

— a ship and components of the ship, such as hull, superstructure, deck, and propeller;

— a part of a ship from different point of views such as assembly, system, and space;

— equipment, such as pump, generator, main engine, or pipes;

— steel structure elements, such as plate, profile, double bottom, frame, and bulkhead;

— features of steel structure elements, such holes, cutouts, and endcuts;

— functional elements, such as ports and logical_connections.

The properties of an item are carried by the definitions. A definition must be defined for an item but an item may exist without any definition. Every property of a concept and therefore, every definition of an item, may be described in many different ways. Thus a definition can have different representations, but in some circumstances there can be a definition without representation. New representations in the shipbuilding APs can be created by subtyping them from representation. The high level relationships between the main constructs of the modelling framework can be restricted in the subtypes of each AP. This is done by re-declarations of the attributes.

K.3.2 Domain models

The domain models provide a set of templates for organizing the product being modelled along a number of different axes or views, such as product by assembly, by system or by space. However, the templates also provide a set of implicit modelling techniques for the organization of the product data. Those domain models represent generic structures which allow the modeller to organize the data of the product according to their need. The benefit of this approach is that it reduces the modelling effort, allows consistency, conformity and interoperability with the other APs that already conform to this approach.

The intention of domain models is to create a new layer of generic elements underneath the main constructs of the modelling framework, with the possibility to restrict the attribute relations using re-declarations. These new elements introduced by the domain models are subtypes of the main constructs of the modelling framework.

The following domain models belong to UoF product_structures of the Ship Common Model:

— features;

— parts;

— product_structure_by_system;

— product_structure_by_assembly;

— product_structure_by_space;

— connection_topology.

K.3.3 Common utilities

Common utilities are a group of constructs that will be required by most shipbuilding APs. The utilities differ from the framework and domain models through the fact that the majority of cases, the utilities are ready for use and do not require any further specialization for use in an ARM. Many have been created specifically for shipbuilding although some may be able to be used externally.

Common utilities are currently available for:

— ships and ship types;

— ship general characteristics;

— configuration management;

— location concepts;

— geometric representations;

— materials;

— measures and units;

— external references.

K.4 Key concepts of ship arrangement

K.4.1 Arrangement (internal subdivision)

The hull form of a ship is internally subdivided early in the design lifecycle by the introduction of many additional surfaces. These surfaces are associated with the moulded form elements such as bulkheads and decks. Structural entities such as plate parts and stiffeners will be defined on these surfaces as the design progresses. A region of the ship, whether it be interior to the hull such as a tank or enclosing one if its exposed decks such as a helicopter landing platform, is designated a space. Two types of spaces are addressed by this AP: compartments and zones. Compartments represent physical, bounded spaces. Zones represent regions surrounded by some abstract boundary or alternatively can be defined as a set of compartments.

The most common type of spatial partioning is the sudivision of a ship into compartments. A compartment is very similar to the idea of a room in a building. The compartment is bounded by the

surfaces representing structural decks and bulkheads and also by non-structural (or non load bearing) surfaces that form joiner bulkheads. Compartments may be classified according to the function they perform with regard to the operation of the ship. The types of spaces supported by this AP are cargo/stowage (both liquid and dry cargo), void, habitable, and machinery/equipment. Collections of attributes have been defined for the various compartments depending on its designated use. Compartments serve a vital function in configuration managing engineering part occurrences throughout the lifecycle of the ship.

In some cases, the same surfaces that subdivide a ship into compartments may also be used to subdivide the ship into zones. In other cases, additional moulded form geometry elements or geometric surfaces may be required to define zone boundaries. On naval ships, multiple zone subdivisions such as pressure (collective protection system), subsafe, damage control, and arrangement zones will be defined and each subdivides the hull into an independent set of spaces. Sometimes, two zones may have the same boundary, however, each zone is still independently represented.

In addition to identifying the various spaces on the ship, it is important to represent the connectivity between these spaces. This model supports several types of relationships between spaces, specifically adjacency, functional, positional, connection, and enclosing. Functional relationships can be used to record the fact that one space's design parameters are dependent on some functional characteristic of another spaces such as a pair of port and starboard ballast tanks used for anti-roll stabilization. Positional relationships capture design intent expressing the fact that certain spaces must maintain geometric characteristics similar to another spaces, such as two spaces that should maintain the same transverse width dimension. Finally, enclosing relationships allow the product model to record the fact that one space may be completely surrounded by another space, such as a free-standing Lube Oil Settling Tank in the Machinery Space.

From a functional standpoint, the model has been developed to associate properties with the various compartments appropriate to their function. These properties include volumetric capacities, length measures, and cross-sectional areas. The ability to specify constraints on these properties is provided for where appropriate so as to assist engineers in the early stages of design. For example, it is possible to specify a minimum length for a compartment, as well as a maximum length for the compartment. Likewise, it is possible to record an estimated compartment volume, as well as a calculated and a measured.

K.4.2 Compartments

A ship is divided horizontally by decks, platforms, flats, levels, and the bottom shell. These divisions apply to the entire ship, both in the main hull and in the superstructure. Deck gratings, false decks, or similar flats are not considered as division boundaries. Between horizontal division boundaries, the ship is divided vertically by tight or nontight bulkheads. Except for spaces designated as voids, cofferdams, or tanks, only tight boundaries are considered.

Every volume enclosed by horizontal and vertical boundaries, except for minor utility areas such as peacoat lockers, linen lockers, cleaning gear lockers, and other similar areas, is considered a compartment. Some compartments, by this definition, may or may not have access closures. Compartments are assigned a compartment name and a compartment number. Compartments that extend vertically through more than one horizontal division boundary, such as machinery spaces and deep tanks, are considered to be located on the lowest horizontal boundary.

K.4.3 Design zones

One common type of internal subdivision is the design zone. Whereas a compartment is a subdivision involving the functional aspects of a completed, or in-service, ship, a design zone is associated with the manufacturing of a ship by the design agent or shipbuilder. Design zones are used to break up the ship into blocks for facilitating design and construction.

Design zones, like compartments, are bounded by surfaces representing decks, bulkheads, the hull, and so on. They may also be bounded by other surfaces. A common use of the design zone is to configuration manage aspects of the ship design process within its bounds. For this reason, the subdivision model model has been developed to allow product structuring by zone. Structural parts, structural assemblies, and engineering parts can all be associated with a particular zone of the ship.

K.4.4 Fire zones

The design of a ship is likely to provide fire containment capablities. A damage control console provides remote control fire containment at a central site. The ship design process entails subdividing the vessel into a number of fire zones. A fire zone boundary is a physical boundary designed to retard the passage of flame and smoke from one area of the ship to the next. All fire zone boundaries are watertight or fumetight bulkheads. Fire zone boundaries in the hull are constructed of steel. Bulkheads in the superstructure, if aluminum, are covered with non-combustible thermal insulation. Each fire zone boundary on the damage control deck is provided with spring loaded, joiner type, fire doors, each held open by an electromagnetic catch. An adjacent quick acting watertight door is also provided. Each fire zone has fire, smoke, and heat sensors which activate a central display on the hazard detection panel of the damage control console and enable the ship's crew to safely and effectively monitor and control onboard fires.

K.4.5 Collective protection system (CPS) zones

A type of zone common on Naval ships is a pressure zone. These pressure zones are used to define regions of the ship that have been designed to maintain a pressure slightly higher than that of the outside atmosphere. These zones, commonly referred to as collective protection system (CPS) zones, are necessary to combat biological and nuclear warfare. The air pumped into these zones is specially filtered to remove harmful contaminants. As with fire zones discussed above, the boundaries of these zones are fitted with special types of automatic closures to secure the zones in an emergency.

K.4.6 Arrangement zones

Arrangement zones are used early in the design to control and manage the arrangement of compartments on the ship. An individual or workgroup may be assigned a collection of compartments that are to be arranged within a given domain (i.e. the arrangement zone). Working within the bounds of this zone, the designers can define the compartment boundaries according to the requirements for the spaces, such as the number of crew, or the amount of cargo.

Bibliography

1. IEEE Std 1320.1-1998, *Standard for Functional Modeling Language - Syntax and Semantics for IDEF0.*

2. ISO 10303-227:—[1)], *Industrial automation systems and integration — Product data representation and exchange — Part 227: Application Protocol: Plant spatial configuration.*

3. GRAU M., *AIM development guideline for the Ship Product Model, ISO TC 184/SC4/WG3 N885*, 2000-06-24.

4. HAENISCH. J., *Template Repository for the Ship Product Model, ISO TC 184/SC4/WG3 N886*, 2000-10-13.

5. PALMER, M., *Guidelines for the Development and Approval of STEP Application protocols, ISO TC 184/SC4 N535*, 1998-12-18.

6. SC4 SECRETARIAT, *Guidelines for the development of mapping specifications, 2nd edition, ISO TC 184/SC4 N1029*, 2000-04-20.

7. SC4 SECRETARIAT, *SC4 Supplementary directives - Rules for the structure and drafting of SC4 standards for industrial data, ISO TC 184/SC4 N1217*, 2001-11-01.

8. TURNER, T., *AP Development Guidelines for Shipbuilding, ISO TC 184/SC4/WG3 N701,* 1997-11-21.

[1)] To be published. (Revision of ISO 10303-227:2001)

Index

Page

ICS 25.040.40

Price based on 1181 pages