INTERNATIONAL STANDARD

ISO 10303-216

First edition
2003-08-01

Industrial automation systems and integration — Product data representation and exchange

Part 216: Application protocol: Ship moulded forms

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

Partie 216: Protocole d'application: Formes moulées de navires

Reference number
ISO 10303-216:2003(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.

The main task of technical committees is to prepare International Standards. 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 document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 10303-216 was prepared by Technical Committee ISO/IEC/TC 184, *Industrial automation systems and integration*, Subcommittee SC 4, *Industrial data*.

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

Each part of this International Standard is a 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 200 series.

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

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

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 ship moulded forms and related hydrostatic properties.

The definition of ship moulded forms supports the geometrical representation of the ship hull, propellers, rudders, appendages, and internal structures of the ship.

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

The series of shipbuilding application protocols assumes that the ship product model can be divided into separate ship systems that each covers a key element of the ship for its whole life cycle. These key elements are: ship moulded forms, ship arrangements, 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 application protocols. The full series of shipbuilding 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 L has additional information on the shipbuilding application protocols and their elements. It also contains information on data common to the shipbuilding application protocols.

Within the series of shipbuilding application protocols this part of ISO 10303 details the geometry of a ship moulded form.

A moulded form is the shape and a set of design parameters of different parts of the ship that does not include information on the thickness of the material from which it is constructed. A moulded form may describe a ship hull, propeller, rudder, appendage, deck, or a ship structural element such as a bulkhead. A moulded form of particular interest is the ship hull, which is referred to as the hull moulded form. The hull moulded form will be exchanged between companies during the initial design and it is the basis of hydrostatic calculations.

All moulded forms are covered by this part of ISO 10303, and a collection of moulded forms that describe the ship as a whole is termed a ship moulded form.

This application protocol satisfies an industrial need of reducing the time required for hull form design, performance prediction, and ship structural design by facilitating the electronic exchange of hull moulded form geometry and hydrostatics between different companies. Also, it satisfies an industrial need for individual companies to integrate computer applications by providing an electronically accessible common view of hull and internal ship geometry for ship design and manufacture.

The fundamental assumptions for ship moulded forms are:

— ship moulded form and each moulded form have a definition;

— the definitions are approved and versioned;

— a ship moulded form is associated with a ship and is composed of moulded forms;

— a moulded form provides a geometric representation for a ship.

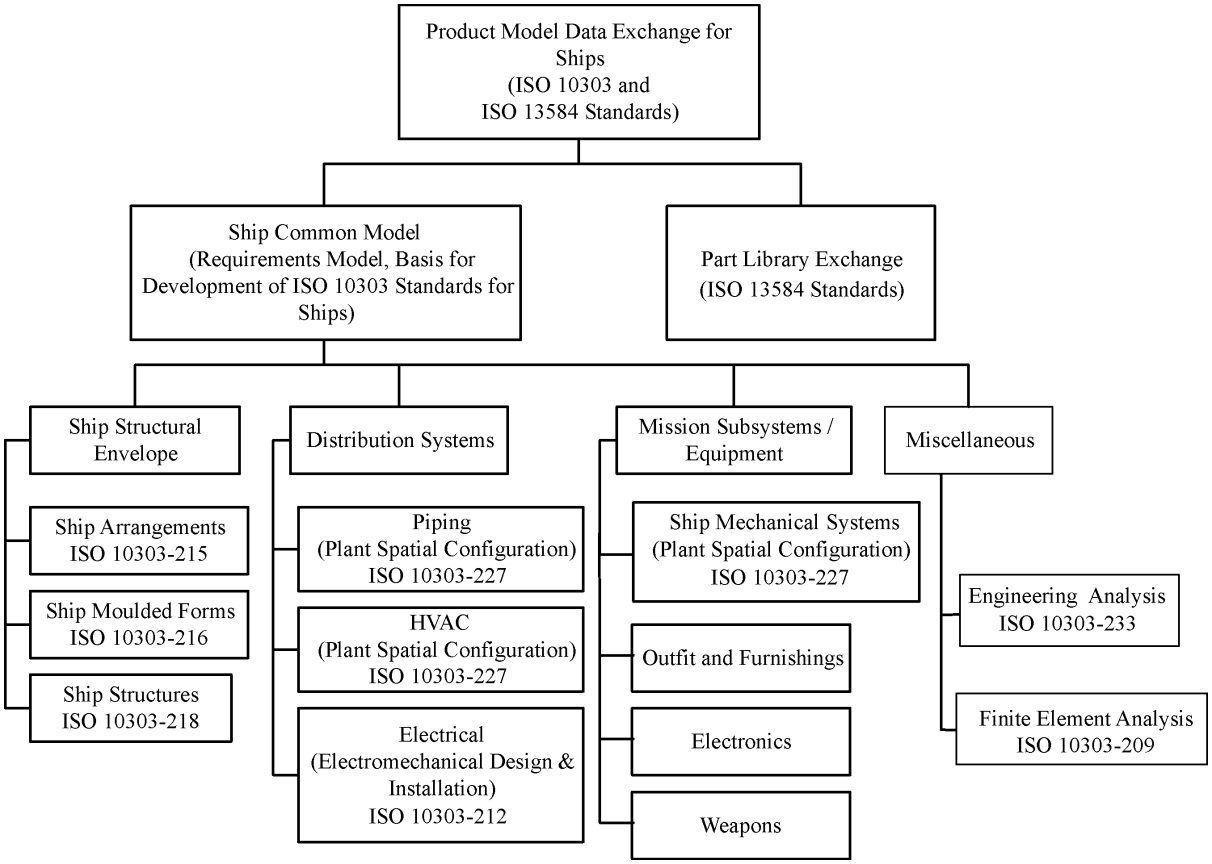


Figure 1 — Shipbuilding application protocols

The representations that are described reflect the different input and output capabilities of CAD, CAM, and other software systems in common usage of the shipbuilding process.

The geometric representations supported are:

— offset table representation;

— wireframe representation;

— surface representation.

All geometric representations assume no knowledge of the thickness of the ship moulded form. All measurements are based on one set of units defined for the ship. Geometric representations are used to

describe a hull surface moulded form or other moulded surfaces of the ship.

The hydrostatic properties are considered as a particular type of lifecycle definition for a moulded form. The hydrostatic properties are those of the intact hull that depend on the ship's draught, such as displacement, centre of buoyancy, and centre of flotation. Damage stability is not covered since no information on ship compartmentation is described by this application protocol.

NOTE ISO 10303-215 maybe used to represent damage stability data.

This application protocol defines the context, scope, and information requirements for the exchange of ship moulded form definitions, geometric representations, related hydrostatic properties, 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 summarizes the functionality and data covered by the AP. Clause 3 lists the words defined in this part of ISO 10303 and gives pointers to words defined elsewhere. An application activity model that is the basis for the definition of the scope is provided in 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. 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 2 contains the data planning model that provides a high level description of the requirements for 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).

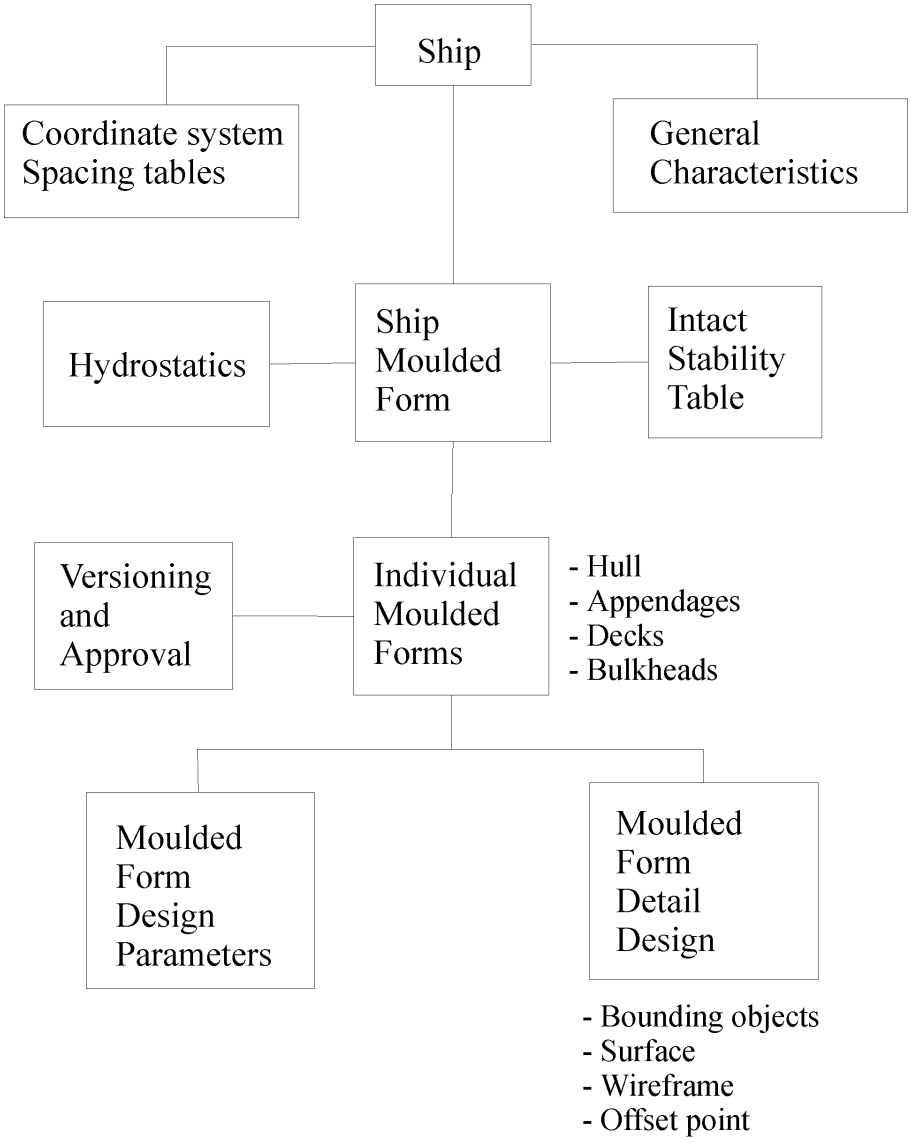


Figure 2 — Data planning model

Industrial automation systems and integration — Product data representation and exchange — Part 216: Application protocol: Ship moulded forms

1 Scope

This part of ISO 10303 specifies the scope and information requirements for the exchange of ship moulded form definitions, geometric representations, and related hydrostatic properties.

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

NOTE 2 An overview of the AP Ship Moulded Forms is given in Figure 3.

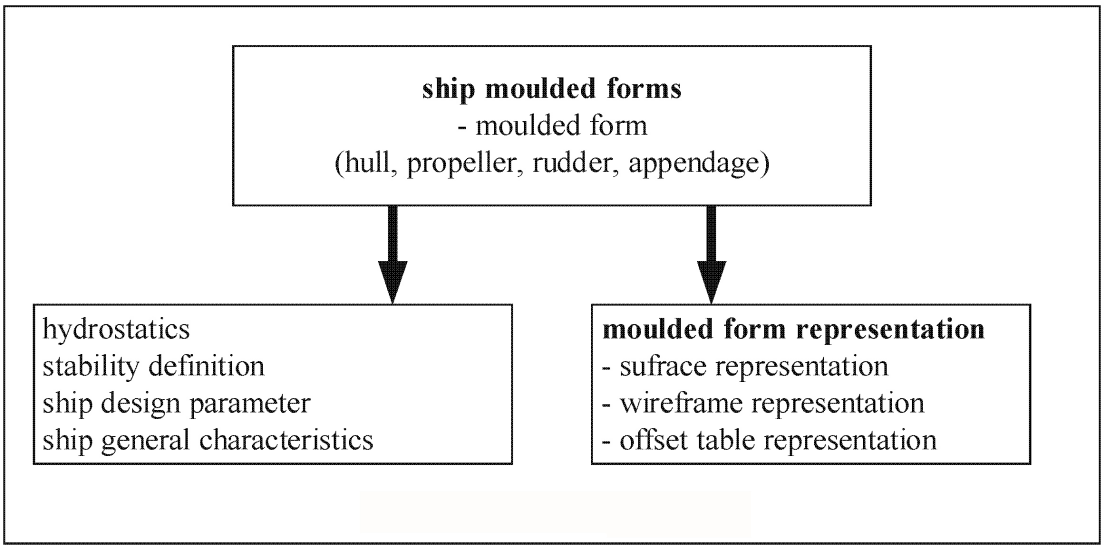


Figure 3 — Ship moulded forms overview

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

— definition of moulded form geometry related to commercial and naval ships;

— definition of moulded form geometry of the preliminary design, detailed design, and production stages of the life cycle of a ship;

— definition of moulded form geometry that describe the hull moulded form of the ship, including mono hullforms, multi-hullforms, the bulbous bow, transom stern, thruster tunnels, and additional appendages;

EXAMPLE 1 Types of moulded form geometry are bilge keel, spray rails, shaft struts, and shaft bossings that are part of the final moulded form of the ship hull.

— definition of moulded form geometry that describe the moulded form of propellers and rudders;

— definition of moulded geometry that describe the moulded form of decks including camber and sheer;

— definition of moulded geometry of internal ship compartment boundaries and the moulded form geometry of ship structural and non-structural elements;

EXAMPLE 2 Bulkheads, girders, and profiles are examples of moulded form geometry of ship structural elements.

— definition of general characteristics;

EXAMPLE 3 Main dimensions, ship type, shipyard, ship owner, and classification data are examples of general characteristics.

— definition of design parameters for the ship hull, bulbous bow, propeller, rudder, and appendages that are necessary to describe the moulded form, and are required to calculate hydrostatic properties;

— definition of hydrostatic properties of the ship moulded form that depend on the draught of the ship;

EXAMPLE 4 Displacement, centre of buoyancy, centre of flotation, metacentric height, and cross curves of stability are example of hydrostatic properties.

— definition of global and local co-ordinate systems and spacing tables used in naval architecture for position purposes;

— shape definition of ship moulded forms that use one of the following specified types of geometric representation:

  — offset table representation;

  — wireframe representation;

  — surface representation.

— geometric representations containing geometric elements used in naval architecture;

EXAMPLE 5 Waterlines and buttock lines are examples of geometric representations.

— version control and approval of moulded forms and related hydrostatics.

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

— product definition data related to hull plating defined on the moulded form;

— product definition data related to ship compartmentation and ship arrangements;

NOTE 3 ISO 10303-215 may be used to represent ship compartmentation and arrangement data.

— product definition data related to ship structures and ship assemblies;

NOTE 4 ISO 10303-218 may be used to represent ship structures such as panel systems.

— product definition data related to ship machinery and ship superstructures;

— mechanical systems and material aspects of propellers, rudders and control surfaces;

— product definition data from the decommissioning stage of the ship life cycle;

— hydromechanic properties of the ship;

— damage stability properties of ships;

NOTE 5 ISO 10303-215 may be used to represent damage stability data.

— ship longitudinal strength.

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 31 (all parts), *Quantities and units*

ISO 1000:1992, *SI units and recommendations for the use of their multiples and of certain other units*

ISO 10303-1:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles*

ISO/IEC 8824-1:1998, *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO 10303-11:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual*

ISO 10303-21:2002, *Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation methods: Clear text encoding of the exchange structure*

ISO 10303-22 :1998, *Industrial automation systems and integration — Product data representation and exchange — Part 22: Implementation methods: Standard data access interface*

ISO/TS 10303-28 —[1)] , *Industrial automation systems and integration — Product data representation and exchange — Part 28: Implementation methods: XML representation of EXPRESS schemas and data*

ISO 10303-31:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 31: Conformance testing methodology and framework: General concepts*

ISO 10303-41:2000, *Industrial automation systems and integration — Product data representation and exchange — Part 41: Integrated generic resource: Fundamentals of product description and support*

ISO 10303-42:2000, *Industrial automation systems and integration — Product data representation and exchange — Part 42: Integrated generic resource: Geometric and topological representation*

ISO 10303-43:2000, *Industrial automation systems and integration — Product data representation and exchange — Part 43: Integrated generic resource: Representation structures*

ISO 10303-45:1998, *Industrial automation systems and integration — Product data representation and exchange — Part 45: Integrated generic resource: Materials*

ISO 10303-501:2000, *Industrial automation systems and integration — Product data representation and exchange — Part 501: Application interpreted construct: Edge-based wireframe*

ISO 10303-508:2001, *Industrial automation systems and integration — Product data representation and exchange — Part 508: Application interpreted construct: Non-manifold surface*

ISO 10303-511:2001, *Industrial automation systems and integration — Product data representation and exchange — Part 511: Application interpreted construct: Topologically bounded surface*

RFC 2396:1998, *Uniform Resources Identifiers (URI): Generic Syntax*

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);

[1)] To be published.

— application context;

— application interpreted model (AIM);

— application object;

— application protocol (AP);

— application reference model (ARM);

— 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 10303-31

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

— conformance testing;

— postprocessor;

— preprocessor.

3.3 Terms defined in ISO 10303-42

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

— boundary;

— connected;

— curve.

3.4 Other terms and definitions

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

NOTE Several of the following definitions have been adapted from Principles of Naval Architecture, Second Edition [10].

3.4.1
after perpendicular; aft perpendicular
a vertical plane at the intersection of the summer load (design) waterline with the after side of the rudder post (stern post), the centreline of the rudder stock if there is no rudder post, or the transom

NOTE For a ship with a transom stern it is the intersection of the design waterline with the transom.

3.4.2
amidships; midship
a vertical plane arranged transversely in the (y,z) direction of the global coordinate system, located either at half of the distance between the after perpendicular and the forward perpendicular, or midway between the ends of the design waterline

3.4.3
appendage
an exterior attachment, addition, or adjunct to the ship's hull

3.4.4
baseline
the main horizontal reference plane at or near the bottom of the ship from which heights are measured in design and production

NOTE Baseline is measured zero vertically.

3.4.5
bilge
the rounded portion of a ship's hull plating that connects the bottom to the sides

3.4.6
bilge keel
a fin fitted on the bilge of the ship to reduce rolling

3.4.7
breadth
the largest transverse measurement from one side of the ship's hull to the other side, ignoring the thickness of hull plating

NOTE The breadth is the widest transverse moulded dimension of the hull surface, not including possibly wider structure in the the ship's superstructure.

3.4.8
bulkhead
vertical wall or partition within a ship

NOTE Bulkheads are often designated by their location, use, kind of material, or method of fabrication.

3.4.9
buttock line
any longitudinal line that is parallel to the centre line of the ship

NOTE An element of the hullform representing the intersection of a vertical longitudinal plane with the hull surface of a ship.

3.4.10
camber
a planar curve that defines the curvature of a deck in the transverse direction. Decks are built with camber to allow the runoff of rainwater or seawater

NOTE A positive value of a camber would mean that the deck at side is lower than the deck at centerline.

3.4.11
centreline
an imaginary line that results from the intersection of the centreplane with the moulded form

NOTE This is the main reference datum line used during design and production from which all transverse measurements are taken.

3.4.12
centreplane
a vertical plane arranged longitudinally in the (x,z) direction of the global coordinate system that passes through the point at half of moulded breadth amidships

NOTE This is the main reference datum plane used during design and production from which all transverse measurements are taken and divides the ship from bow to stern into port and starboard sides.

3.4.13
centroid
the centre of an item, area, or volume measured with respect to some defined location

3.4.14
classification
the process of ensuring that a ship is designed, built, and maintained to a prescribed standard

NOTE This is accomplished by a classification society or the owner of the ship through approval of the design, inspection of the the construction, and periodic surveys of the ship during operations.

3.4.15
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 life cycle

3.4.16
configuration management
version control, design organization and classification society approval status, and product model change management for the ship's product model data

3.4.17
coordinate system; co_ordinate_system; co ordinate system:
a reference system that associates a unique set of parameters with each point in an n-dimensional space

3.4.18
deck
a vertical division of the internal space of a ship, equivalent to the function of a floor in a building

NOTE A deck may extend completely over or only partially across the ship. Likewise it may extend over the entire length of the ship or only over selected portions of the length. Deck designations are often derived by their location or their functional purpose.

3.4.19
deck house
a comparatively light structure, above the hull, that does not extend from side to side of the ship

NOTE It commonly is composed of spaces that are used for crew accommodations and control of the ship.

3.4.20
depth
the vertical distance between the ship's baseline and a horizontal level in the ship

3.4.21
fair
a smooth curve without irregularities

3.4.22
fin
a small projecting surface or attachment to the ship's hull

3.4.23
form parameters, hullform parameters
empirical mathematical equations, coefficients, cross-sectional areas, and volumes of the ship

NOTE They are used to analyze the expected performance, stability, resistance and powering requirements of a ship design.

3.4.24
forward perpendicular
the vertical plane at the intersection of the summer load (design) waterline with the fore side of the stem profile

NOTE This is often used as a reference point for measurements or locating items longitudinally on the ship.

3.4.25
frame
one of the transverse members that make up the rib-like skeleton part of the ship

NOTE Frames act as stiffeners, holding the outside plating in shape and maintaining the transverse form of the ship.

3.4.26
furnishings
items that are placed in the habitable spaces of the ship to provide sleeping, living, and work support for the passengers and crew

EXAMPLE Bunks, lockers, furniture, galley equipment are considered furnishings.

3.4.27
general arrangements
the subdivision of the ship into compartments and zones for ship operation, machinery spaces, crew and passenger habitation and support, and passageways

3.4.28
half-breadth
a transverse dimension taken from the centreplane

NOTE Because of the port and starboard symmetry of a ship's hull, many naval architectural calculations and drawings reflect only one-half of the hull and are assumed to be mirrored across the centreplane.

3.4.29
hullform
the collection of geometry that defines the shape of the hull of the ship

NOTE In some naval architecture references the hullform is considered to include the uppermost watertight deck of the ship. For the purposes of this part of ISO 10303, the hull surface and uppermost watertight deck are defined as separate moulded form objects.

3.4.30
hydrodynamic
forces resulting from the flow of liquid around a ship

3.4.31
hydrostatics
forces acting on a ship in still water as a result of air and water pressure on the outside under-water part of the ship

3.4.32
hydrostatic property
empirical mathematical equations, locations of centres of gravity and bouyancy, cross-sectional areas, and volumes of the ship that are used to analyze the expected stability of a ship under various loading conditions and at various drafts

NOTE For a list of property types see 4.2.47.1.

3.4.33
knuckle
the meeting of any two surfaces of a ship at an angle that is not a continuous curve or a plane

NOTE A hullform element representing a mathematical 1st order (tangency) discontinuity between adjoining portions of the hullform, such as a chine.

3.4.34
length between perpendiculars
the horizontal distance measured longitudinally between the forward and aft perpendiculars

3.4.35
longitudinal
a measurement along the length of a ship

NOTE In the ship global coordinate system defined in this part of 10303, longitudinal dimensions are along the X axis.

3.4.36
main dimensions
principal geometric measurements of the hullform that are used in analysis and design

NOTE For a list of principal measurements see 4.2.78.

3.4.37
mid-body
for those ships with a parallel portion, the part of the hullform between the aft-body and the fore-body

EXAMPLE Tankers have a parallel mid-body portion, but Naval ships seldom have a parallel mid-body.

3.4.38
moulded form
the shape and significant design parameters for different parts of the ship that does not include information on the thickness of the material from which the parts are constructed

NOTE A moulded form may describe a ship hull, propeller, rudder, appendage, deck, superstructure, or the surface underlying a ship structural element such as a bulkhead.

3.4.39
offset
the coordinate value of a point (height and half-breadth) on the intersection line of an orthogonal plane with a ship moulded form

NOTE The offset is used to define the distance to the centre plane, or the distance to amidships if the plane lies in the longitudinal direction.

3.4.40
outfit, outfitting
items that are permanently attached to the ship but that are not part of the primary structural systems. For the purposes of this part of ISO 10303, piping, electrical, and HVAC systems are considered distributed systems rather than outfitting.

EXAMPLE Hatchways, railings, doors, ladders.

3.4.41
pitching
a rotation of the ship about the transverse axis of the ship

3.4.42
rolling
a rotation of the ship about the longitudinal axis of the ship

3.4.43
sheer
a planar curve that defines the curvature of a deck in the longitudinal direction

NOTE In the case of a deck with camber, the deck centerline may also have sheer.

3.4.44
ship structure
the structural systems, plates, and stiffeners that comprise the hull, decks and bulkheads of the ship

NOTE The moulded forms defined in this part of ISO 10303 may define the surfaces upon which structural plates and stiffeners are defined. The geometric definition of the ship structure is partially defined through references from ISO 10303-218 to the underlying geometric surfaces of the moulded forms defined in this part of 10303.

3.4.45
station
the intersection of a transverse YZ plane with the hull moulded form

NOTE Typically the shape of the ship is defined by 20 stations equally spaced along the length of the vessel. These stations define the cross-sectional shape and are used along with the waterlines and buttocks to represent the 3-dimensional wireframe shape of the ship.

3.4.46
stem
a planar curve that extends from the base of the ship to the uppermost deck at the centreline of the ship to form the bow of the ship

3.4.47
summer load waterline; design waterline
the maximum draught at which a ship is permitted to sail during the summer months

3.4.48
superstructure
a decked-over structure above the upper deck

NOTE The outboard sides of a superstructure are formed by the shell plating, as distinguished from a deckhouse that does not extend outboard to the shell plating.

3.4.49
thruster
an auxiliary propulsion device located in the hull

EXAMPLE Thrusters are often used to move the ship sideways.

3.4.50
transom
the flat plate stern of a ship that is the result of truncating the waterlines and the buttock lines

3.4.51
transverse
a measurement across the ship

NOTE In the ship global coordinate system defined in this part of 10303, longitudinal dimensions are along the Y axis.

3.4.52
uniform resource identifier
a compact string of characters for identifying an abstract or physical resource. It defines a simple and extensible means for identifying a resource.

3.4.53
vertical
a measurement of height in the ship

NOTE In the ship global coordinate system defined in this part of 10303, vertical dimensions are along the Z axis.

3.4.54
waterline
the intersection line of the water's surface with the ship's hull when the ship is afloat

NOTE In general, a waterline may be the intersection of any horizontal plane with the hull moulded form.

3.4.55
weather deck; main deck
the uppermost watertight deck of the ship

3.4.56
yawing
a rotation of the ship about the vertical axis of the ship

3.5 Abbreviations

For the purposes of this document, the following abbreviations apply:

AAM	application activity model
AIM	application interpreted model
AP	application protocol
ARM	application reference model
B	centre of buoyancy
BB	building block
CoG	centre of gravity
CAD	computer aided design
CAM	computer aided manufacture
CFD	computational fluid dynamics
GUID	globally unambiguous identifier
IACS	International Association of Classification Societies
IMO	International Maritime Organisation
PICS	protocol implementation conformance statement
SCM	Ship Common Model
SI	Système International
SOLAS	safety of life at sea
UoF	unit of functionality

4 Information requirements

This clause specifies the information required for the exchange of ship moulded form definitions, geometric representations and related hydrostatic properties.

The information requirements are specified as a set of units of functionality, application objects, and application assertions. These assertions pertain to individual application objects and to relationships between application objects. The information requirements are defined using the terminology of the subject area of this application protocol.

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 list 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 moulded forms application protocol. This part of ISO 10303 specifies the following units of functionality:

— basic_geometry;

— configuration_management;

— definitions;

— external_references;

— hull_class_applicability;

— hydrostatics;

— items;

— location_concepts;

— offset_table_representations;

— ship_design_parameter;

— ship_general_characteristics;

— ship_measures;

— ship_moulded_forms;

— surface_representations;

— wireframe_representations.

4.1.1 basic_geometry

This UoF provides the naval architecture nomenclature and categorization given to points, curves and surfaces used in the representation of ship moulded forms.

The following application objects are used by the basic_geometry UoF:

— Ship_curve;

— Ship_curve_with_spacing_position;

— Ship_point;

— Ship_surface;

4.1.2 configuration_management

The configuration-management UoF specifies the information required to track the approval, versioning, and changes to ship moulded forms definitions.

NOTE The approval information describes when, who, and what has been approved and to what level of approval, as well as how approvals are related to each other. Versions identify and describe what definition is subject to version control and how different versions are related to each other to provide a version history. Changes describe the state of the change, when and who changed what definition and describe the impact of the change in terms of whether or not other definitions are created, modified or deleted.

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;

4.1.3 definitions

The definitions UoF describes the abstract concept for the definition of items, item structures and item relationships.

The following application objects are used by the definitions UoF:

— Definition;

— Design_definition;

— Functional_definition;

— General_characteristics_definiton.

4.1.4 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 in external libraries or 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.

4.1.5 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 and/or geometry.

The following application objects are used by the hull_class_applicability UoF:

— Hull_applicability.

4.1.6 hydrostatics

The hydrostatics UoF provides the results of calculations of hydrostatic properties based on the extreme form associated to the displacement in still water. The extreme form describes the geometry, which is defined by the moulded form plus the thickness of the material.

The following application objects are used by the hydrostatics UoF:

— Addition_of_moulded_form;

— Centre_location;

— Displacement_operation;

— Floating_position;

— Hydrostatic_definition;

— Hydrostatic_position_value;

— Hydrostatic_properties_for_constant_floating_position;

— Hydrostatic_property;

— Hydrostatic_property_value;

— Hydrostatic_scalar_value;

— Hydrostatic_table;

— Stability_definition;

— Stability_properties_for_one_floating_position;

— Stability_property;

— Stability_table;

— Subtraction_of_moulded_form.

4.1.7 items

The items UoF deals with the generic product structures of a ship, reflected by the concept of so-called “items” as well as by the concept of item structures and item relationships in which instances of items are collected and related to each other in a well defined way.

The following application objects are used by the items UoF:

— Definable_object;

— Global_id;

— Item;

— Item_relationship;

— Item_structure;

— Ship;

— Versionable_object.

4.1.8 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 descriptions 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 may be 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.

4.1.9 offset_table_representations

The offset_table_representations UoF provides the information required to geometrically represent a shape as a table of offset points. An offset table representation represents a shape entirely by a set of Ship_point objects. The Ship_point objects are listed in sections, normally these sections are 2D sections. In special cases there may be a need for using 3D sections.

The following application objects are used by the offset_table_representations UoF.

— Offset_point_table_model;

— Offset_table_shape_representation;

— Section_of_offset_point_table.

4.1.10 ship_design_parameter

The ship_design_parameter UoF provides the measurable factors that define the design parameters for each moulded form that make up the complete ship moulded form.

NOTE This information contains dimensions and ratios important to the simple characterization of the hull, deck, bow, bulb, propeller, rudder, appendage, thruster, and keel details. The UoF also contains the information concerning the overall dimensions of the ship moulded form.

The following application objects are used by the ship_design_parameter UoF:

— Appendage_moulded_form_design_parameter;

— Bottom_moulded_form_design_parameter;

— Bulb_moulded_form_design_parameter;

— Deck_moulded_form_design_parameter;

— Hull_moulded_form_design_parameter;

— Midship_tumble;

— Moulded_form_characteristics_definition;

— Propeller_location;

— Propeller_moulded_form_design_parameter;

— Rudder_moulded_form_design_parameter;

— Ship_overall_dimensions;

— Thruster_moulded_form_design_parameter.

4.1.11 ship_general_characteristics

The ship_general_characteristics UoF specifies the basic information that details the ship's dimension and identification. This information is independent of any geometric context. This information includes scalar values for principal dimensions of a ship, designation information for ship related companies, ship's class notation and all relevant rules and regulations.

The following application objects are used by the ship_general_characteristics UoF:

— Carrier;

— Class_and_statutory_designation;

— Class_notation;

— Class_parameters;

— Navy_ship;

— Owner_designation;

— Principal_characteristics;

— Regulation;

— Research_ship;

— Ship_designation;

— Shiptype;

— Shipyard_designation;

— Working_ship.

4.1.12 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.7 for additional discussion on the use of measures and units in this part of ISO 10303

4.1.13 ship_moulded_forms

The ship_moulded_forms UoF provides the information required for the description of a moulded form, the composition of moulded forms to a ship moulded form, the representation of a moulded form and the handling of symmetry properties.

The following application objects are used by the ship_moulded_forms UoF:

— Moulded_form;

— Moulded_form_boundary_relationship;

— Moulded_form_design_definition;

— Moulded_form_functional_definition;

— Moulded_form_relationship;

— Moulded_form_representation_item;

— Moulded_form_representation_relationship;

— Moulded_form_shape_representation;

— Planar_symmetry;

— Rotational_symmetry;

— Ship_moulded_form;

— Ship_moulded_form_revision;

— Symmetry.

4.1.14 surface_representations

The surface_representations UoF specifies the information required to represent a moulded form 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. Additionally these may be associated with naval architecture terms. If the curves and surfaces are inter-connected then such information is stated explicitly by the use of topology.

The following application objects are used by the surface_representations UoF:

— Non_manifold_surface_shape

— Surface_shape_representation.

4.1.15 wireframe_representations

The wireframe_representations UoF specifies the information required to represent a moulded form as a wireframe. The UoF combines topological and geometric information together to describe a wireframe as a set of intersecting 2D or 3D curves. Both simple geometry such as straight lines and complex

geometry such as B-spline curves are described by this UoF. Additionally these may be associated with naval architecture terms.

The following application objects are used by the wireframe_representations UoF:

— Edge_based_wireframe_shape;

— Knot;

— Ship_curve_segment;

— Wireframe_shape_representation.

4.2 Application objects

This subclause specifies the application objects for the ship moulded forms 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 arenot 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 Addition_of_moulded_form

An Addition_of_moulded_form is a type of Displacement_operation (see 4.2.26) in which the displacement of the Ship_moulded_form (see 4.2.93) will be increased by the displacement of the new Moulded_form (see 4.2.61).

4.2.2 Alternative_version_relationship

An Alternative_version_relationship is a relationship between two Versionable_object (see 4.2.120) objects 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.

4.2.2.1 alternative_1

The alternative_1 specifies an alternative design for the Versionable_object (see 4.2.120) defined by alternative_2. See 4.3.1 for the application assertion.

4.2.2.2 alternative_2

The alternative_2 specifies an alternative design for the Versionable_object (see 4.2.120) defined by alternative_1. See 4.3.1 for the application assertion.

4.2.2.3 reason

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

4.2.3 Appendage_moulded_form_design_parameter

An Appendage_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.63) which defines the dimensions and ratios for each type of ship hull appendage.

The data associated with an Appendage_moulded_form_design_parameter are the following:

— appendage_breadth;

— appendage_depth;

— appendage_length;

— type_of_appendage;

— user_def_appendage_type.

4.2.3.1 appendage_breadth

The appendage_breadth specifies the breadth of the appendage.

NOTE The breadth is the second largest dimension of the appendage.

4.2.3.2 appendage_depth

The appendage_depth specifies the depth of the appendage.

NOTE The depth is the smallest dimension of the appendage.

4.2.3.3 appendage_length

The appendage_length specifies the length of the appendage.

NOTE The length is the maximum dimension of the appendage.

4.2.3.4 type_of_appendage

The type_of_appendage specifies the type of the appendage defined by Appendage_moulded_form_-design_parameter.

The type_of_appendage shall be one of the following:

— active_fin;

— air_emitter;

— arrays;

— bilge_keel;

— bow_thruster;

— cathodic_protection_anode;

— duck_tail;

— passive_fin;

— rudder;

— sea_chest;

— shaft_bossings;

— shaft_strut;

— skeg;

— spray_rail;

— stern_flap;

— stern_thruster;

— user_defined.

4.2.3.4.1 active_fin

Automated stabilizers installed on a ship hull to reduce roll and yaw.

4.2.3.4.2 air_emitter

A defensive system which forms an air bubble screen around the hull of the ship, reducing transmission of machinery noise to the surrounding waters to mask potential targets or to provide alternate targets.

4.2.3.4.3 arrays

A structure which houses the equipment used for towed or fixed sonar systems or magnetometers.

4.2.3.4.4 bilge_keel

An external fin at the round of the bilge to reduce rolling.

EXAMPLE Figure 4 illustrates a bilge keel.

Figure 4 — Bilge keel

4.2.3.4.5 bow_thruster

A small propeller mounted near the bow with its axis transverse to the length of the ship; used for maneuvering.

4.2.3.4.6 cathodic_protection_anode

The blocks of sacrificial material installed external to the hull to reduce corrosion.

4.2.3.4.7 duck_tail

A structure at the stern of a ship which increases the amount of water around the propellers to reduce vibration, improve stability, and increase speed.

4.2.3.4.8 passive_fin

A fixed fin attached to the ship hull that changes the flow in this area.

4.2.3.4.9 rudder

A structure that is attached vertically to the stern of a ship and operates to control the ship's direction.

4.2.3.4.10 sea_chest

A casting or steel plate structure installed in the hull to allow connection of internal seawater piping systems, ballasting systems, or other through-hull access.

4.2.3.4.11 shaft_bossings

The curved portion of the ship hull that surrounds and supports the propeller shaft.

EXAMPLE Figure 5 illustrates a shaft bossing.

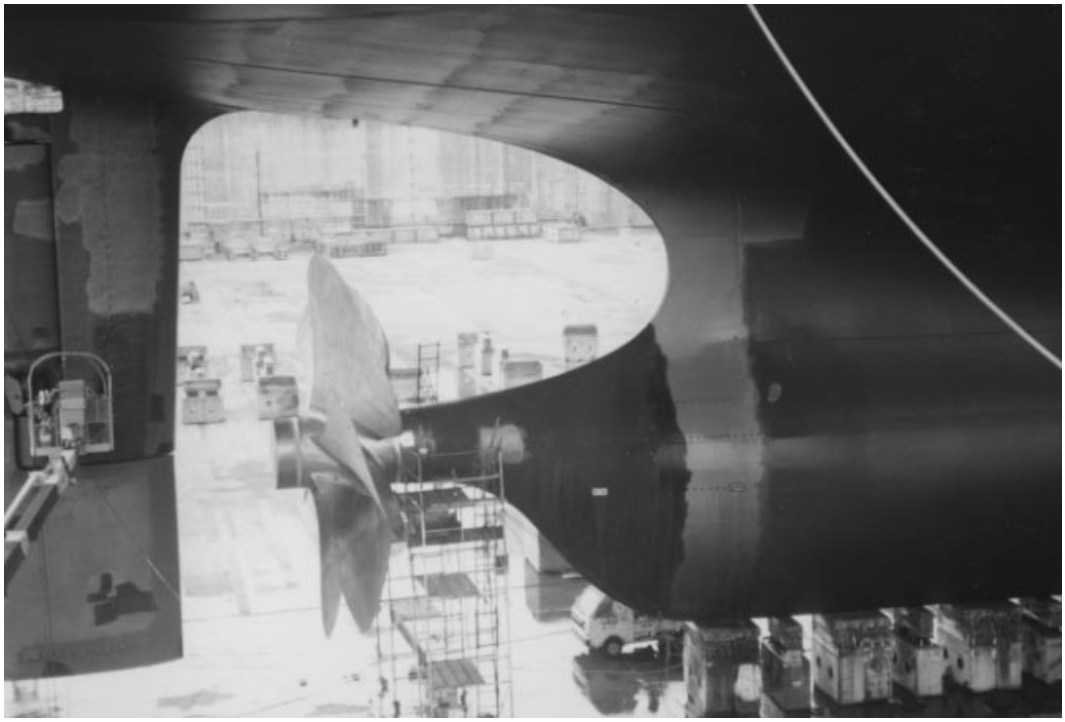


Figure 5 — Shaft bossings

4.2.3.4.12 stern_flap

A structure installed on the stern of a ship to create a vertical lift at the transom and to modify the distribution of pressure on the after portion of the hull to reduce drag and increase speed.

4.2.3.4.13 shaft_strut

A bearing stool for the part of the propeller shaft outside the hull (normally for twin or multi-screw-arrangements).

EXAMPLE Figure 6 and Figure 7 illustrates types of shaft struts.

4.2.3.4.14 skeg

A fin-like projection on the bottom of the ship hull that supports the lower edge of the rudder.

4.2.3.4.15 spray_rail

A ledge on the ship hull that keeps away spray water.

4.2.3.4.16 stern_thruster

A small propeller mounted near the stern with its axis transverse to the length of the ship; used for maneuvering.

4.2.3.4.17 user_defined

The appendage type is specified in the user_def_appendage_type attribute.

4.2.3.5 user_def_appendage_type

The user_def_appendage_type specifies a user-defined role or purpose for the appendage. The user_def_appendage_type need not be specified for Appendage_moulded_form_design_parameter.

NOTE User_def_appendage_type is not optional when the result of an type_of_appendage is USER_DEFINED.

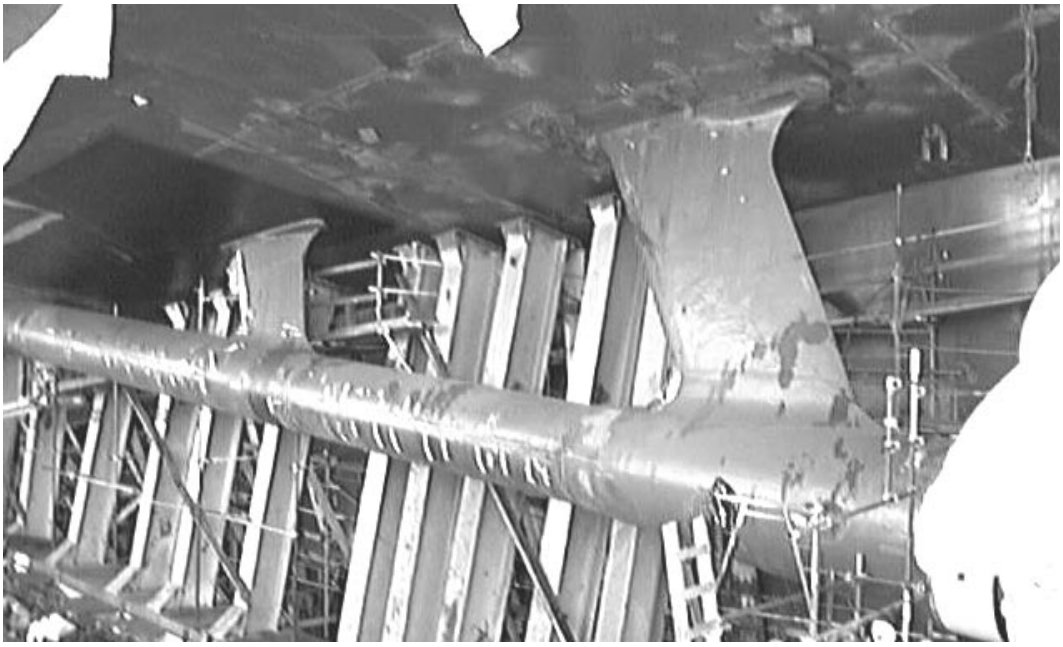


Figure 6 — Shaft strut

Figure 7 — Shaft struts

4.2.4 Approval_event

An Approval_event is an Event (see 4.2.33) that records a change in the status of the organizational review 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 (see 4.2.5) effected by the Approval_event. Every Approval_event shall refer to exactly one Approval_history. See 4.3.2 for the application assertion.

NOTE The approval_reference attribute is the inverse of the attribute approvals for Approval_history.

4.2.4.2 result

The result specifies the approval decision for a version of the design that requires approval.

The result shall be 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 accepted or certified for use in the ship.

4.2.4.2.2 rejected

The product data has been reviewed by the appropriate organization and is not accepted or not certified for use in the ship.

NOTE Other product data will 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 acceptance or certification by the organization.

4.2.4.2.4 user_defined

A project-specific approval status code, assigned to specific versions of product data, 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.

NOTE User_defined_result is not optional when the result of an Approval_event is USER_DEFINED.

4.2.5 Approval_history

An Approval_history is a collection of all Approval_events (see 4.2.4) 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 (see 4.2.4) that have occurred up to the present time. The Approval_history shall consist of at least one Approval_event. See 4.3.2 for the application assertion.

NOTE The sequence of Approval_events is assumed to build up in chronological order.

4.2.5.2 status

The status specifies the current state of approval.

The status shall be 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 accepted or certified for use in the ship.

4.2.5.2.2 rejected

The product data has been reviewed by the appropriate organization and is not accepted or not certified for use in the ship.

NOTE Other product data will 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 acceptance or certification by the organization.

4.2.5.2.4 user_defined

A project-specific approval status code, assigned to specific versions of product data, to be determined by two or more exchanging organizations.

4.2.5.3 subject

The subject specifies the product data this approval is related to. See 4.3.3 for the application assertion.

NOTE A Definition (see 4.2.23) may have zero, one or many associated Approval_history objects. In case the Definition has more then one associated Approval_history, no Approval_history objects will be the same.

4.2.6 Bottom_moulded_form_design_parameter

A Bottom_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.63) that contains dimensions and ratios of the keel and the bottom of a ship hull.

EXAMPLE Figure 8 illustrates keel dimensions of the keel and the bottom of a ship hull.

The data associated with a Bottom_moulded_form_design_parameter are the following:

— aft_end_of_flat_of_bottom;

— bilge_radius;

— flat_of_bottom_breadth;

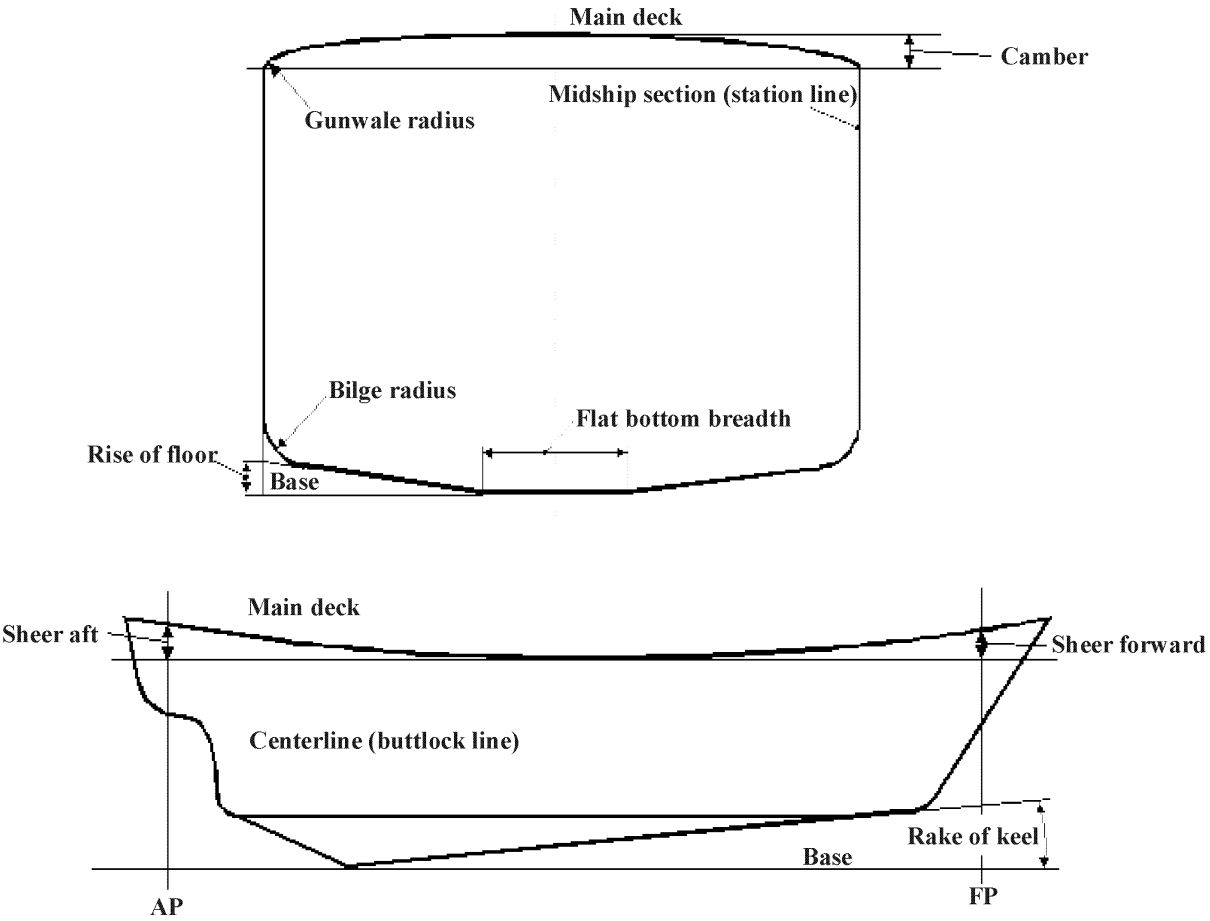


Figure 8 — Keel dimensions

— front_end_of_flat_of_bottom;

— length_of_flat_of_bottom;

— rake_of_keel;

— rise_of_floor.

4.2.6.1 aft_end_of_flat_of_bottom

The aft_end_of_flat_of_bottom specifies the longitudinal distance from the after perpendicular to the aft end of the flat bottom.

4.2.6.2 bilge_radius

The bilge_radius specifies the radius of the arc that joins the flat side of the hull to the bottom. This is measured at the transverse section amidships.

4.2.6.3 flat_of_bottom_breadth

The flat_of_bottom_breadth specifies the breadth of the flat bottom measured in transverse direction amidships.

4.2.6.4 front_end_of_flat_of_bottom

The front_end_of_flat_of_bottom specifies the longitudinal distance from the after perpendicular to the front end of the flat bottom.

4.2.6.5 length_of_flat_of_bottom

The length_of_flat_of_bottom specifies the length of the flat bottom measured horizontally at the centreline.

4.2.6.6 rake_of_keel

The rake_of_keel specifies the angle between the baseline and the keel of the ship.

4.2.6.7 rise_of_floor

The rise_of_floor specifies the vertical distance from the baseline to the point where the line of the continuation of the sloped bottom meets the continuation of the line of the side of the ship. The rise_of_-floor is measured at the transverse section amidships.

4.2.7 Bulb_moulded_form_design_parameter

A Bulb_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.63) that contains dimensions and ratios of the bulb of a ship hull

EXAMPLE Figure 9 illustrates dimensions of the bulbous bow of a ship hull.

The data associated with a Bulb_moulded_form_design_parameter are the following:

— bulb_breadth;

— bulb_breadth_pp;

— bulb_depth;

— bulb_depth_pp;

— bulb_frame_section_area_at_pp;

— bulb_length;

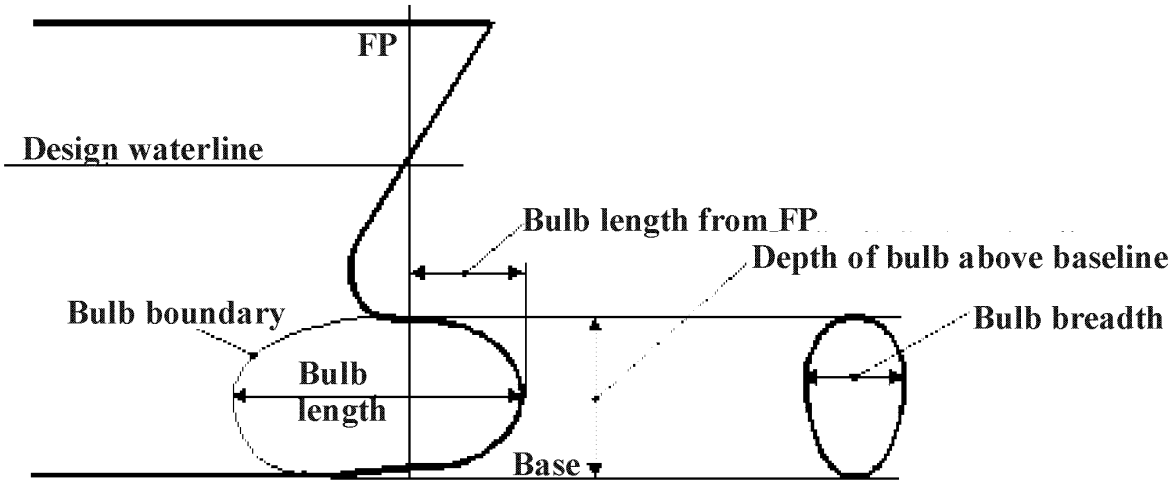


Figure 9 — Dimensions of the bulbous bow

— bulb_length_from_pp;

— bulb_location.

4.2.7.1 bulb_breadth

The bulb_breadth specifies the maximum breadth of the bulb.

4.2.7.2 bulb_breadth_pp

The bulb_breadth_pp specifies the breadth of the bulb at the aft or forward perpendicular, depending on the placement of the bulb.

4.2.7.3 bulb_depth

The bulb_depth specifies the maximum vertical distance from the baseline to the uppermost point of the bulb.

4.2.7.4 bulb_depth_pp

The bulb_depth_pp specifies the vertical distance from the baseline to the uppermost point of the bulb at the aft or forward perpendicular, depending on the placement of the bulb.

4.2.7.5 bulb_frame_section_area_at_pp

The bulb_frame_section_area_at_pp specifies the surface of the bulb in the frame section at the aft or forward perpendicular, depending on the placement of the bulb.

4.2.7.6 bulb_length

The bulb_length specifies the maximum length of the bulb measured horizontally at the centreline.

4.2.7.7 bulb_length_from_pp

The bulb_length_from_pp specifies the length of the bulb measured horizontally from the aft or forward perpendicular, depending on the placement of the bulb.

4.2.7.8 bulb_location

The bulb_location specifies whether the bulb is placed at the bow or at the stern.

The bulb_location shall be one of the following:

— bow;

— stern.

See 4.2.7.8.1 - 4.2.7.8.2 for the definition of each allowable value for bulb_location.

4.2.7.8.1 bow

The placement of the bulb is at the bow of the ship hull.

4.2.7.8.2 stern

The placement of the bulb is at the stern of the ship hull.

4.2.8 Buttock_table

A Buttock_table is a type of Transversal_table (see 4.2.113) that has positions that reference the placement of buttocks and are located on the global Y-axis.

4.2.9 Carrier

A Carrier is a type of Shiptype (see 4.2.98) that represents a Ship (see 4.2.88) that transports goods or passengers.

The data associated with a Carrier are the following:

— has_type.

4.2.9.1 has_type

The has_type specifies the type of the carrier ship.

The has_type shall be one of the following:

— barge;

— barge_for_deck_loading;

— barge_for_liquifiedgas;

— 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;

— liquified_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 4.2.9.1.1 - 4.2.9.1.29 for definition of allowable values for has_type.

4.2.9.1.1 barge

A Carrier that has no machinery for self-propulsion and is designed for transporting cargo.

4.2.9.1.2 barge_for_deck_loading

A Carrier designed as a service platform for loading decks on ships.

4.2.9.1.3 barge_for_liquefied_gas

A Carrier that has no machinery for self-propulsion and is designed for transporting liquefied gas.

4.2.9.1.4 barge_for_oil

A Carrier that has no machinery for self-propulsion and is designed for transporting oil.

4.2.9.1.5 barge_pontoon

A Carrier with pontoons coupled to a barge that serves as a floating utility platform.

4.2.9.1.6 bulk_carrier

A Carrier constructed for the transportation of bulk cargo.

4.2.9.1.7 car_carrier

A Carrier used for transporting automobiles from manufacturer to end-user.

4.2.9.1.8 car_ferry

A Carrier constructed to transport automobiles across water.

4.2.9.1.9 cargo_ship_carrying_passengers

A Carrier constructed for the transport of cargo and passengers.

4.2.9.1.10 chemical_tanker

A Carrier specially designed to transport chemicals.

4.2.9.1.11 chemical_tanker_type_1

A Carrier specially designed to transport Type 1 chemicals.

4.2.9.1.12 container_carrier

A Carrier constructed to transport containers.

4.2.9.1.13 cruise_liner

A Carrier constructed for the transport of passengers for pleasure.

4.2.9.1.14 dry_cargo_vessel

A Carrier constructed to transport dry, loose cargo.

EXAMPLE Types of dry cargo are grain or coal.

4.2.9.1.15 ferry

A Carrier constructed to transport passengers, cars, coaches, lorries and trains across water.

NOTE The ferry could be designed for only one type of cargo e.g., passengers.

4.2.9.1.16 gas_carrier

A Carrier specially designed for transporting gaseous products.

4.2.9.1.17 general_cargo_carrier

A Carrier specially designed for transporting general cargo.

NOTE The general_cargo_carrier carries any dry cargo which is not packed but carried loose such as grain or coal.

4.2.9.1.18 highspeedcraft_cargo

A Carrier designed for carrying cargo at high speeds.

4.2.9.1.19 highspeedcraft_passenger

A Carrier designed for carrying passengers at high speeds.

4.2.9.1.20 hydrofoil

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.

4.2.9.1.21 liquefied_gas_tanker

A Carrier specially designed for transporting liquefied gas products.

4.2.9.1.22 LNG_carrier

A Carrier specially designed to transport liquid natural gas.

4.2.9.1.23 LPG_carrier

A Carrier specially designed to transport liquid petroleum gas.

4.2.9.1.24 oil_tanker

A Carrier specially designed for transporting oil.

4.2.9.1.25 ore_carrier

A Carrier constructed for transporting ore

EXAMPLE A type of ore is iron ore.

4.2.9.1.26 passenger_vessel

A Carrier constructed for transporting passengers.

4.2.9.1.27 product_tanker

A Carrier specially designed for the transportation of oil products.

4.2.9.1.28 refrigerated_cargo_carrying_ship

A Carrier constructed for transporting freight which requires cooling or chilling.

4.2.9.1.29 roro_vessel

A Carrier constructed for the transport of roll-on and roll-off type vehicles.

4.2.9.1.30 tanker_for_refrigerated_fruit_juice

A Carrier specially designed for transporting fruit juices which require cooling or chilling.

4.2.9.1.31 user_defined

A Carrier type that is not in the list mentioned above (see 4.2.9.1.1- 4.2.9.1.30). Details can be found in the description attribute for Shiptype (see 4.2.98).

4.2.10 Centre_location

A Center_location is the X, Y, and Z measurement values for centre_of_buoyancy (see 4.2.48.1.1), centre_of_flotation (see 4.2.48.1.2), and centre_of_lateral_resistance (see 4.2.48.1.3).

4.2.11 Change

A Change is a type of Item (see 4.2.52) 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.

NOTE 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_definition (see 4.2.12) objects.

The data associated with a Change are the following:

— the_class.

4.2.11.1 the_class

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

EXAMPLE Types of classes are: Headquarter Modification Request or Engineering Change Proposal.

4.2.12 Change_definition

A Change_definition is a type of Definition (see 4.2.23) that is the generalization of the major discrete stages of a Change (see 4.2.11). A Change_definition may be a Change_plan (see 4.2.14), a Change_-realization (see 4.2.15) or a Change_request (see 4.2.16).

The data associated with a Change_definition are the following:

— author;

— date_time;

— defined_for;

— local_units.

4.2.12.1 author

The author specifies the person or organization responsible for the Change (see 4.2.11) activities during the period, lasting from the end of the previous , if it exists, up to the end of this Change_definition.

4.2.12.2 date_time

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

4.2.12.3 defined_for

The defined_for specifies the Change (see 4.2.11) to which the Change_definition is applicable. See 4.3.4 for the application assertion.

4.2.12.4 local_units

The local_units specifies the units that are used by the definition and differ from the units globally defined for the ship. Local_units may be either a Derived_unit (see 4.2.24) or a Named_unit (see 4.2.70). Local_units need not be specified for a particular Change_definition. There may be more than one local_units for a Change_definition. See 4.3.5 and 4.3.6 for application assertions.

4.2.13 Change_impact

A Change_impact is a set of events which defines the effect a Change (see 4.2.11) will cause or has caused.

The data associated with a Change_impact are the following:

— impact.

4.2.13.1 impact

The impact specifies the effect of a Change (see 4.2.11) in terms of the creation, modification or deletion of some Definition (see 4.2.23) objects, Item_structure (see 4.2.54) objects, or Item_relationship (see 4.2.53) objects. See 4.3.7 for the application assertion.

4.2.14 Change_plan

A Change_plan is a type of Change_definition (see 4.2.12) that represents the proposed solution for a Change (see 4.2.11). A Change_plan is the basis for the activities 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.

4.2.14.1 checks

The checks specifies the verifications planned for the Change (see 4.2.11). See 4.3.9 for the application assertion.

4.2.14.2 chosen_solution_for

The chosen_solution_for specifies the identification of the Change_request (see 4.2.16) for which a Change_plan is applicable. See 4.3.10 for the application assertion.

4.2.14.3 planned_impact

The planned_impact specifies the estimated or calculated effects of the Change (see 4.2.11). See 4.3.8 for the application assertion.

NOTE This impact is usually chosen from the set of solution_alternatives of a Change_request (see 4.2.16).

4.2.15 Change_realization

A Change_realization is a type of Change_definition (see 4.2.12) that defines the actual, observed effects of the Change (see 4.2.11).

The data associated with a Change_realization are the following:

— checks;

— impact;

— realization_of.

4.2.15.1 checks

The checks specifies the organizational approval of the product model revisions made to implement the Change (see 4.2.11). See 4.3.12 for the application assertion.

4.2.15.2 impact

The impact specifies the identification of the revisions made to the product model. See 4.3.11 for the application assertion.

4.2.15.3 realization_of

The realization_of specifies the Change_plan (see 4.2.14) for which a product model change is being implemented. See 4.3.13 for the application assertion.

4.2.16 Change_request

A Change_request is a type of Change_definition (see 4.2.12) that represents the first phase of a Change (see 4.2.11), 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.

4.2.16.1 addressee

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

4.2.16.2 initiator

The initiator specifies the person and or person and organization where the request is coming from.

4.2.16.3 problem

The problem specifies a textual description of the problem that induced the request.

4.2.16.4 solution_alternatives

The solution_alternatives specifies the alternative solutions proposed to solve the problem. A solution is described in terms of the effect on the Versionable_object (see 4.2.120) objects. See 4.3.14 for the application assertion.

4.2.16.5 solution_description

The solution_description specifies a textual description of one or more possible solutions for the problem. The solution_description need not be specified for a particular Change_request.

NOTE This textual description should be present if the solution_alternatives are not yet established, or may enhance the information provided by the solution_alternatives.

4.2.17 Check

A Check is a type of Event (see 4.2.33) that defines the details of a planned or fulfilled approval with an organization for a Change_plan (see 4.2.14) or a Change_realization (see 4.2.15).

4.2.18 Class_and_statutory_designation

A Class_and_statutory_designation is a type of General_characteristics_definition (see 4.2.40) that specifies the identification given to the Ship (see 4.2.88) by the classification society for the purpose of design, manufacture and in service approval.

The data associated with a Class_and_statutory_designation are the following:

— class_number;

— local_units;

— the_class;

— the_statutory.

4.2.18.1 class_number

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

4.2.18.2 local_units

The local_units specifies the units that are used by the definition and differ from the units globally defined for the ship. Each local_units may be one of the following: a Derived_unit (see 4.2.24) or a Named_unit (see 4.2.70). Local_units need not be specified for a particular Class_and_statutory_designation. There may be more than one local_units for a Class_and_statutory_designation. See 4.3.17 and 4.3.16 for application assertions.

4.2.18.3 the_class

The the_class specifies the applicable Class_notation (see 4.2.19) with information about the ship type and the
cargo. See 4.3.15 for the application assertion.

4.2.18.4 the_statutory

The the_statutory specifies the set of national and international regulations and standards with which the ship is intended to comply. See 4.3.18 for the application assertion.

4.2.19 Class_notation

A Class_notation is the notations given to the ships hull and machinery 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.

4.2.19.1 approval_required_for_heavy_cargo

The approval_required_for_heavy_cargo specifies 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 approval_required_for_heavy_cargo shall be one of the following:

— HC;

— HC_E;

— HC_EA.

NOTE See 4.2.19.1.1 - 4.2.19.1.3 for the definition of each allowable value for approval_required_for_heavy_-cargo.

4.2.19.1.1 HC

The cargo carrier is strengthened for heavy cargo. The heavy bulk cargo may be unevenly distributed among the cargo holds.

4.2.19.1.2 HC_E

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

EXAMPLE A usage of HC_E is: Holds 2, 3, 5 empty.

4.2.19.1.3 HC_EA

The cargo carrier is strengthened for heavy cargo, and any cargo hold may be empty at full draught. The approved combinations of empty holds are added to the notation.

EXAMPLE A usage of HC_EA is: holds 2, 3, 5 empty or Holds 1 through 6 empty.

4.2.19.2 approval_required_for_oil_cargo

The approval_required_for_oil_cargo specifies whether or not approval is required for the carriage of oil cargoes.

4.2.19.3 approval_required_loading_unloading_aground

The approval_required_loading_unloading_aground specifies whether or not approval for loading and unloading aground is necessary.

4.2.19.4 approval_required_loading_unloading_grabs

The approval_required_loading_unloading_grabs specifies whether or not approval for loading and unloading using grabs is necessary.

4.2.19.5 class_notations_hull

The class_notations_hull specifies the values 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.

4.2.19.6 class_notations_machinery

The class_notations_machinery specifies the values 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 on class_notations_machinery for a Class_notation.

4.2.19.7 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.

4.2.19.8 ice_class_notation

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

4.2.19.9 service_area

The service_area specifies the area or route in which the ship operates.

NOTE This may include information about waterway, wave, weather and wind conditions.

4.2.19.10 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 1.0. The service_factor need not be specified for a particular Class_notation.

4.2.20 Class_parameters

A Class_parameters is a type of General_characteristics_definition (see 4.2.40) that specifies the length and speed 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.

4.2.20.1 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.

4.2.20.2 design_speed_ahead

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

4.2.20.3 design_speed_astern

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

4.2.20.4 length_class

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

4.2.20.5 length_solas

The length_solas specifies a length measurement for the ship measured in accordance with the international convention on the safety of life at sea.

4.2.20.6 scantlings_draught

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

4.2.21 Deck_moulded_form_design_parameter

A Deck_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.63) that specifies the main parameter of the deck.

The data associated with a Deck_moulded_form_design_parameter are the following:

— camber;

— defined_for;

— sheer_at_AP;

— sheer_at_FP.

4.2.21.1 camber

The camber specifies the transverse curvature of a deck at the midship section. It is measured as the difference between the deck height at the centre plane of the hull and the height of the deck at the edge where it joins the flat side of the hull.

4.2.21.2 defined_for

The defined_for specifies the Moulded_form (see 4.2.61) for which the Deck_moulded_form_design_-parameter are defined. See 4.3.19 for the application assertion.

4.2.21.3 sheer_at_AP

The sheer_at_AP specifies the longitudinal curvature of the deck in the aft half of the ship. It is measured in the centre plane of the hull as the difference between the deck height amidships and the height of the deck at the after perpendicular.

4.2.21.4 sheer_at_FP

The sheer_at_FP specifies the longitudinal curvature of the deck in the front half of the ship. It is measured in the centre plane of the hull as the difference between the deck height amidships and the height of the deck at the forward perpendicular.

4.2.22 Definable_object

A Definable_object is any type of product that the basic properties can be described. Each Definable_-object is either an Item (see 4.2.52), an Item_relationship (see 4.2.53) or an Item_structure (see 4.2.54).

The data associated with a Definable_object are the following:

— id.

4.2.22.1 id

The id specifies the globally unique identifier for the Definable_object. See 4.3.20 for the application assertion.

4.2.23 Definition

A Definition is a type of Versionable_object (see 4.2.120) that is the basis for all types of Definable_object (see 4.2.22) objects. Each Definition is either a Design_definition (see 4.2.25), a Functional_definition (see 4.2.39), a General_characteristics_definition (see 4.2.40), a Moulded_form_-characteristics_definition (see 4.2.64), a Change_definition (see 4.2.12), a Spacing_table (see 4.2.102) or a Local_co_ordinate_system (see 4.2.56). These definitions support the following shipbuilding concepts: 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.

4.2.23.1 defined_for

The defined_for specifies the Definable_object (see 4.2.22) objects, which are defined by a Definition. See 4.3.21 for the application assertion.

4.2.23.2 id

The id specifies the globally unambiguous identifier for the definition. See 4.3.23 for the application assertion.

4.2.23.3 local_units

The local_units specifies the units that are used by the definition and differ from the units globally defined for the ship. Local_units may be either a Derived_unit (see 4.2.24) or a Named_unit (see 4.2.70). Local_units need not be specified for a particular Definition. There may be more than one local_units for a Definition. See 4.3.22 and 4.3.24 for application assertions.

4.2.24 Derived_unit

A Derived_unit is a unit of measure that is composed of elements that are pre-defined Named_unit (see 4.2.70) objects with exponents.

4.2.25 Design_definition

A Design_definition is a type of Definition (see 4.2.23) 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 Moulded_form_design_definition (see 4.2.64), a Hydrostatic_definition (see 4.2.45) or a Stability_definition (see 4.2.103).

The data associated with a Design_definition are the following:

— representations.

4.2.25.1 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.

EXAMPLE A Design_definition may have multiple shape representations like wireframe or surface.

4.2.26 Displacement_operation

A Displacement_operation is a calculation for modifying the displacement of a Ship_moulded_form (see 4.2.93). The modification is described by the addition or subtraction of a Moulded_form (see 4.2.61). A Displacement_operation may be an Addition_of_moulded_form (see 4.2.1) or a Subtraction_of_-moulded_form (see 4.2.108).

The data associated with a Displacement_operation are the following:

— displacement_of_moulded_form_to_add_or_subtract.

4.2.26.1 displacement_of_moulded_form_to_add_or_subtract

The displacement_of_moulded_form_to_add_or_subtract specifies the Moulded_form (see 4.2.61) that is either added or subtracted to the Ship_moulded_form (see 4.2.93). The displacement of the Moulded_-form can be found in the specific moulded form characteristics. See 4.3.25 for the application assertion.

4.2.27 Document

A Document is a type of Versionable_object (see 4.2.120) that represents an unambiguous identification of some human readable data item defined outside ISO 10303. A document has an author, a title and may be versioned.

The data associated with a Document are the following:

— author;

— source_type;

— summary;

— title.

4.2.27.1 author

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

4.2.27.2 source_type

The source_type specifies the type of the document, whether it is a printed document, or an electronic document including the type.

EXAMPLE Types of electronic document are PDF, Word, WordPerfect, Text, etc.

4.2.27.3 description

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

4.2.27.4 title

The title specifies a descriptive title consisting of a word, or group of words.

4.2.28 Document_portion

A Document_portion is the qualification of a specific subset or portion of a Document (see 4.2.27) in generic terms.

EXAMPLE A subsection of a report could be a range of pages identified by their numbers.

The data associated with a Document_portion are the following:

— element_type;

— element_value;

— source.

4.2.28.1 element_type

The element_type specifies the name for this subset of the Document (see 4.2.27).

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

4.2.28.2 element_value

The element_value specifies the value for this subset of the Document (see 4.2.27).

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

4.2.28.3 source

The source specifies the Document (see 4.2.27) to which the specified subset is related. See 4.3.26 for application assertion.

4.2.29 Document_reference

A Document_reference is a qualification of a Document (see 4.2.27) in terms of its source or location. A Document_reference may be a Document_reference_with_address (see 4.2.30).

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.

4.2.29.1 assigned_document

The assigned_document specifies the Document (see 4.2.27), or Document_portion (see 4.2.28), that is to be associated with the product data. See 4.3.27 and 4.3.28 for the application assertions.

4.2.30 Document_reference_with_address

A Document_reference_with_address is a type of Document_reference (see 4.2.29) and External_-reference (see 4.2.35) that specifies a pointer to a location inside the source.

4.2.31 Edge_based_wireframe_shape

An Edge_based_wireframe_shape is a shape representation that conforms to ISO 10303-501.

4.2.32 Envisaged_version_creation

An Envisaged_version_creation is a type of Versionable_object_change_event (see 4.2.121) that is the event leading to a new Versionable_object (see 4.2.120). The event is an understood event that has not yet happened. The Definition (see 4.2.23), Item_structure (see 4.2.54) or Item_relationship (see 4.2.53) as the subjects of the event do not yet exist and are described in terms of descriptive, non-formal properties.

The data associated with an Envisaged_version_creation are the following:

— base;

— category.

4.2.32.1 base

The base specifies the Versionable_object (see 4.2.120) objects the envisaged new version is derived from. See 4.3.29 for the application assertion.

4.2.32.2 category

The category specifies the classification the envisaged definition belongs to.

4.2.33 Event

An Event is the identification that something has happened at a certain time, activated by a certain person or organization for a certain reason. An Event may be one of the following: Check (see 4.2.17), a Versionable_object_change_event (see 4.2.121) or an Approval_event (see 4.2.4).

The data associated with an Event are the following:

— caused_by;

— caused_when;

— description.

4.2.33.1 caused_by

The caused_by specifies the person or organization creating the Event.

4.2.33.2 caused_when

The caused_when specifies the date and time the Event occurred.

4.2.33.3 description

The description specifies additional textural information for the Event.

4.2.34 External_instance_reference

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

The data associated with an External_instance_reference are the following:

— entity_type;

— schema_name;

— target_GUID.

4.2.34.1 entity_type

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

4.2.34.2 schema_name

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

4.2.34.3 target_GUID

The target_GUID specifies the globally unambiguous identifier of the externally referenced instance. See 4.3.30 for the application assertion.

4.2.35 External_reference

An External_reference is the abstract denotation of a data source external to the data set where an instance of this object exists. An External_reference may be a Document_reference_with_address (see 4.2.30).

EXAMPLE A Universal_resource_locator (see 4.2.114) denotes such a data source.

The data associated with an External_reference are the following:

— description;

— location.

4.2.35.1 description

The description specifies some additional information regarding the External_reference.

4.2.35.2 location

The location specifies the location of the external reference. In the case of an Universal_resource_locator (see 4.2.114), the location is computer accessible by a specified transmission protocol. See 4.3.31 and 4.3.32 for the application assertions.

4.2.36 External_storage

A External_storage is the location of physical documents or items external to the current exchange. The object may be relevant, when identifying the location of CD_ROMs, floppy disks, or video tapes.

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

The data associated with an External_storage are the following:

— location.

4.2.36.1 location

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

4.2.37 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.

4.2.37.1 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.

4.2.37.2 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.

4.2.37.3 breadth_of_waterline

The breadth_of_waterline specifies the breadth of the current waterline.

4.2.37.4 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.

4.2.37.5 length_of_waterline

The length_of_waterline specifies the length of the current waterline.

4.2.37.6 moulded_form_displacement

The moulded_form_displacement specifies the wetted displacement of the ship.

4.2.38 Frame_table

A Frame_table is a type of Longitudinal_table (see 4.2.59) that has positions that reference the placement of frames that are located on 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.

4.2.39 Functional_definition

The Functional_definition is a type of Definition (see 4.2.23) that is the basis for all types of functional performance product data. It provides the capability to specify a role or purpose for a Definition. Each Functional_definition is either a Moulded_form_functional_definition (see 4.2.65) or a Ship_type (see 4.2.98).

The data associated with a Functional_definition are the following:

— local_units;

— user_def_function.

4.2.39.1 local_units

The local_units specifies the units that are used by the definition and differ from the units globally defined for the ship. Each Local_units is either a Derived_unit (see 4.2.24), or a Named_unit (see 4.2.70). The local_units need not be specified for a particular Functional_definition. There may be more than one local_units for a Functional_definition. See 4.3.33 and 4.3.34 for the application assertions.

4.2.39.2 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.

4.2.40 General_characteristics_definition

The General_characteristics_definition is a type of Definition (see 4.2.23) that provides a major part of the documentation of a ship. It includes primary dimensions and capacities due to the contract of the Ship (see 4.2.88). Each General_characteristics_definition is either a Class_and_statutory_designation (see 4.2.18), an Owner_designation (see 4.2.75), a Ship_designation (see 4.2.92), a Shipyard_designation (see

4.2.99), Ship_overall_dimensions (see 4.2.95), Principal_characteristics (see 4.2.78), Class_parameters (see 4.2.20) or a Global_axis_placement (see 4.2.41).

The data associated with a General_characteristics_definition are the following:

— defined_for.

4.2.40.1 defined_for

The defined_for specifies the Ship (see 4.2.88) for which the General_characteristics_definition applies. There may be more than one defined_for for a Ship. See 4.3.35 for the application assertion.

4.2.41 Global_axis_placement

A Global_axis_placement is a type of General_characteristics_definition (see 4.2.40) 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 upwards 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 and in the other it is directed from the forward part of the ship to the after 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

EXAMPLE Figure 10 illustrates a global axis for a ship.

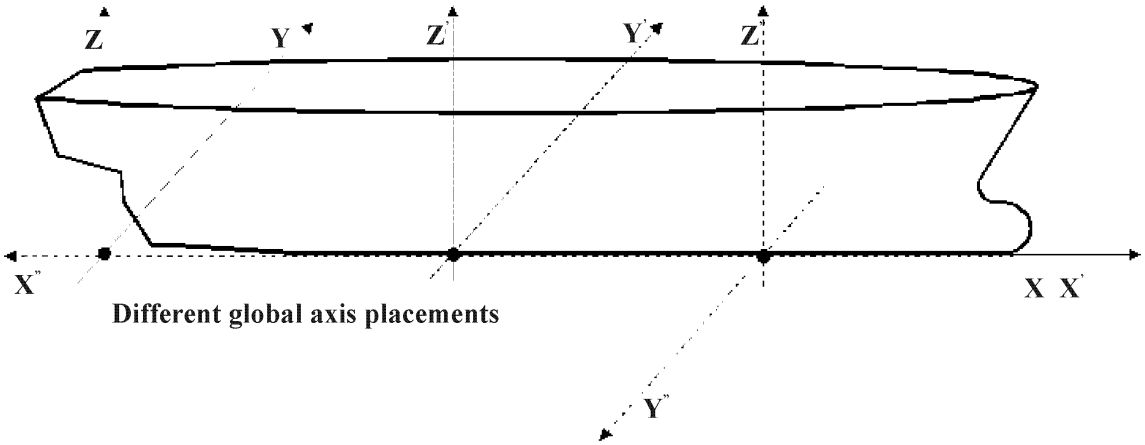


Figure 10 — Global axis placements

The data associated with a Global_axis_placement are the following:

— after_perpendicular_offset;

— orientation.

4.2.41.1 after_perpendicular_offset

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

4.2.41.2 orientation

The orientation specifies the direction of the X-axis.

The orientation shall be one of the following:

— aft_pointing;

— forward_pointing.

NOTE See 4.2.42.2.1 - 4.2.42.2.2 for the definition of each allowable value for orientation.

4.2.41.2.1 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.

4.2.41.2.2 forward_pointing

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

4.2.42 Global_id

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

The data associated with a Global_id are the following:

— id.

4.2.42.1 id

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

4.2.43 Hull_applicability

A Hull_applicability is the identification of a ship hull, or a range of hulls within a class of ships, for which a 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.

4.2.43.1 definitions_for_hulls

The definitions_for_hulls specifies the Definition (see 4.2.23) objects that are applicable for the range of hulls specified in start_hull and end_hull. See 4.3.36 for the application assertion.

4.2.43.2 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.

4.2.43.3 items_for_hulls

The items_for_hulls specifies the Item (see 4.2.52) objects that are applicable for the range of hulls specified in start_hull and end_hull. See 4.3.37 for the application assertion.

4.2.43.4 start_hull

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

4.2.44 Hull_moulded_form_design_parameter

A Hull_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.63) that contain dimensions and ratios of the ship hull

EXAMPLE Figure 43 illustrates hull dimensions for a ship.

The data associated with a Hull_moulded_form_design_parameter are the following:

— aft_end_of_flat_of_side;

— aft_end_of_parallel_midbody_at_design_draught;

— block_coefficient;

— front_end_of_flat_of_side;

— front_end_of_parallel_midbody_at_design_draught;

— gunwale_radius;

— hull_breadth;

— hull_depth;

— hull_design_draught;

— hull_length_pp;

— hull_length_waterline;

— length_of_flat_of_side;

— length_of_parallel_midbody_at_design_draught;

— length_to_beam_ratio;

— max_frame_section_area_location;

— max_wetted_frame_section_area;

— midship_tumble_data;

— prismatic_coefficient;

— waterline_angle_of_entrance_at_bow;

— waterline_angle_of_entrance_at_stern;

— waterplane_coefficient.

4.2.44.1 aft_end_of_flat_of_side

The aft_end_of_flat_of_side specifies the longitudinal distance from the after perpendicular to the aft end of the flat of side, where the ship hull is of vertical shape.

4.2.44.2 aft_end_of_parallel_midbody_at_design_draught

The aft_end_of_parallel_midbody_at_design_draught specifies the longitudinal distance from the after perpendicular to the aft end of the wetted midship body with sections of constant shape.

4.2.44.3 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.

4.2.44.4 front_end_of_flat_of_side

The front_end_of_flat_of_side specifies the longitudinal distance from the after perpendicular to the front end of the flat of side, where the ship hull is of vertical shape.

4.2.44.5 front_end_of_parallel_midbody_at_design_draught

The front_end_of_parallel_midbody_at_design_draught specifies the longitudinal distance from the after perpendicular to the front end of the wetted midship body with sections of constant shape.

4.2.44.6 gunwale_radius

The gunwale_radius specifies the radius of the arc that joins the side of the hull with the upper deck. This is measured at the transverse section amidships. The gunwale_radius need not be specified for a particular Hull_moulded_form_design_parameter.

4.2.44.7 hull_breadth

The hull_breadth specifies the maximum breadth of the individual ship hull amidships and at the design_-draught. The hull_breadth need not be specified for a particular Hull_moulded_form_design_parameter.

NOTE The hull_breadth need not be specified for a particular Hull_moulded_form_design_parameter for normal shapes where the hull_breadth is identical to the moulded_breadth in the Principal_characteristics (see 4.2.78). This parameter is only necessary for special ships e.g., multi hull ships with different ship hulls and different hull breadths.

4.2.44.8 hull_depth

The hull_depth specifies for each individual ship hull the vertical distance above the baseline to the uppermost deck where the deck joins the side of the ship. The hull_depth need not be specified for a particular Hull_moulded_form_design_parameter.

NOTE The hull_depth need not be specified for a particular Hull_moulded_form_design_parameter for normal shapes where the hull_depth is identical to the moulded_depth in the Principal_characteristics (see 4.2.78). This parameter is only necessary for special ships e.g., multi hull ships with different ship hulls and different hull_depths.

4.2.44.9 hull_design_draught

The hull_design_draught specifies the draught to which the individual ship hull has been designed to operate. The hull_design_draught need not be specified for a particular Hull_moulded_form_design_-parameter.

NOTE The hull_design_draught need not be specified for a particular Hull_moulded_form_design_parameter for normal shapes where the hull_design_draught is identical to the design_draught in the Principal_characteristics (see 4.2.78). This parameter is only necessary for special ships e.g., multi hull ships with different ship hulls and different hull_design_draughts.

4.2.44.10 hull_length_pp

The hull_length_pp specifies the length measured from the after perpendicular to the forward perpendicular of the individual ship hull. The hull_length_pp need not be specified for a particular Hull_moulded_form_design_parameter.

NOTE The hull_length_pp need not be specified for a particular Hull_moulded_form_design_parameter for normal shapes where the hull_length_pp is identical to the length_between_perpendiculars in the Principal_characteristics (see 4.2.78). This parameter is only necessary for special ships e.g. multi hull ships with different ship hulls and different hull_length_pps.

4.2.44.11 hull_length_waterline

The hull_length_waterline specifies the overall length of the hull measured along the longitudinal axis of the ship at the design_draught waterline. The hull_length_waterline need not be specified for a particular Hull_moulded_form_design_parameter.

4.2.44.12 length_of_flat_of_side

The length_of_flat_of_side specifies the length of the flat of side where the ship hull is of vertical shape.

4.2.44.13 length_of_parallel_midbody_at_design_draught

The length_of_parallel_midbody_at_design_draught specifies the length of the wetted midship body with sections of a constant shape.

4.2.44.14 length_to_beam_ratio

The length_to_beam_ratio specifies the ratio of the hull length over the breadth of the ship's draught at the design waterline.

4.2.44.15 max_frame_section_area_location

The max_frame_section_area_location specifies the longitudinal distance of the frame section with the maximum immersed area at design draught from the after perpendicular. The max_frame_section_area_-location need not be specified for a particular Hull_moulded_form_design_parameter.

NOTE Most ships with parallel midship body the max_frame_section_area_location is amidships at one half of length between perpendiculars.

4.2.44.16 max_wetted_frame_section_area

The max_wetted_frame_section_area specifies the maximum wetted area of a frame section at design draught.

NOTE For normal shaped ship hulls this is the frame section amidships. For special shapes, e.g. sailing yachts, the max_wetted_frame_section_area can be a section located at 1/3 of the ships length from the stern.

4.2.44.17 midship_tumble_data

The midship_tumble_data specifies specific design parameter used for the definition of the midship section. The midship_tumble_data need not be specified for a particular Hull_moulded_form_design_-parameter. See 4.3.38 for the application assertion.

4.2.44.18 prismatic_coefficient

The prismatic_coefficient specifies the ratio of the moulded displacement volume to the volume of a prism having a length equal to the length between perpendiculars and a cross-sectional area equal to that of the cross-sectional area of the hull amidships.

4.2.44.19 waterline_angle_of_entrance_at_bow

The waterline_angle_of_entrance_at_bow specifies the angle between the tangent of the horizontal projection of the hull form at the ship's bow on the design waterline and the X-axis. The waterline_-angle_of_entrance_at_bow need not be specified for a particular Hull_moulded_form_design_parameter.

4.2.44.20 waterline_angle_of_entrance_at_stern

The waterline_angle_of_entrance_at_stern specifies the angle between the tangent of the horizontal projection of the hull form at the ship's stern on the design waterline and the X-axis. The waterline_-angle_of_entrance_at_stern need not be specified for a particular Hull_moulded_form_design_parameter.

4.2.44.21 waterplane_coefficient

The waterplane_coefficient specifies the ratio of the ship's area at the design draught waterline to the area of a rectangle that has its length equal to the length between perpendiculars and its breadth equal to the moulded breadth at the design draught.

4.2.45 Hydrostatic_definition

A Hydrostatic_definition is a type of Design_definition (see 4.2.25) which describes a relationship between a Ship_moulded_form (see 4.2.93), a set of modifications to the displacement of that Ship_-moulded_form and different Hydrostatic_table (see 4.2.51) objects. The modifications will be due to different combinations of inlets and appendages with the Ship_moulded_form.

The data associated with a Hydrostatic_definition are the following:

— defined_for;

— displacement_changes;

— displacement_changes_description;

— representations.

4.2.45.1 defined_for

The defined_for specifies the Ship (see 4.2.88) for which the Hydrostatic_definition is defined. See 4.3.41 for the application assertion.

4.2.45.2 displacement_changes

The displacement_changes specifies the list of Displacement_operation (see 4.2.26) objects that detail additions or subtractions to the Ship_moulded_form (see 4.2.93) due to combinations of inlets and appendages. There may be more than one displacement_changes for a Hydrostatic_definition. See 4.3.39 for the application assertion.

EXAMPLE For a hull with a bow thruster tunnel, the hydrostatic properties will be modified by the inclusion of the thruster tunnel into the ship hull. Such an inclusion would be a displacement change.

4.2.45.3 displacement_changes_description

The description specifies the textual description of the change in the displacement. The displacement_changes_description need not be specified for a particular Hydrostatic_definition.

4.2.45.4 representations

The representations specifies the Hydrostatic_table (see 4.2.51) objects that represent the Hydrostatic_-definition. See 4.3.40 for the application assertion.

4.2.46 Hydrostatic_position_value

A Hydrostatic_position_value is a type of Hydrostatic_property_value (see 4.2.49) that specifies the placement of a hydrostatic property value in the global coordinate system. When applicable, a Hydrostatic_property (see 4.2.48) is said to act through its Hydrostatic_position_value.

The data associated with a Hydrostatic_position_value are the following:

— position.

4.2.46.1 position

The position specifies the placement in the global coordinate system through which the Hydrostatic_-property (see 4.2.48) acts. See 4.3.42 for the application assertion.

4.2.47 Hydrostatic_properties_for_constant_floating_position

A Hydrostatic_properties_for_constant_floating_position is the hydrostatic properties for one specific Floating_position (see 4.2.37).

The data associated with a Hydrostatic_properties_for_constant_floating_position are the following:

— definition_of_floating_position;

— hydrostatic_property_values;

— related_hydrostatic_table.

4.2.47.1 definition_of_floating_position

The definition_of_floating_position specifies the Floating_position (see 4.2.37) for which the Hydrostatic_property_value (see 4.2.49) objects are valid. See 4.3.43 for the application assertion.

4.2.47.2 hydrostatic_property_values

The hydrostatic_property_values specifies the hydrostatic properties for the ship for the specific Floating_position (see 4.2.37). See 4.3.44 for the application assertion.

4.2.47.3 related_hydrostatic_table

The related_hydrostatic_table specifies the set of Hydrostatic_table (see 4.2.51) objects with which the Hydrostatic_properties_for_constant_floating_position are associated. See 4.3.46 for the application assertion.

NOTE The related_hydrostatic_table attribute is the inverse of the attribute items for Hydrostatic_table.

4.2.48 Hydrostatic_property

A Hydrostatic_property is a hydrostatic parameter used to describe the hydrostatic characteristic of a ship. All hydrostatic properties are dependent on the Floating_position (see 4.2.37) of the ship.

The data associated with a Hydrostatic_property are the following:

— property_type;

— property_unit.

4.2.48.1 property_type

The property_type specifies the type of parameter of interest.

The property_type shall be one of the following:

— centre_of_buoyancy;

— centre_of_flotation;

— centre_of_lateral_resistance;

— longitudinal_metacentric_height;

— longitudinal_second_moment_of_area_of_waterplane;

— mid_ship_section_area;

— moment_for_unit_change_of_trim;

— transverse_metacentric_height;

— transverse_second_moment_of_area_of_waterplane;

— waterplane_area;

— wetted_surface_area.

4.2.48.1.1 centre_of_buoyancy

The parameter specifies the centroid of the volume displaced by the ship, measured in the global coordinate system.

NOTE The centre of buoyancy is symbolized by the uppercase B

4.2.48.1.2 centre_of_flotation

The parameter specifies the geometric centroid of the waterplane area of the hull moulded form, measured in the global coordinate system.

4.2.48.1.3 centre_of_lateral_resistance

The parameter specifies the placement of the centre of all forces caused by the lateral resistance of the ship if it moves sideways. This placement is a property of the shape of the hull moulded form and is measured in the global coordinate system. This placement is of interest for instance for the design of thrusters at the bow and the stern of the ship.

4.2.48.1.4 longitudinal_metacentric_height

The parameter specifies the vertical height from the centre of gravity of the ship to the longitudinal meta-centre. The longitudinal meta-centre is defined as the intersection point of the plane amidships with the line of action of the correcting force through the centre of buoyancy that results from a pitch angle.

4.2.48.1.5 longitudinal_second_moment_of_area_of_waterplane

The parameter specifies the second moment of the area of the waterplane about the transverse axis.

4.2.48.1.6 mid_ship_section_area

The parameter specifies the area of the immersed part of the cross section of a hull moulded form at amidships.

4.2.48.1.7 moment_for_unit_change_of_trim

The parameter specifies the moment about the transverse axis required to change the difference between fore and aft draughts by one unit.

4.2.48.1.8 transverse_metacentric_height

The parameter specifies the vertical height from the centre of gravity of the ship to the transverse meta-centre. The transverse meta-centre is defined as the intersection point of the centre plane with the line of action of the correcting force through the centre of buoyancy that results from a roll angle.

EXAMPLE Figure 11 illustrates the centre of gravity of the ship to the transverse meta-centre. Figure 50 illustrates the form stability.

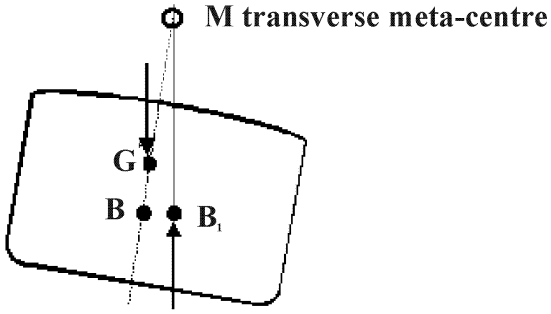


Figure 11 — Transverse meta- centre

4.2.48.1.9 transverse_second_moment_of_area_of_waterplane

The parameter specifies the second moment of the area of the waterplane about the longitudinal axis.

4.2.48.1.10 waterplane_area

The parameter specifies the enclosed area of the plane formed by the intersection of the hull moulded form with the surface of the water.

4.2.48.1.11 wetted_surface_area

The parameter specifies the area of the outer surface of the ship that is in contact with water.

4.2.48.2 property_measure

The property_measure specifies the measurement unit of the Hydrostatic_property. A property_measure may be a Centre_location (see 4.2.10). See 4.3.45 for the application assertion.

4.2.49 Hydrostatic_property_value

A Hydrostatic_property_value is a hydrostatic parameter, which describes a specific hydrostatic characteristic for a ship. A Hydrostatic_property_value is dependent on the Floating_position (see 4.2.37) of the ship. A Hydrostatic_property_value may be a Hydrostatic_position_value (see 4.2.46) or a Hydrostatic_scalar_value (see 4.2.50).

4.2.50 Hydrostatic_scalar_value

A Hydrostatic_scalar_value is a type of Hydrostatic_property_value (see 4.2.49) that specifies a Hydrostatic_property_value that is measured by a value without direction.

The data associated with a Hydrostatic_scalar_value are the following:

— scalar.

4.2.50.1 scalar

The scalar specifies the value for the Hydrostatic_property (see 4.2.48).

4.2.51 Hydrostatic_table

A Hydrostatic_table is the hydrostatic properties and values for a given Hydrostatic_definition (see 4.2.45). A Hydrostatic_table consists of a list of Hydrostatic_properties_for_constant_floating_position (see 4.2.47), where each entry covers all hydrostatic properties of a ship for a different Floating_position (see 4.2.37). Because the hydrostatic properties are related to the extreme form, a uniform shell thickness is used to generate the extreme form from the moulded form.

The data associated with a Hydrostatic_table are the following:

— items;

— mean_shell_thickness;

— name;

— properties_in_the_hydrostatic_table.

4.2.51.1 items

The items specifies the Hydrostatic_properties_for_constant_floating_position (see 4.2.47) that describes all hydrostatic properties for a ship, for a specific Floating_position (see 4.2.37). See 4.3.46 for the application assertion.

4.2.51.2 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 hydrostatic properties for the ship including the thickness.

EXAMPLE Figure 11 illustrates measuring shell thickness.

4.2.51.3 name

The name specifies a user or system defined name for the Hydrostatic_table.

Figure 12 — Measuring shell thickness

4.2.51.4 properties_in_the_hydrostatic_table

The properties_in_the_hydrostatic_table specifies the list of Hydrostatic_property (see 4.2.48) objects for the Hydrostatic_table. See 4.3.47 for the application assertion.

4.2.52 Item

An Item is a type of Definable_object (see 4.2.22) 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 needs not represent a

physical realizable object; it may also represent some abstract concept. Item provides the functionality to have relationships to other Items and to be member in an Item_structure (see 4.2.54). Each Item is either a Change (see 4.2.11), a Ship (see 4.2.88), a Ship_moulded_form (see 4.2.93) or a Moulded_form (see 4.2.61).

The data associated with an Item are the following:

— description;

— documentation;

— name;

— ship_context.

4.2.52.1 description

The description specifies the textual characterization of the Item. The description need not be specified for a particular Item.

4.2.52.2 documentation

The documentation specifies additional supporting document references available for the Item. See 4.3.48 for the application assertion.

4.2.52.3 name

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

4.2.52.4 ship_context

The ship_context specifies an Item that is part of a Ship (see 4.2.88) in terms of its applicability to a Ship. The ship_context need not be specified for a particular Item. See 4.3.49 for the application assertion.

4.2.53 Item_relationship

An Item_relationship is a type of both Definable_object (see 4.2.22) and Versionable_object (see 4.2.120) that defines the association of two Item (see 4.2.52) objects. The related Item objects share a common function or activity, or are dependent on each other. Each Item_relationship is a Moulded_-form_relationship (see 4.2.66).

The data associated with an Item_relationship are the following:

— external_item_1;

— external_item_2;

— item_1;

— item_2;

4.2.53.1 external_item_1

The external_item_1 specifies the relating Item of the Item_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 4.3.50 for the application assertion.

4.2.53.2 external_item_2

The external_item_2 specifies the related Item of the Item_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 4.3.50 for the application assertion.

4.2.53.3 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 4.3.51 for the application assertion.

4.2.53.4 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 4.3.51 for the application assertion.

4.2.54 Item_structure

An Item_structure is a type of both Definable_object (see 4.2.22) and Versionable_object (see 4.2.120) that is a collection of Item (see 4.2.52) objects possibly related by Item_relationship (see 4.2.53) objects. An Item_structure forms a graph without any restrictions regarding the number of entries, the connectivity nor the cyclicity. Each Item_structure is a Ship_moulded_form (see 4.2.93).

The data associated with an Item_structure are the following:

— external_items;

— external_relationships;

— items;

— relationships.

4.2.54.1 external_items

The external_items specifies the Item objects belonging externally to an Item_structure. See 4.3.52 for the application assertion.

4.2.54.2 external_relationships

The external_relationships specifies the relationships (external) between the Item (see 4.2.52) objects belonging to an Item_structure. See 4.3.52 for the application assertion.

4.2.54.3 items

The items specifies the Item objects belonging locally to an Item_structure. See 4.3.53 for the application assertion.

4.2.54.4 relationships

The relationships specifies the relationships (local) between the Item (see 4.2.52) objects belonging to an Item_structure. See 4.3.54 for the application assertion.

4.2.55 Knot

A Knot is a vertex point with additional information for a ship specific wireframe representation. The Knot is the corner point of a cell in a wireframe with information about the intersecting ship curves at this point and information about the tangent at this point.

The data associated with a Knot are the following:

— intersecting_ship_curves;

— tangent_information.

4.2.55.1 intersecting_ship_curves

The intersecting_ship_curves specifies the reference to the ship curves, which are intersecting at the Knot. See 4.3.55 for the application asertion.

4.2.55.2 tangent_information

The tangent_information specifies the direction and the magnitude of the tangent at the Knot. The tangent_information need not be specified for a particular Knot.

NOTE This information can be useful for systems which do not have sufficient curve geometry information.

4.2.56 Local_co_ordinate_system

A Local_co_ordinate_system is a type of Definition (see 4.2.23) 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 a Global_axis_placement (see 4.2.41) or another Local_co_ordinate_system that is a member in the same hierarchy. Each Local_co_ordinate_system may be a Local_co_ordinate_system_with_position_-reference (see 4.2.57).

EXAMPLE Figure 13 illustrates a local coordinate system.

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.

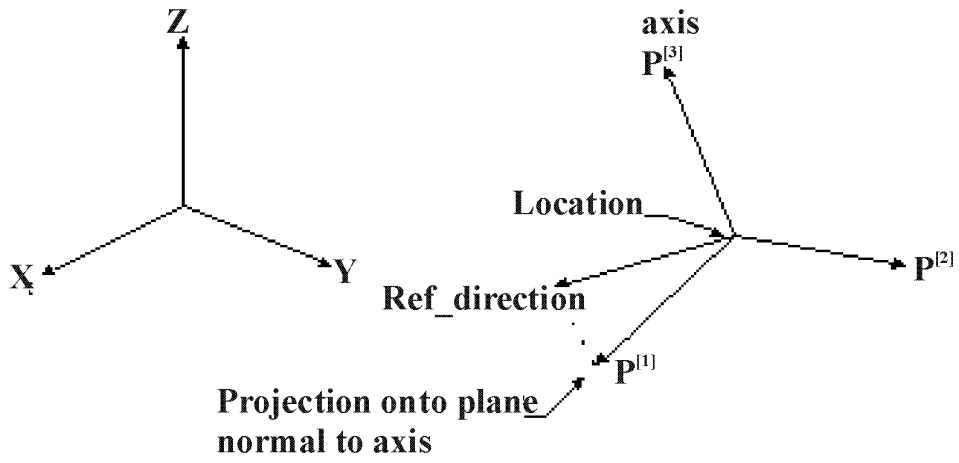


Figure 13 — Local coordinate system

The data associated with a Local_co_ordinate_system are the following:

— parent.

4.2.56.1 parent

The parent specifies the underlying coordinate system, which serves as definition space for the current coordinate system. See 4.3.56 and 4.3.57 for the application assertions.

4.2.57 Local_co_ordinate_system_with_position_reference

A Local_co_ordinate_system_with_position_reference is a type of Local_co_ordinate_system (see 4.2.56) that directly refers to the unique Global_axis_placement (see 4.2.41) as its parent. Its placement is defined by references to longitudinal, vertical or transversal frames, possibly using an additional offset value (a distance). Alternatively absolute coordinates may be specified. Also combinations of coordinates

and references are also valid. A Local_co_ordinate_system_with_position_reference shall not specify rotations as transformations to the global system as a result, that is 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.

4.2.57.1 longitudinal_ref

The longitudinal_ref specifies a Longitudinal_position (see 4.2.58), 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 4.3.58 for the application assertion.

4.2.57.2 transversal_ref

The transversal_ref specifies a Transversal_position (see 4.2.112), 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 4.3.58 for the application assertion.

4.2.57.3 vertical_ref

The vertical_ref specifies a Vertical_position (see 4.2.122), 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 4.3.58 for the application assertion.

4.2.58 Longitudinal_position

A Longitudinal_position is a type of Spacing_position (see 4.2.100) that is located on the global X-axis.

4.2.59 Longitudinal_table

A Longitudinal_table is a type of Spacing_table (see 4.2.102) 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 4.2.38) or a Station_table (see 4.2.107).

The data associated with a Longitudinal_table are the following:

— spacing_table_representations.

4.2.59.1 spacing_table_representations

The spacing_table_representations specifies the longitudinal positions that make up the Longitudinal_table. See 4.3.59 for the application assertion.

4.2.60 Midship_tumble

A Midship_tumble is a specific design parameter used for the definition of the midship section.

EXAMPLE Figure 14 illustrates midship tumble.

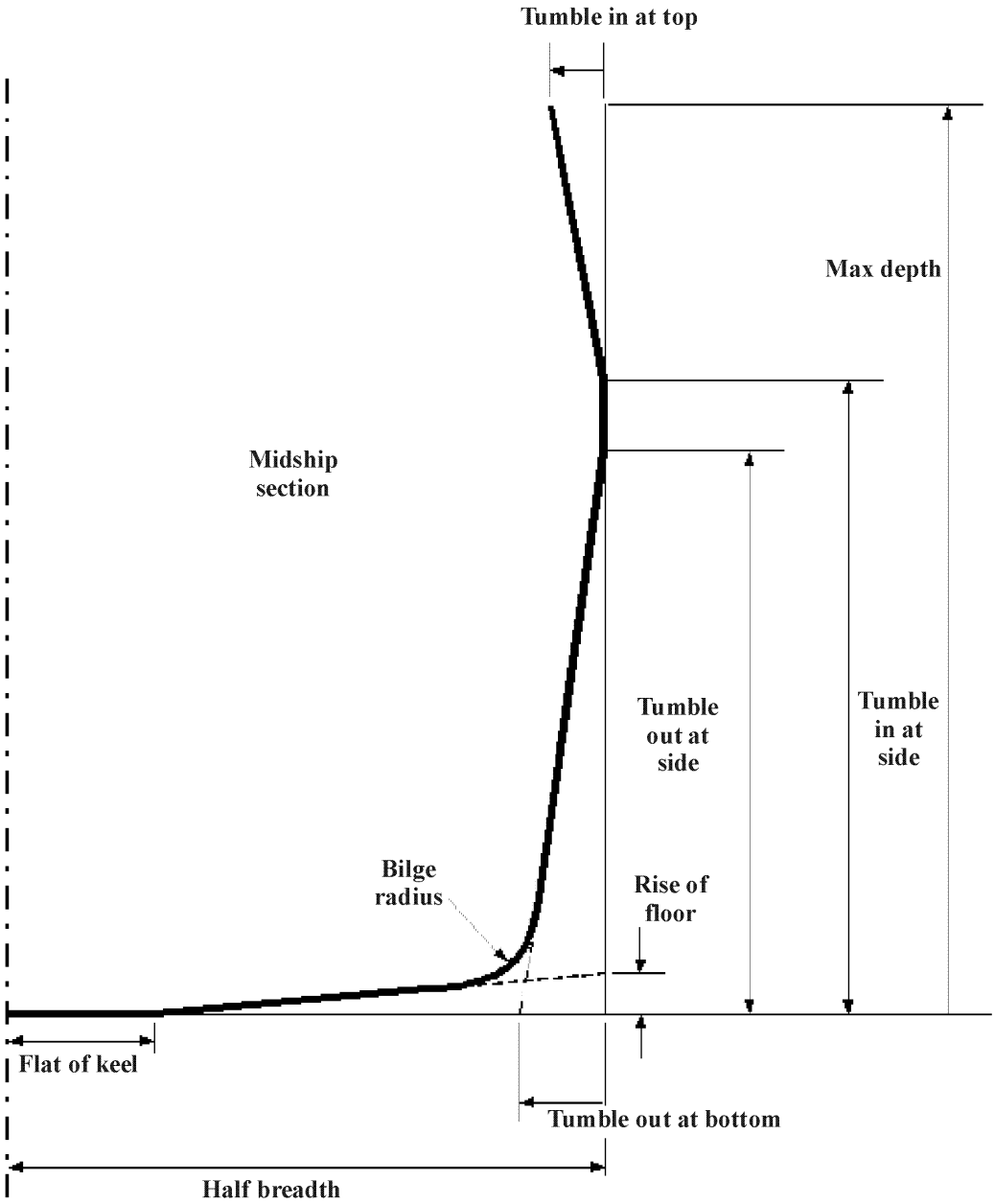


Figure 14 — Midship tumble

The data associated with a Midship_tumble are the following:

— tumble_in_at_side;

— tumble_in_at_top;

— tumble_out_at_bottom;

— tumble_out_at_side.

4.2.60.1 tumble_in_at_side

The tumble_in_at_side specifies the height of the upper extent of the midship half width above the projection of the flat of keel.

4.2.60.2 tumble_in_at_top

The tumble_in_at_top specifies the distance from the top of the midship section at the maximum depth above the base to the maximum half beam.

4.2.60.3 tumble_out_at_bottom

The tumble_out_at_bottom specifies the distance outboard on the flat of keel extended from the projection of the lower side of the midship section to the perpendicular through the maximum half beam.

4.2.60.4 tumble_out_at_side

The tumble_out_at_side specifies the height of the lower extent of the midship half width above the projection of the flat of keel.

4.2.61 Moulded_form

A Moulded_form is a type of Item (see 4.2.52) that is part of the Ship_moulded_form (see 4.2.93). A Moulded_form defines the geometric shape of a constituent part of the ship. Certain types of Moulded_-form objects may also be defined by a set of dimensions.

EXAMPLE Figure 15 illustrates types of moulded forms.

NOTE A Moulded_form does not include the thickness, or any other information, of the material from which it is constructed. There is a difference between a Moulded_form and a Ship_surface (see 4.2.97). The Moulded_form is a physical item, which is part of the Ship_moulded_form. A Ship_surface is a representation of a Moulded_form. One or more Ship_surface objects represent a Moulded_form.

4.2.62 Moulded_form_boundary_relationship

A Moulded_form_boundary_relationship is a type of Moulded_form_relationship (see 4.2.66) that defines a relationship between two moulded forms that specifies one of the borders of the first moulded

form as its intersection with the second moulded form.

The data associated with a Moulded_form_boundary_relationship are the following:

— item_1;

— item_2.

4.2.62.1 item_1

The item_1 specifies the Moulded_form (see 4.2.61) for which the boundary is being defined. The item_1 need not be specified for a particular Moulded_form_boundary_relationship. See 4.3.60 for application assertion.

NOTE If item_1 is not specified,external_item_1 attribute for Item_relationship (see 4.2.53.1) must be specified.

4.2.62.2 item_2

The item_2 specifies the Moulded_form (see 4.2.61) that defines one of the boundaries of the Moulded_form defined by item_1. The item_2 need not be specified for a particular Moulded_form_boundary_relationship. See 4.3.61 for application assertion.

NOTE If item_2 is not specified,external_item_2 attribute for Item_relationship (see 4.2.53.2) must be specified.

4.2.63 Moulded_form_characteristics_definition

A Moulded_form_characteristics_definition is a type of Definition (see 4.2.23) that specifies design parameters for the different types of Moulded_form (see 4.2.61) objects. This information contains dimensions and ratios important to the simple characterization of the hull, bow, propeller, rudder, appendage, thruster and bottom details.

A Moulded_form_characteristics_definition may be a Hull_moulded_form_design_parameter (see 4.2.44), a Deck_moulded_form_design_parameter (see 4.2.21), a Bottom_moulded_form_design_-parameter (see 4.2.6), a Bulb_moulded_form_design_parameter (see 4.2.7), a Propeller_moulded_form_-design_parameter (see 4.2.80), a Rudder_moulded_form_design_parameter (see 4.2.86), a Thruster_-moulded_form_design_parameter (see 4.2.111), or an Appendage_moulded_form_design_parameter (see 4.2.3).

NOTE If the moulded form characteristics data differ from the information available from any geometric representation then the geometric data have precedence. The displacement and wetted outer surface are based on the moulded form. From a naval architectural viewpoint the same parameter based on the extreme form, which includes the thickness, might be more practical.

Figure 15 — Ship hull with bulbous bow and thruster tunnels

The data associated with a Moulded_form_characteristics_definition are the following:

— defined_for;

— moulded_form_displacement;

— moulded_form_outer_surface.

4.2.63.1 defined_for

The defined_for specifies the Moulded_form (see 4.2.61) for which the Moulded_form_characteristics_-definition is defined for. See 4.3.62 for the application assertion.

4.2.63.2 moulded_form_displacement

The moulded_form_displacement specifies the wetted displacement of the individual Moulded_form (see 4.2.61).

4.2.63.3 moulded_form_outer_surface

The moulded_form_outer_surface specifies the wetted outer surface of the individual Moulded_form (see 4.2.61).

4.2.64 Moulded_form_design_definition

A Moulded_form_design_definition is a type of Design_definition (see 4.2.25) that specifies the definition of a Moulded_form (see 4.2.61).

The data associated with a Moulded_form_design_definition are the following:

— borders;

— defined_for;

— moulded_surface;

— representations;

— status.

4.2.64.1 borders

The borders specifies the borders of the moulded form specified by explicit geometry or by way of relationships to other Moulded_form (see 4.2.61) objects. The order of the borders is to be significant but there is no rule that requires end point of border[n] = start point of border[n+1]. Furthermore two borders may intersect or it might be necessary to (straight) elongate one or both borders in order to make them intersecting. The borders need not be specified for a particular Moulded_form_design_definition. See 4.3.66, 4.3.67, and 4.3.70 for the application assertions.

4.2.64.2 defined_for

The defined_for specifies the Moulded_form (see 4.2.61) objects which the Moulded_form_design_-definition is defining. See 4.3.65 for the application assertion.

4.2.64.3 moulded_surface

The moulded_surface specifies the underlying surface geometry for definition of the moulded form. The geometry may be defined by a single surface, by a wire shell consisting of a mesh of curves, by a face based surface model, or by a table of offset points. The moulded_surface need not be specified for a particular Moulded_form_design_definition. See 4.3.67, 4.3.69, 4.3.63, and 4.3.64 for the application assertions.

4.2.64.4 representations

The representations specifies the shape definition for the moulded form. The shape definition may be either an Edge_based_wireframe_shape (see 4.2.31) or a Non_manifold_surface_shape (see 4.2.72). See 4.3.63 and 4.3.64 for the application assertions.

4.2.64.5 status

The status specifies whether the moulded_surface and representations data associated with the Moulded_form_design_definition contain a complete or partial geometric definition. The status need not be specified for a particular Moulded_form_design_definition.

The status shall be one of the following:

— complete;

— partial.

4.2.64.5.1 complete

The geometric definition given for the Moulded_form_design_definition is complete.

4.2.64.5.2 partial

The geometric definition given for the Moulded_form_design_definition is only a partial definition.

4.2.65 Moulded_form_functional_definition

The Moulded_form_functional_definition is a type of Functional_definition (see 4.2.39) that defines the function of a part of a ship, which is defined by the underlying Moulded_form (see 4.2.61).

The data associated with a Moulded_form_functional_definition are the following:

— defined_for;

— the_function.

4.2.65.1 defined_for

The defined_for specifies the Moulded_form (see 4.2.61) for which the Moulded_form_functional_-

definition is defined. There may be more than one Moulded_form for a Moulded_form_functional_definition. See 4.3.71 for the application assertion.

4.2.65.2 the_function

The the_function specifies the functionality of the Moulded_form (see 4.2.61) for which the Moulded_-form_functional_definition is defined.

The the_function shall be one of the following:

— appendage;

— bulbous_bow;

— bulbous_stern;

— deck;

— double_bottom;

— double_ship_hull;

— frame;

— grating;

— horizontal_girder;

— keel;

— longitudinal_bulkhead;

— longitudinal_girder;

— non_structural_bulkhead;

— pressure_hull;

— propeller;

— rudder;

— ship_hull;

— superstructure;

— thruster;

— transom;

— transverse_bulkhead;

— user_defined.

4.2.65.2.1 appendage

A part that is added to the ship hull outer surface to improve the properties of the ship.

EXAMPLE Figure 4 , Figure 5 , Figure 6 , and Figure 7 illustrate types of appendage. Appendages can be bilge keel, shaft strut, shaft bossings, fins and others.

4.2.65.2.2 bulbous_bow

The underwater shape of the bow designed to smooth out bow wave and to reduce the resistance of the ship.

EXAMPLE Figure 16 illustrates a ship hull with bulbous bow.

Figure 16 — Ship hull with bulbous bow

4.2.65.2.3 bulbous_stern

The underwater shape of the stern designed to improve the flow to the propeller and to improve the efficiency of the propeller.

4.2.65.2.4 deck

A horizontal dividing structure of the ship hull which is above the bottom or double bottom.

EXAMPLE Figure 17 illustrates a deck and bulkheads.

4.2.65.2.5 double_bottom

The internal bottom of a ship hull in the case of double plating on the ship hull.

EXAMPLE Figure 18 and Figure 19 illustrate a double bottom.

Figure 17 — Bulkheads and decks

Figure 18 — Double bottom

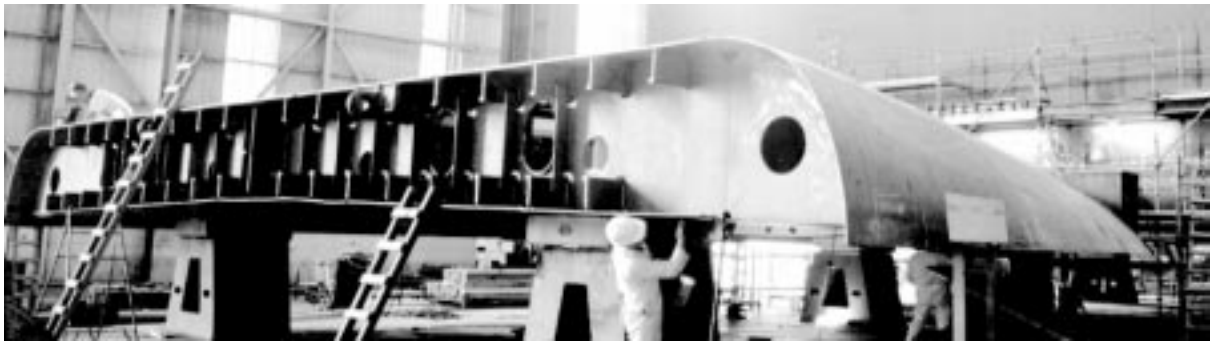


Figure 19 — Outer and inner bottom with profiles

4.2.65.2.6 double_ship_hull

The internal sides of the ship hull in case of double plating on the ship hull.

EXAMPLE Figure 20 illustrates a double ship hull.

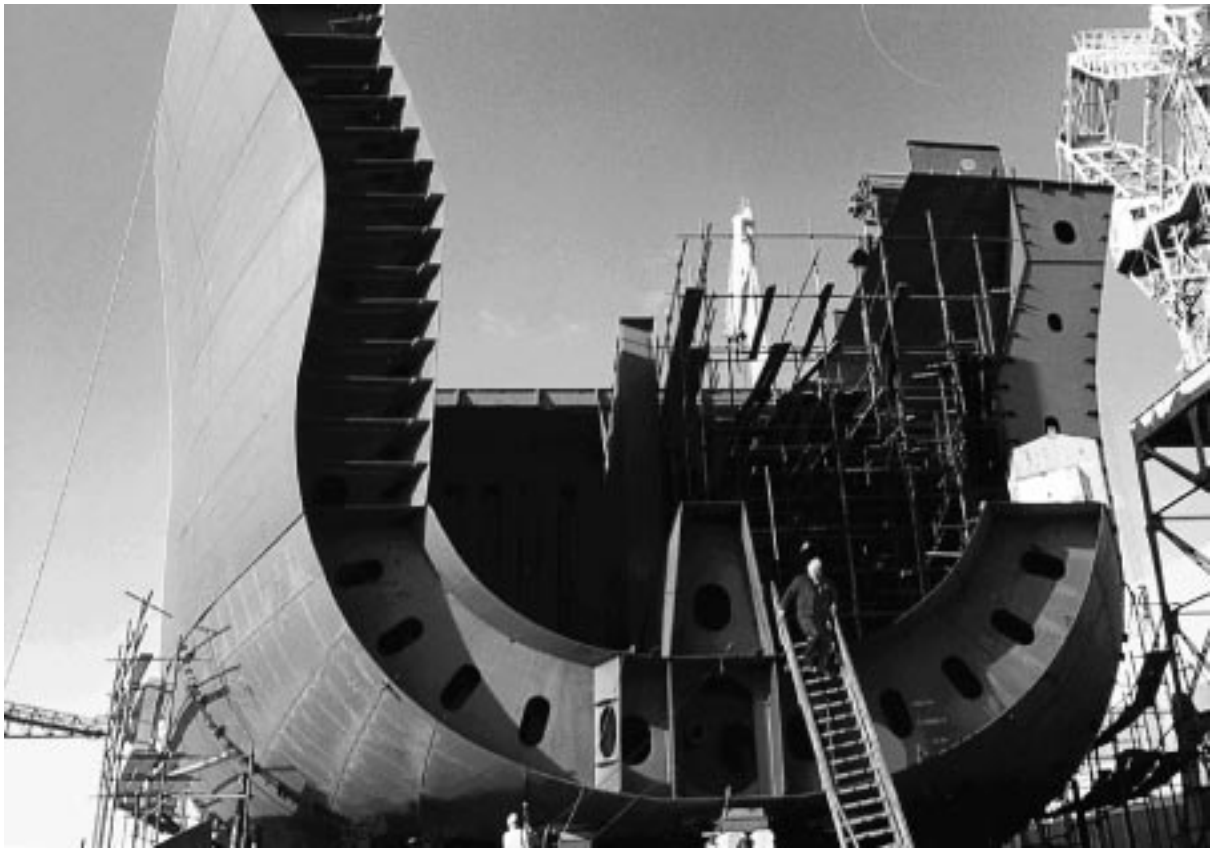


Figure 20 — Double ship hull

4.2.65.2.7 frame

A vertical structural element arranged transversely in the ship that contributes to the ship's strength.

EXAMPLE Figure 21 illustrates frames and decks.

4.2.65.2.8 grating

A decking within a compartment which is not a part of the main structural systems but which allows access by the ship's crew.

4.2.65.2.9 horizontal_girder

A longitudinal primary structural element that lies in the horizontal plane of the ship.

EXAMPLE Figure 22 illustrates horizontal girders.

4.2.65.2.10 keel

The uninterrupted plating which extends over the full length of the bottom from fore to aft of the ship.

NOTE For sailing yachts the keel is specially shaped and includes additional weight to improve weight stability.

4.2.65.2.11 longitudinal_bulkhead

A vertical structural element arranged longitudinally in the ship that forms a compartment boundary and contributes significantly to the ship's sub-division and strength.

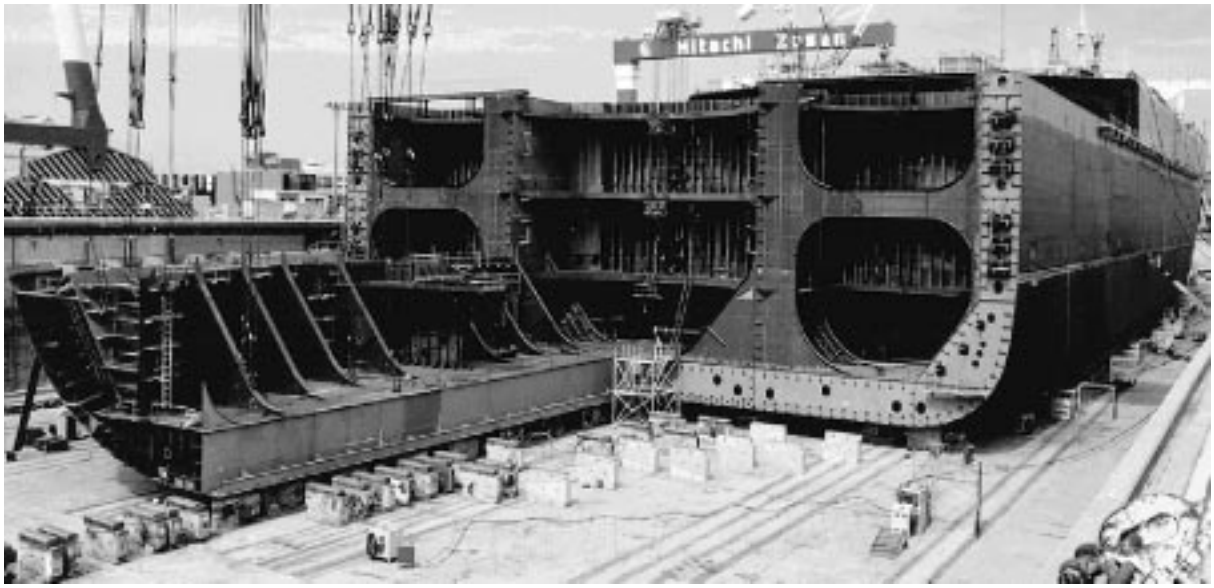


Figure 21 — Frames and decks

4.2.65.2.12 longitudinal_girder

A longitudinal primary structural element that lies in the vertical plane of the ship.

EXAMPLE Figure 22 illustrates longitudinal girders.

4.2.65.2.13 non_structural_bulkhead

A bulkhead used for division of compartmentation rather than as a load-bearing structure.

4.2.65.2.14 pressure_hull

A special construction of the hull for underwater operation. The hull has to cope with underwater pressure and is used for submarines or any industrial underwater equipment.

4.2.65.2.15 propeller

A structure which converts the power of any engine into a force in the water for propelling or maneuvering a ship.

EXAMPLE Figure 24 illustrates a propeller.

4.2.65.2.16 rudder

The main maneuvering instrument used for directional control of the ship.

EXAMPLE Figure 23 illustrates a rudder.

4.2.65.2.17 ship_hull

The main body of a ship designed for carrying goods or passengers, or a body for stability and additional buoyancy.

EXAMPLE Figure 20 illustrates a ship hull.

4.2.65.2.18 superstructure

A decked structure that is above the upper deck.

EXAMPLE Figure 26 illustrates a superstructure.

4.2.65.2.19 thruster

A transverse underwater tunnel, which runs across and through the ship hull at the bow or stern of the ship. A propeller usually operates in the tunnel to enable better maneuvering of the ship.

EXAMPLE Figure 25 illustrates a thruster.

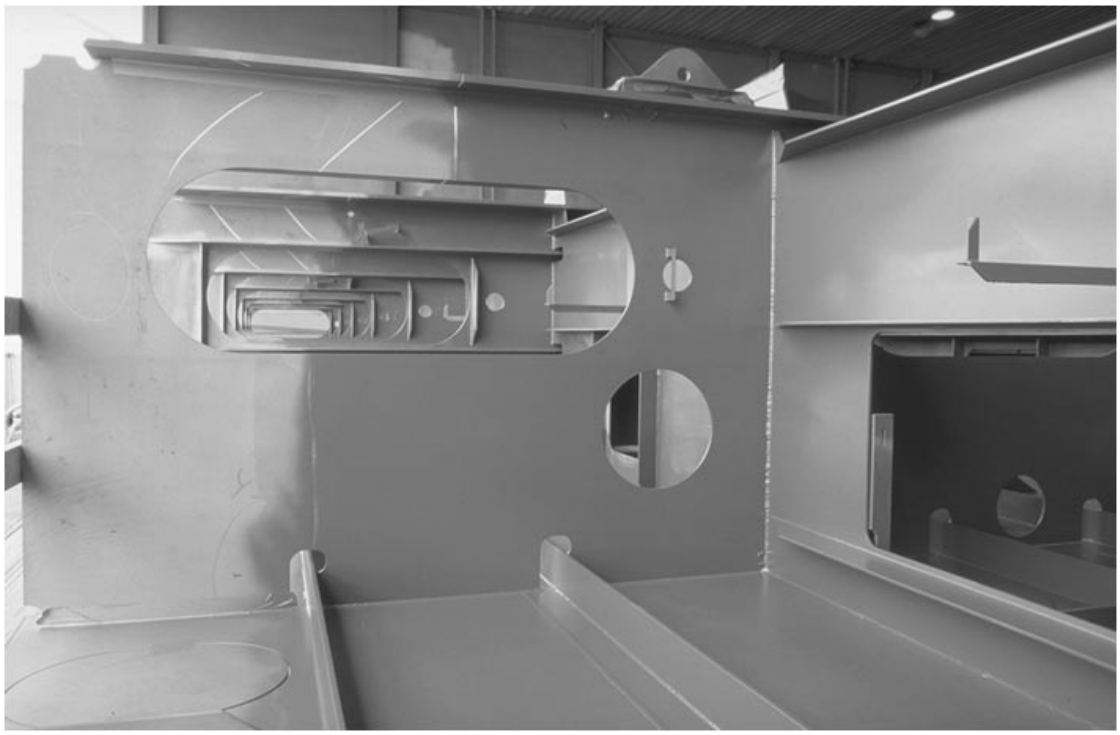


Figure 22 — Girders

4.2.65.2.20 transom

The after part of a squaterned ship hull.

EXAMPLE Figure 27 illustrates a transom.

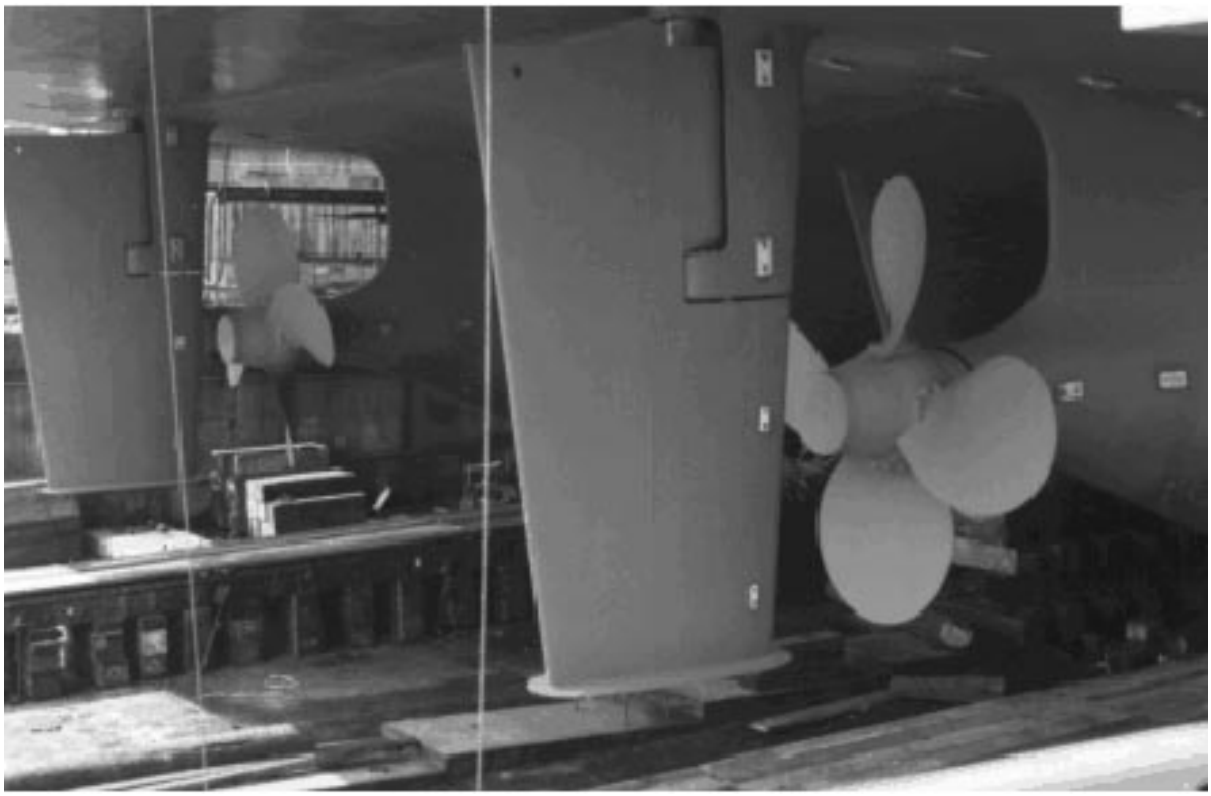


Figure 23 — Rudder and propeller

Figure 24 — Screw propeller model

Figure 25 — Thruster

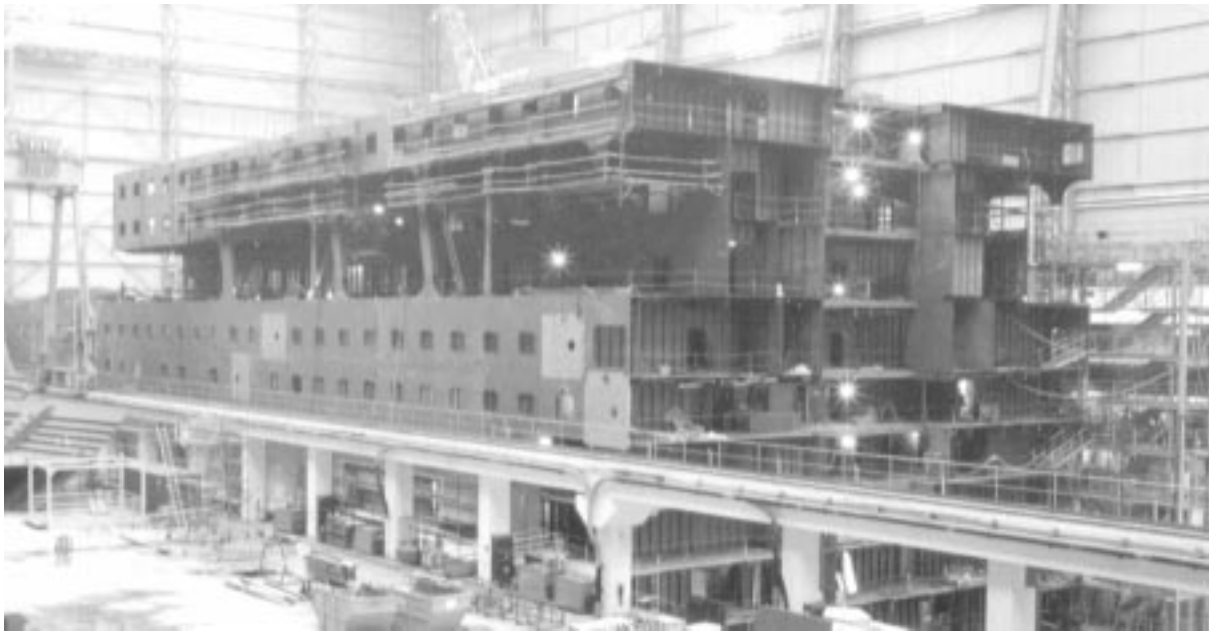


Figure 26 — Decks and superstructure

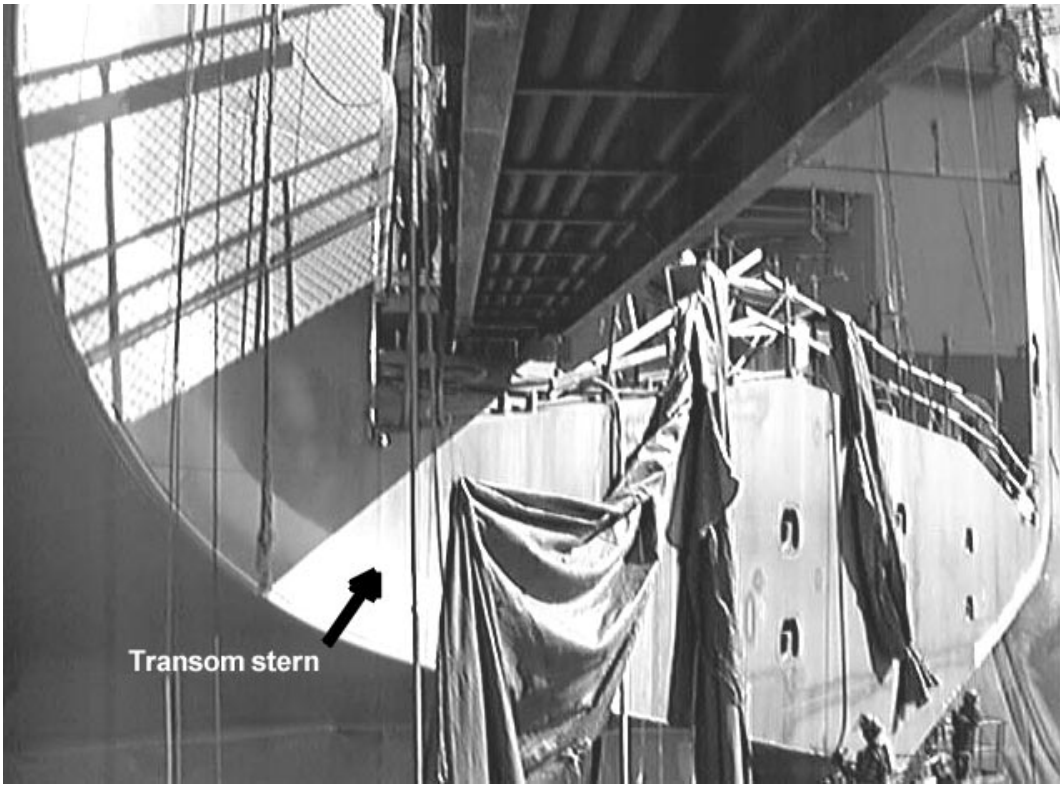


Figure 27 — Transom stern

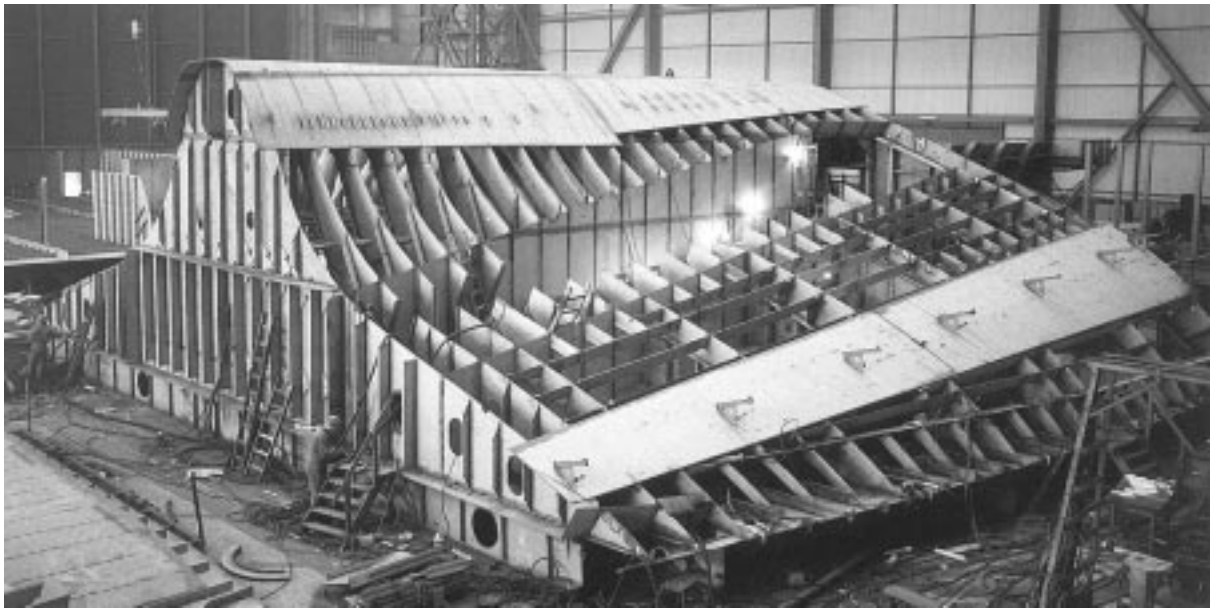


Figure 28 — Transverse bulkheads

4.2.65.2.21 transverse_bulkhead

A vertical primary structural element arranged transversely in the ship that forms a compartment boundary and contributes significantly to the ship's sub-division and strength.

EXAMPLE Figure 28 illustrates a transverse bulkhead.

4.2.65.2.22 user_defined

The function of the Moulded_form (see 4.2.61) is unknown or not covered by any of the above.

4.2.66 Moulded_form_relationship

A Moulded_form_relationship is a type of Item_relationship (see 4.2.53) that describes a relationship between two Moulded_form (see 4.2.61) objects. A collection of Moulded_form_relationship objects may describe a hierarchy of inter-related Moulded_form objects that together provide a description of the Ship_moulded_form (see 4.2.93).

EXAMPLE A Moulded_form_relationship could be the relationship between the Moulded_form of a ship hull and the related propeller.

The data associated with a Moulded_form_relationship are the following:

— item_1;

— item_2.

4.2.66.1 item_1

The item_1 specifies the Moulded_form (see 4.2.61) that forms the basis of the Moulded_form_relationship. The item_1 need not be specified for a particular Moulded_form_relationship. See 4.3.72 for the application assertion.

NOTE If item_1 is not specified, the external_item_1 attribute for Item_relationship (see 4.2.53.1) must be specified.

4.2.66.2 item_2

The item_2 specifies the Moulded_form (see 4.2.61) that is to be associated with item_1. The item_2 need not be specified for a particular Moulded_form_relationship. See 4.3.72 for the application assertion.

NOTE If item_2 is not specified, the external_item_2 attribute for Item_relationship (see 4.2.53.2) must be specified.

4.2.67 Moulded_form_representation_item

A Moulded_form_representation_item is the moulded form representation of some part of the ship. This representation uses either a grid of points, a grid of geometric curves, or surfaces of zero thickness to define the moulded form of the ship or part of the ship. A Moulded_form_representation_item may be a Ship_point (see 4.2.96), a Ship_curve (see 4.2.89) or a Ship_surface (see 4.2.97).

4.2.68 Moulded_form_representation_relationship

A Moulded_form_representation_relationship is the relationship between two Moulded_form_shape_-representation (see 4.2.69) objects.

The data associated with a Moulded_form_representation_relationship are the following:

— description;

— id;

— rep_1;

— rep_2.

4.2.68.1 description

The description specifies a user-defined description of the purpose of the relationship. A description need not be specified for a particular Moulded_form_representation_relationship.

4.2.68.2 id

The id specifies the context-specific identifier for the Moulded_form_representation_relationship.

4.2.68.3 rep_1

The rep_1 specifies the Moulded_form_shape_representation (see 4.2.69) that forms the basis of the Moulded_form_representation_relationship. See 4.3.73 for the application assertion.

4.2.68.4 rep_2

The rep_2 specifies the Moulded_form_shape_representation (see 4.2.69) that is to be associated with rep_1. See 4.3.73 for the application assertion.

4.2.69 Moulded_form_shape_representation

A Moulded_form_shape_representation is the shape of a Moulded_form (see 4.2.61) using geometric models, topological models, and geometric elements. The geometric constructs in a Moulded_form_-shape_representation may reference a Ship_point (see 4.2.96), a Ship_curve (see 4.2.89), or a Ship_-surface (see 4.2.97). Each Moulded_form_shape_representation is either an Offset_table_shape_-representation (see 4.2.74), a Wireframe_shape_representation (see 4.2.125) or a Surface_shape_-

representation (see 4.2.109).

NOTE A Moulded_form_shape_representation is abstract in concept and does not detail the geometry but provides references to the topology and geometry that it may possess.

The data associated with a Moulded_form_shape_representation are the following:

— moulded_form_representation_id;

— moulded_form_symmetry.

4.2.69.1 moulded_form_representation_id

The moulded_form_representation_id specifies the context-specific label for the Moulded_form_shape_-representation. The moulded_form_representation_id need not be specified for a particular Moulded_form_shape_representation.

4.2.69.2 moulded_form_symmetry

The moulded_form_symmetry specifies the symmetry for the Moulded_form_shape_representation. Planar and rotational symmetry are possible for a Moulded_form (see 4.2.61). The moulded_form_symmetry need not be specified for a particular Moulded_form_shape_representation. See 4.3.74 for the application assertion.

4.2.70 Named_unit

A Named_unit is a pre-defined unit object specified in ISO 10303-41.

4.2.71 Navy_ship

A Navy_ship is a type of Shiptype (see 4.2.98) that is a ship operating under military command.

The data associated with a Navy_ship are the following:

— has_type.

4.2.71.1 has_type

The has_type specifies the type of the Navy_ship.

The has_type shall be 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.

4.2.71.1.1 aircraft_carrier

A type of ship designed to deploy aircraft and 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(STOVAL), short take off but arrested recovery (STOBAR), or conventional take off (CTOL).

4.2.71.1.2 auxiliary_oiler

A type of ship designed to replenish ships at sea with liquids during world-wide operations, with vertical-replenishment (VERTREP) 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.

4.2.71.1.3 corvette

A type of ship that is designed as a small escort vessel to a task group.

NOTE Corvettes are fitted primarily to fulfil an anti-submarine warfare role.

4.2.71.1.4 cruiser

A type of ship that 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.

4.2.71.1.5 destroyer

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.

4.2.71.1.6 fleet_auxiliary_vessel

A type of ship that 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.

4.2.71.1.7 frigate

A type of ship that can be described as 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 modern day air threats.

NOTE Frigates may also provide command facilities and accommodation. Secondary capabilities include anti-surface warfare, naval gunfire support and anti-submarine warfare.

4.2.71.1.8 landing_platform_dock

A type of ship specifically designed to transport a large embarked military force and support equipment, across open oceans; support a flexible landing on hostile shores using onboard helicopters and landing craft, and co-ordinate the naval, air and land aspects of amphibious operations through command, control and communications facilities.

4.2.71.1.9 landing_platform_helicopter

A type of ship specifically 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.

4.2.71.1.10 mine_warfare_ship

A mine countermeasures ship (MCMV) 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, MCMV hulls are often constructed of low magnetic steel or laminated wood.

4.2.71.1.11 patrol_force_vessel

A ship that can vary in size according to the roles for which they have been designed.

NOTE Larger offshore patrol vessels (OPVs) are used for fire fighting, 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 OPVs have ice-strengthened hulls and a helicopter landing deck for operation as survey vessels in the Arctic or Antarctic regions. The size, maneuverability and other operational characteristics of smaller coastal patrol craft means that they are ideally suited for patrol, search and rescue duties in coastal areas, or ports, harbors and other restricted waters.

4.2.71.1.12 service_craft

A small ship that is designed to provide specific 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, hospital ships, etc.

4.2.71.1.13 submarine

A high-capability, ocean-going vessel designed to perform both anti-submarine and anti-surface warship tasks.

NOTE To perform these functions, generally a submarine carries heavyweight torpedoes, underwater-to-surface guided weapons or submarine-laid mines. Submarines are able to operate submerged in shallow waters, in open ocean, and being capable of operation in waters from the tropics to the arctic.

4.2.71.1.14 user_defined

The ship type is not one of the pre-defined types listed above.

NOTE Details may be found in the Shiptype (see 4.2.99) description attribute.

4.2.72 Non_manifold_surface_shape

A Non_manifold_surface_shape is a shape representation that conforms to ISO 10303-508.

4.2.73 Offset_point_table_model

An Offset_point_table_model is the shape of the Moulded_form (see 4.2.61) defined by a set of Ship_-point (see 4.2.96) objects. The Ship_point objects are listed in sections.

NOTE 2 Normally the sections in the Offset_point_table_model are 2d sections. In special circumstances there might be a need for using 3d sections. Then the offset_point_table_type should be user_defined_table and the sections of the offset table representing 3d curves can be of any 3d shape.

The data associated with an Offset_point_table_model are the following:

— offset_point_table_sections;

— offset_point_table_type.

4.2.73.1 offset_point_table_sections

The offset_point_table_sections specifies the list of sections that detail the separate sections of points in
the offset table. See 4.3.75 for the application assertion.

EXAMPLE A common illustration can be an offset table with waterline offsets. Then all points in a single section represent one waterline.

4.2.73.2 offset_point_table_type

The offset_point_table_type specifies the type of the offset table and the type of sections where the offset points are located.

The offset_point_table_type shall be one of the following:

— buttock_table;

— station_table;

— user_defined_table;

— waterline_table.

NOTE See 4.2.74.2.1 - 4.2.74.3.4 for the definition of each allowable value for offset_point_table_type.

4.2.73.2.1 buttock_table

an Offset_point_table_model that specifies that the offset points in the offset table are points located on buttocks in the XZ plane.

4.2.73.2.2 station_table

an Offset_point_table_model that specifies that the offset points in the offset table are points located on stations in the YZ plane.

4.2.73.2.3 user_defined_table

an Offset_point_table_model that the section for the offset points is not specified.

4.2.73.2.4 waterline_table

an Offset_point_table_model that specifies that the offset points in the offset table are points located on waterlines in the XY plane.

4.2.74 Offset_table_shape_representation

An Offset_table_shape_representation is a type of Moulded_form_shape_representation (see 4.2.69) which describes the shape of the Moulded_form (see 4.2.61) using offset points.

The data associated with an Offset_table_shape_representation are the following:

— items.

4.2.74.1 items

The items specifies the set of Offset_point_table_model (see 4.2.73) objects that build the Offset_table_-shape_representation. See 4.3.76 for the application assertion.

4.2.75 Owner_designation

An Owner_designation is a type of General_characteristics_definition (see 4.2.40) 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.

4.2.75.1 local_units

The local_units specifies the units that are used by the definition and differ from the units globally defined for the ship. Local_units may be either a Derived_unit (see 4.2.24) or a Named_unit (see 4.2.70). Local_units need not be specified for a particular Owner_designation. There may be more than one local_units for an Owner_designation. See 4.3.77 and 4.3.78 for application assertions.

4.2.75.2 managing_company

The managing_company specifies the organization that is responsible for managing and operating the ship.

4.2.75.3 ordering_company

The ordering_company specifies the organization that ordered the ship at a shipyard.

4.2.75.4 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.

4.2.75.5 owning_company

The owning_company specifies the organization that legally owns the ship.

4.2.76 Planar_symmetry

A Planar_symmetry is a type of Symmetry (see 4.2.110) that provides the planar symmetry information for any part of the ship. The Planar_symmetry details the plane of symmetry for this part of the ship.

The data associated with a Planar_symmetry are the following:

— the_symmetry_plane.

4.2.76.1 the_symmetry_plane

The the_symmetry_plane specifies the plane of symmetry in the defining coordinate system of this part. If only one half of the part is defined by the geometry, then this should be the geometry describing the part to which the normal axis is pointing.

4.2.77 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.

4.2.77.1 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.

4.2.78 Principal_characteristics

A Principal_characteristics is a type of General_characterisics_definition (see 4.2.40) that specifies the main shape parameters of the hull moulded form. Principal_characteristics also include data that is required in subsequent iterations of the hull development process when one is considering hydrostatics.

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.

4.2.78.1 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 should be defined only for mono hull ships. The block_coefficient need not be specified for a particular Principal_-characteristics.

4.2.78.2 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.

4.2.78.3 design_draught

The design_draught specifies the draught to which the ship has been designed to operate.

4.2.78.4 length_between_perpendiculars

The length_between_perpendiculars specifies the length measured from the after perpendicular to the forward perpendicular of the ship.

4.2.78.5 max_draught_at_ap

The max_draught_at_ap specifies the maximum possible draught at the after perpendicular during the operation of the ship. The max_draught_at_ap is used for hull cross section approval for ice class notation.

4.2.78.6 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 is used for hull cross section approval for ice class notation.

4.2.78.7 min_draught_at_ap

The min_draught_at_AP specifies the minimum possible draught at the after perpendicular during the operation of the ship. The min_draught_at_AP is used for hull cross section approval for ice class notation.

4.2.78.8 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 is used for hull cross section approval for ice class notation.

4.2.78.9 moulded_breadth

The moulded_breadth specifies the maximum breath of the ship amidships and at the design_draught.

4.2.78.10 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.

4.2.79 Propeller_location

A Propeller_location is the placement of the propeller relative to the ship hull

EXAMPLE Figure 29 illustrates propeller location and Figure 31 illustrates a propeller relative to an axis.

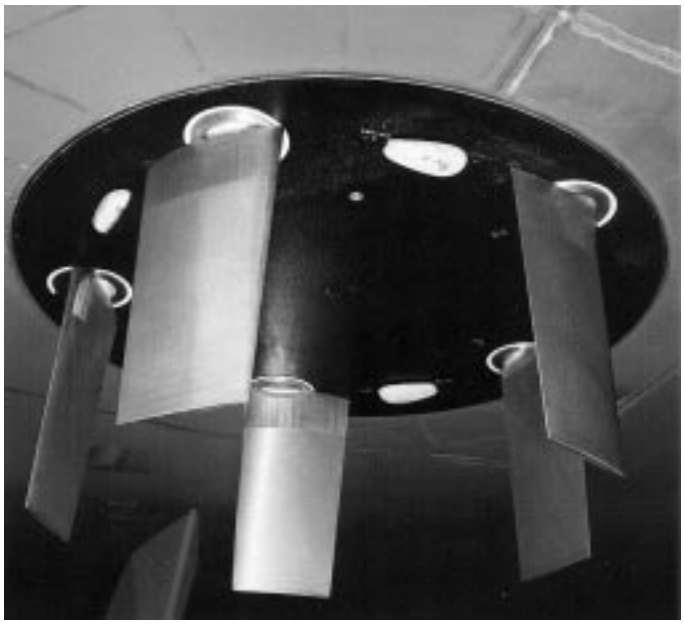


Figure 29 — Vertical axis propeller

Figure 30 — Vertical axis propeller technology

The data associated with a Propeller_location are the following:

— propeller_location;

— shaft_line_inclination_x;

— shaft_line_inclination_y;

— shaft_line_location.

4.2.79.1 propeller_location

The propeller_location specifies the placement of the centreline of the propeller shaft at the point where the propeller shaft is attached to the propeller. The X coordinate is measured from the after perpendicular, the Y coordinate is measured from the centreline and the Z coordinate is measured from the baseline. See 4.3.79 for the application assertion.

4.2.79.2 shaft_line_inclination_x

The shaft_line_inclination_x specifies the angle between the propeller shaft and the X-axis of the ship hull.

4.2.79.3 shaft_line_inclination_y

The shaft_line_inclination_y specifies the angle between the propeller shaft and the Y-axis of the ship hull.

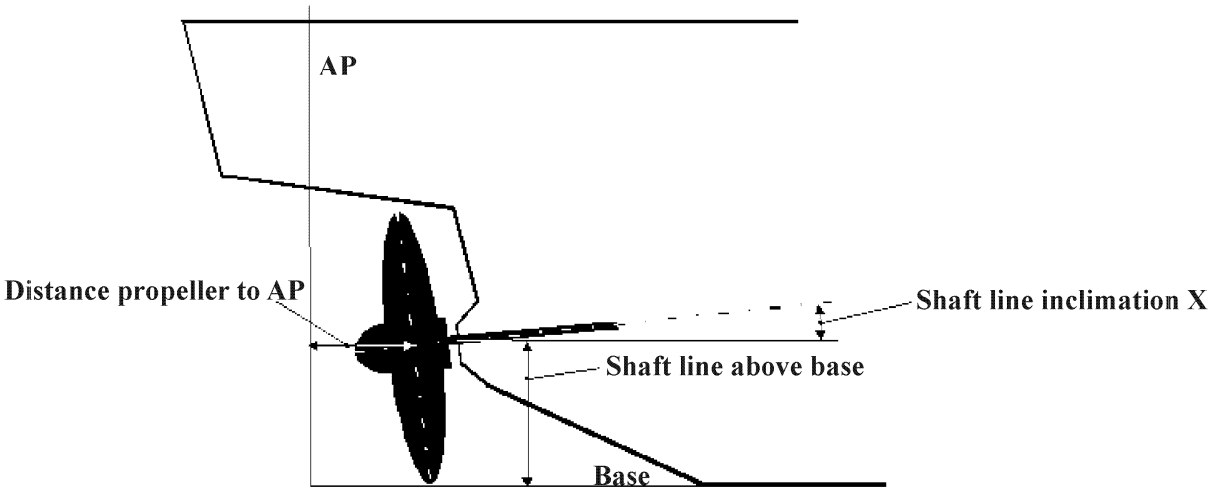


Figure 31 — Propeller location

4.2.79.4 shaft_line_location

The shaft_line_location specifies the placement of the centreline of the propeller shaft at the point where the propeller shaft comes out of the ship hull. The X coordinate is measured from the after perpendicular, the Y coordinate is measured from the centreline and the Z coordinate is measured from the baseline.

4.2.80 Propeller_moulded_form_design_parameter

A Propeller_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.63) that contains dimensions and ratios of the propeller.

EXAMPLE Figure 32 illustrates propeller dimensions.

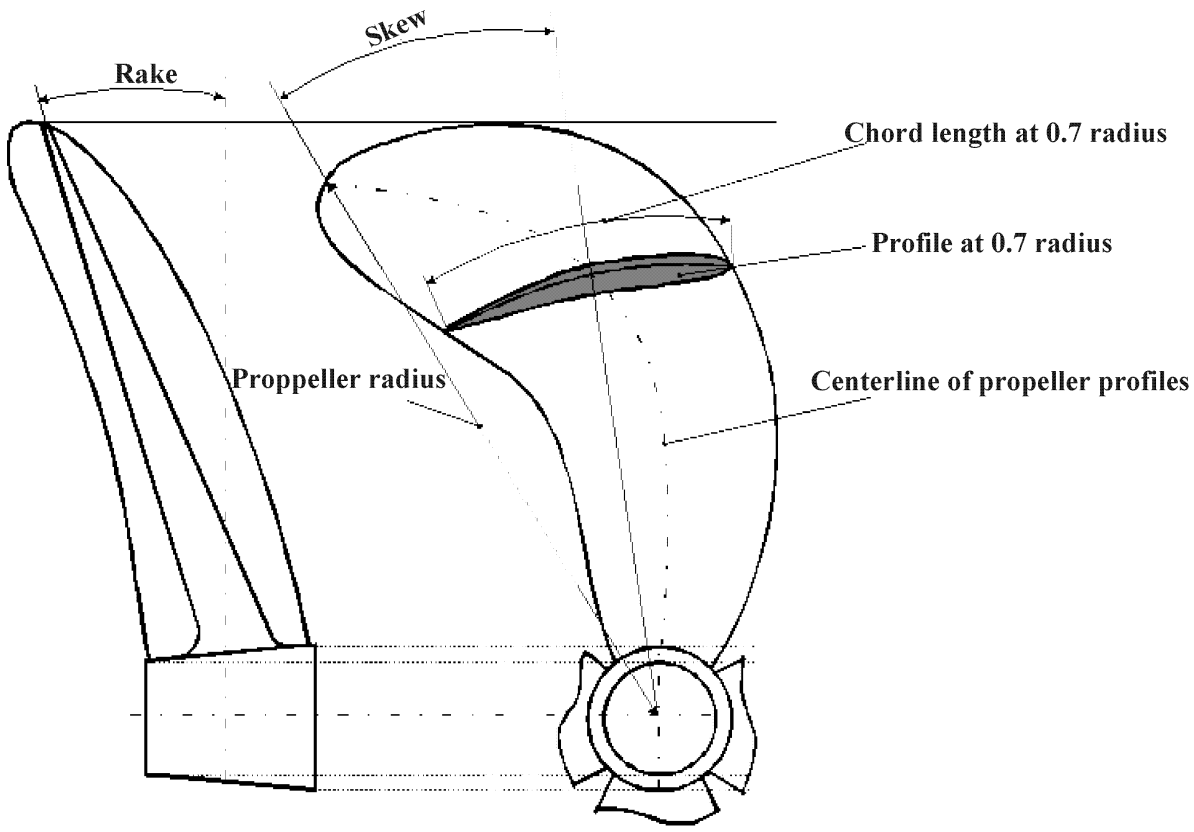


Figure 32 — Propeller dimensions

The data associated with a Propeller_moulded_form_design_parameter are the following:

— blade_mean_height;

— chord_length_at_0_7_radius;

— design_sense_of_rotation;

— expanded_area_ratio;

— hub_diameter_ratio;

— location_of_the_propeller_at_the_ship_hull;

— nominal_design_pitch_ratio;

— number_of_propeller_blades;

— propeller_diameter;

— rake;

— skew;

— thickness_at_0_7_radius;

— type_of_propeller_blades;

— type_of_propulsion.

4.2.80.1 blade_mean_height

The blade_mean_height specifies the mean height of blades for non-screw propellers. The blade_mean_-height need not be specified for a particular Propeller_moulded_form_design_parameter.

NOTE The mean height of the blade is the largest dimension of the blade and larger than the mean breadth.

4.2.80.2 chord_length_at_0_7_radius

The chord_length_at_0_7_radius specifies the chord length of the propeller blade section located in 0.7 of the propeller radius or the mean breadth of blades for non screw propellers.

4.2.80.3 design_sense_of_rotation

The design_sense_of_rotation specifies whether the design sense of rotation for the propeller is right-handed or left-handed.

The design_sense_of_rotation shall be one of the following:

— left;

— right.

NOTE See 4.2.80.3.1 - 4.2.80.3.2 for the definition of each allowable value for design_sense_of_rotation.

4.2.80.3.1 left

The propeller rotates in an anti-clockwise direction.

4.2.80.3.2 right

The propeller rotates in a clockwise direction.

4.2.80.4 expanded_area_ratio

The expanded_area_ratio specifies a dimension-less parameter, which is the result of the division of the developed area of the propeller by the area of the circle described by the propeller diameter. The developed propeller area is the total area of the blades assuming no pitch, as if the blades were flattened into one plane.

4.2.80.5 hub_diameter_ratio

The hub_diameter_ratio specifies a dimension-less parameter, which is the result of the division of the diameter of the propeller hub by the propeller diameter.

4.2.80.6 location_of_the_propeller_at_the_ship_hull

The location_of_the_propeller_at_the_ship_hull specifies the placement of the propeller relative to the ship hull. The location_of_the_propeller_at_the_ship_hull need not be specified for a particular Propeller_moulded_form_design_parameter. See 4.3.80 for the application assertion.

4.2.80.7 nominal_design_pitch_ratio

The nominal_design_pitch_ratio specifies a dimension-less parameter, which is the result of the division of the mean design pitch by the propeller diameter.

4.2.80.8 number_of_propeller_blades

The number_of_propeller_blades specifies the quantity of blades on the propeller.

4.2.80.9 propeller_diameter

The propeller_diameter specifies the diameter of the largest circle that is prescribed by the rotation of the propeller blades.

4.2.80.10 rake

The rake specifies the angle between the propeller plane and the generator line.

4.2.80.11 skew

The skew specifies the angle between a straight line through the centre of the propeller and the centre of the blade profile at 0.7 propeller radius and a second straight line through the centre of the propeller and the centre of the blade profile at 1 propeller radius.

4.2.80.12 thickness_at_0_7_radius

The thickness_at_0_7_radius specifies the profile thickness of the propeller blade section located in 0.7 of the propeller radius or the maximum thickness of blades for non screw propellers.

4.2.80.13 type_of_propeller_blades

The type_of_propeller_blades specifies whether the type of the propeller blade is fixed, adjustable or controllable.

The type_of_propeller_blades shall be one of the following:

— adjustable_blade;

— controllable_blade;

— fixed_blade.

NOTE See 4.2.81.13.1 - 4.2.81.13.3 for definition of allowable values for type_of_propellor_blades.

4.2.80.13.1 adjustable_blade

The propeller blades can be adjusted manually in a dry dock to meet varying operating conditions.

4.2.80.13.2 controllable_blade

The propeller blades can be controlled hydraulically to vary the pitch and it is possible to vary thrust to the point of providing astern thrust without reversing the direction of rotation.

EXAMPLE Figure 33 illustrates a controllable propeller blade.

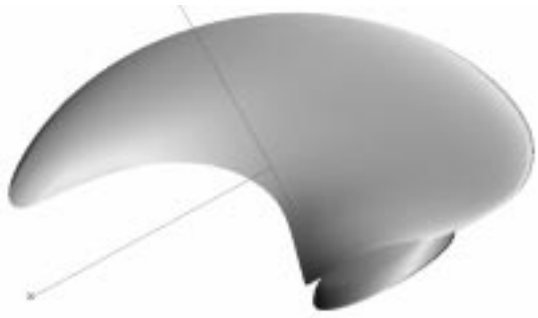


Figure 33 — Controllable propeller blade

4.2.80.13.3 fixed_blade

The propeller blades have a fixed pitch.

EXAMPLE Figure 34 illustrates a fixed blade propeller.

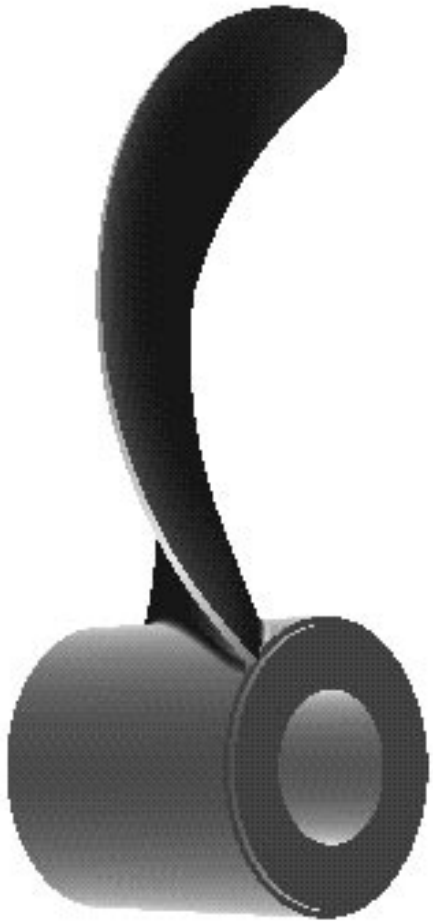


Figure 34 — Fixed propeller blade

4.2.80.14 type_of_propulsion

The type_of_propulsion specifies the individual type of propulsion system.

The type_of_propulsion shall be one of the following:

— ducted_propeller;

— paddle_wheel;

— screw_propeller;

— thruster;

— user_defined;

— vertical_axis_propeller;

— water_jet.

NOTE See 4.2.81.14.1 - 4.2.81.14.7 for definition of allowable values for type_of_propulsion.

4.2.80.14.1 ducted_propeller

The propeller works in a duct to increase efficiency.

EXAMPLE Figure 35 illustrates a ducted propeller.

4.2.80.14.2 paddle_wheel

The propulsion system consists of paddle blades rotating around a horizontal axis that is parallel to the Y axis of the ship.

NOTE This old fashioned type of propulsion is not used in modern ships.

4.2.80.14.3 screw_propeller

This is the most common propulsion system which operates without a duct or tunnel.

EXAMPLE Figure 36 illustrates a screw propeller.

Figure 35 — Ducted propeller

4.2.80.14.4 thruster

The propeller works in a tunnel located inside the ship hull, and is primarily for maneuvering purposes.

EXAMPLE Figure 25 illustrates a thruster propeller.

4.2.80.14.5 user_defined

The propulsion system type is user defined.

4.2.80.14.6 vertical_axis_propeller

the propeller has wings rotating around a vertical axis parallel to the Z axis of the ship. A special mechanism changes the angle of attack for each wing during rotation.

EXAMPLE Figure 29 and Figure 30 illustrate a vertical axis propeller.

4.2.80.14.7 water_jet

the propeller works in a tunnel inside the ship hull.

4.2.81 Regulation

A Regulation is a set of international and national regulations as well as other standards, which apply to the ship.

The data associated with a Regulation are the following:

— international_regulations;

— national_regulations;

— standards.

4.2.81.1 international_regulations

The international_regulations specifies all relevant international regulations, which apply to the ship. See 4.3.81 for the application assertion.

4.2.81.2 national_regulations

The national_regulations specifies all relevant national regulations, which apply to the ship. See 4.3.81 for the application assertion.

4.2.81.3 standards

The standards specifies all relevant specifications which apply to the ship. See 4.3.81 for the application assertion.

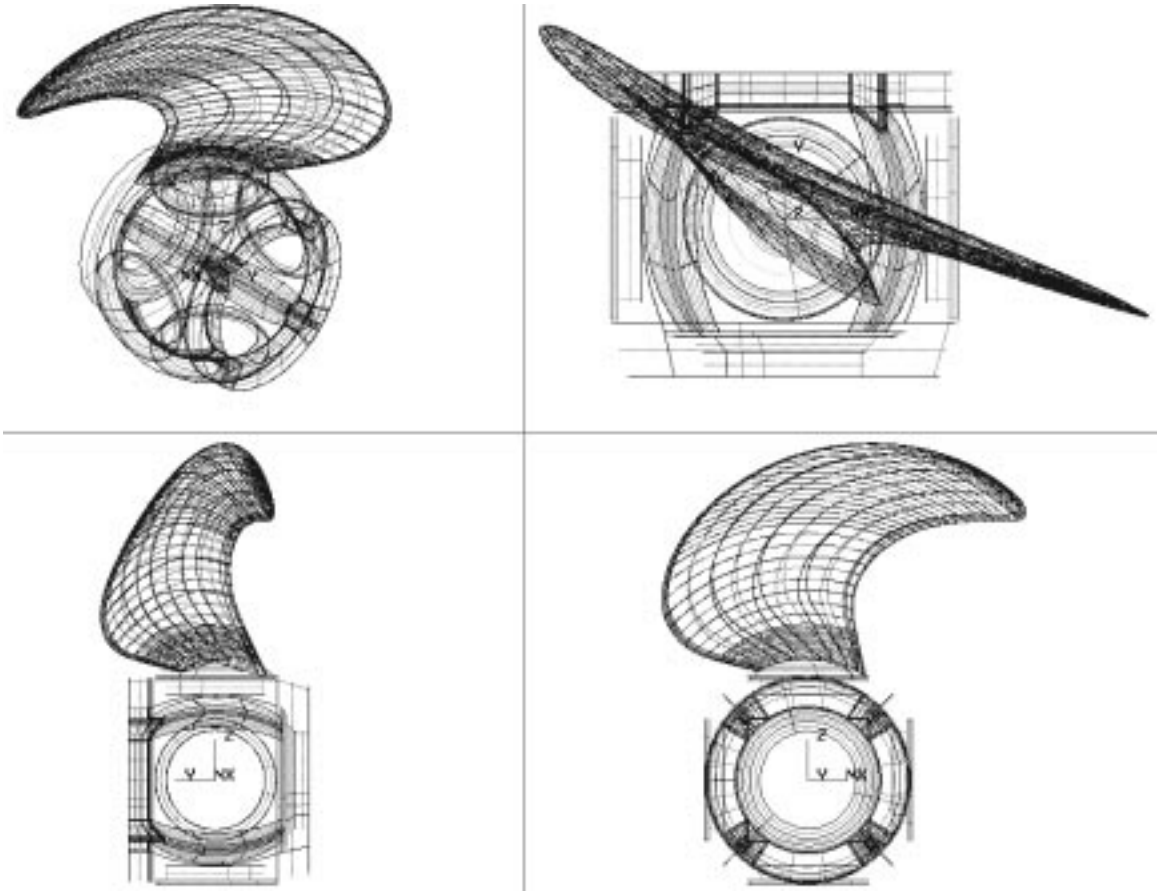


Figure 36 — Controllable screw propeller

4.2.82 Research_ship

A Research_ship is a type of Shiptype (see 4.2.98) that specifies the research function performed by a ship.

The data associated with a Research_ship are the following:

— has_type.

4.2.82.1 has_type

The has_type specifies the type of the research ship.

The has_type shall be one of the following:

— user_defined.

4.2.82.1.1 user_defined

a ship type that is not in the has_type list.

NOTE Details may be found in the Shiptype (see 4.2.99) description attribute.

4.2.83 Revision

A Revision is a type Versionable_object (see 4.2.120) that serves as the link between the object of interest and the definitions of its constituents and associated members. A Revision is not created automatically, but has to be created explicitly each time it is needed. Each Revision may be a Revision_with_context (see 4.2.84).

EXAMPLE The object of interest can be a hull cross section whose members are plate definitions, but only those plate definitions that belong to the same version.

The data associated with a Revision are the following:

— members;

— name.

4.2.83.1 members

The members specifies the Versionable_object (see 4.2.120) objects of the Revision. See 4.3.82 for the application assertion.

4.2.83.2 name

The name specifies a label which identifies a particular revision.

4.2.84 Revision_with_context

A Revision_with_context is a type of Revision (see 4.2.83) 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 Ship_moulded_form_revision (see 4.2.94).

The data associated with a Revision_with_context are the following:

— context_of_revision.

4.2.84.1 context_of_revision

The context_of_revision specifies the link to a Definable_object (see 4.2.22), which gets defined by the revision. See 4.3.83 for the application assertion.

4.2.85 Rotational_symmetry

A Rotational_symmetry is a type of Symmetry (see 4.2.110) that provides rotational symmetry information for any part of the ship. The rotational axis details the axis of rotation for this part of the ship.

The data associated with a Rotational_symmetry are the following:

— the_rotational_axis.

4.2.85.1 the_rotational_axis

The the_rotational_axis specifies the axis of rotation and is defined in the valid coordinate system of this part of the ship.

4.2.86 Rudder_moulded_form_design_parameter

A Rudder_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.63) that contains dimensions and ratios of the rudder.

EXAMPLE Figure 37 illustrates rudder dimensions.

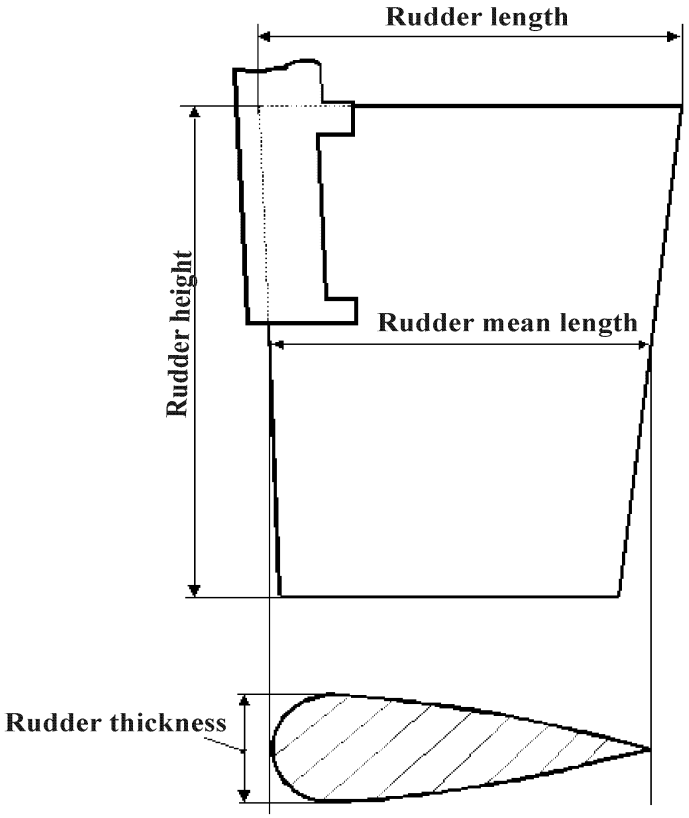


Figure 37 — Rudder dimensions

The data associated with a Rudder_moulded_form_design_parameter are the following:

— aspect_ratio;

— projected_rudder_area;

— rudder_height;

— rudder_length;

— rudder_location;

— rudder_mean_height;

— rudder_mean_length;

— rudder_thickness;

— type_of_the_rudder.

4.2.86.1 aspect_ratio

The aspect_ratio specifies the aspect ratio between the mean values of its height and length.

4.2.86.2 projected_rudder_area

The projected_rudder_area specifies the surface of the rudder in its operating placement projected into the XY-plane of the ship.

4.2.86.3 rudder_height

The rudder_height specifies the height of the rudder measured from its lowest to its highest point in the operating placement.

4.2.86.4 rudder_length

The rudder_length specifies the maximum length of the rudder in its operating placement measured lengthways of the ship.

4.2.86.5 rudder_location

The rudder_location specifies the placement of the centreline of the rudder shaft at the point where it is attached to the top of the rudder. The X coordinate is measured from the after perpendicular, the Y coordinate is measured from the centreline and the Z coordinate is measured from the baseline. See 4.3.84 for the application assertion.

4.2.86.6 rudder_mean_height

The rudder_mean_height specifies the mean height of the rudder in its operating placement.

4.2.86.7 rudder_mean_length

The rudder_mean_length specifies the mean length of the rudder in its operating placement measured in the length direction of the ship.

4.2.86.8 rudder_thickness

The rudder_thickness specifies the maximum thickness of the rudder.

4.2.86.9 type_of_the_rudder

The type_of_the_rudder specifies whether it is a balanced or unbalanced rudder.

The type_of_the_rudder shall be one of the following:

— balanced;

— unbalanced.

NOTE See 4.2.86.9.1 - 4.2.86.9.2 for definitions for each allowable value for type_of_the_rudder.

4.2.86.9.1 balanced

The rudder area is equally distributed in front and behind the rudder turning axis in order to reduce the operating torque at the rudder stock.

4.2.86.9.2 unbalanced

The rudder area is located behind the rudder axis of turning.

4.2.87 Section_of_offset_point_table

A Section_of_offset_point_table is an ordered list of Ship_point (see 4.2.96) objects.

NOTE This is similar to a Ship_curve (see 4.2.89) represented by a polyline.

The data associated with a Section_of_offset_point_table are the following:

— for_table;

— section_identifier;

— section_points.

NOTE The for_table attribute is the inverse of the attribute offset_point_table_sections for Offset_point_table_-model (see 4.2.73).

4.2.87.1 for_table

The for_table specifies the associated Offset_point_table_model (see 4.2.73) objects for the Section_of_-offset_point_table. See 4.3.75 for the application assertion.

4.2.87.2 section_identifier

The section_identifier specifies the context-specific identification given to the section of the offset table.

4.2.87.3 section_points

The section_points specifies the list of Ship_point (see 4.2.96) objects representing the offsets of the shape for this section. See 4.3.85 for the application assertion.

4.2.88 Ship

A Ship is a type of Item (see 4.2.52) that is the primary product supported by the suite of ISO 10303 shipbuilding application protocols. All data defining the product shall be related to a Ship. Product model definition data related to the Ship object 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.

The data associated with a Ship are the following:

— single_hull_or_class;

— ship_items;

— units.

4.2.88.1 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 single_hull_or_class shall be one of the following:

— design_for_multiple_hulls;

— design_for_single_hulls.

4.2.88.1.1 design_for_multiple_hulls

The range of ship hulls for which a particular Item (see 4.2.52) or Definition (see 4.2.23) are applicable shall be specified using the Hull_applicability (see 4.2.43).

4.2.88.1.2 design_for_single_hulls

The one hull for which a particular Item (see 4.2.52) or Definition (see 4.2.23) is applicable shall be specified using the Hull_applicability (see 4.2.43).

4.2.88.2 ship_items

The ship_items specifies the constituent objects that are applicable to a particular Ship. See 4.3.49 for the application assertion.

4.2.88.3 units

The units specifies the reference to a set of pre-defined units for all types of measures that may appear in the Ship model. Each units may be either a Derived_unit (see 4.2.24) or a Named_unit (see 4.2.70). The units need not be specified for a particular Ship. There may be more than one units for a Ship. See 4.3.86 and 4.3.87 for application assertions.

4.2.89 Ship_curve

A Ship_curve is a type of Moulded_form_representation_item (see 4.2.67) that represents a curve with associated naval architectural information. A Ship_curve may be a Ship_curve_with_spacing_position (see 4.2.91).

EXAMPLE A type of a Ship_curve is a waterline that has geometry defined by a B-spline curve. Figure 37 illustrates ship curves.

The data associated with a Ship_curve are the following:

— curve_class;

— curve_shape;

— side_condition.

4.2.89.1 curve_class

The curve_class specifies the naval architectural categories for the Ship_curve that are required for the design definition of Moulded_form (see 4.2.61) objects.

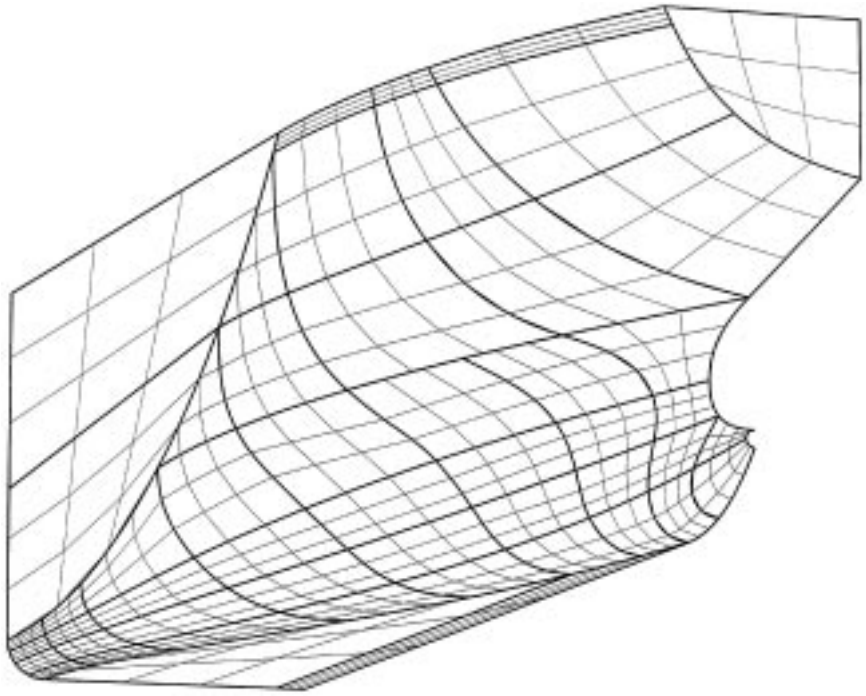


Figure 38 — Ship curves

The curve_class shall be one of the following:

— bounding_line;

— buttock_line;

— camber;

— centreline;

— flat_of_bottom;

— flat_of_side;

— intersection_line;

— rise_of_floor;

— sheer;

— station_line;

— trace_line;

— unspecified;

— waterline.

4.2.89.1.1 bounding_line

The curve is the bounding curve of a Moulded_form (see 4.2.61) or a Ship_surface (see 4.2.97).

4.2.89.1.2 buttock_line

The curve lies on the moulded surface of a hull at the intersection of a longitudinal plane and the hull moulded form.

EXAMPLE Figure 39 illustrates buttock lines.

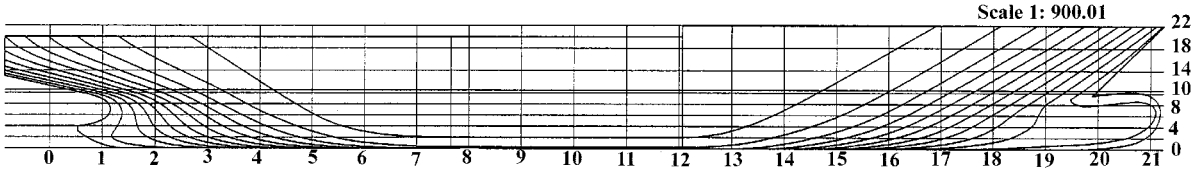


Figure 39 — Buttock lines

4.2.89.1.3 camber

The curve lies in the moulded surface of a deck.

NOTE It is generated by the intersection of a transverse plane with the deck moulded form.

4.2.89.1.4 centreline

The curve lies in the moulded surface of a hull.

NOTE It is generated by the intersection of the longitudinal centreplane with the hull moulded form.

4.2.89.1.5 flat_of_bottom

The boundary curve of the planar surface at the base of a hull moulded form.

4.2.89.1.6 flat_of_side

The boundary curve of the planar surface at the outer-most port or starboard side of the hull moulded form.

4.2.89.1.7 intersection_line

A curve that is generated by the intersection of two Moulded_form (see 4.2.61) objects or surfaces.

4.2.89.1.8 rise_of_floor

The boundary curve of the rise of floor at the bottom of the hull moulded form where the bilge starts.

4.2.89.1.9 sheer

The curve lies in the moulded surface of a deck.

NOTE It is generated by the intersection of a longitudinal plane with the deck moulded form.

4.2.89.1.10 station_line

The curve lies in the moulded surface of a hull.

NOTE It is generated by the intersection of a transverse plane with the hull moulded form.

EXAMPLE Figure 40 illustrates station lines.

4.2.89.1.11 trace_line

A curve in the moulded surface of a structural part, which acts as the boundary or as the trace line for the attachment of a profile or girder.

4.2.89.1.12 unspecified

The naval architectural meaning of the curve is not specified.

4.2.89.1.13 waterline

The curve lies in the moulded surface of a hull.

NOTE It is generated by the intersection of the water plane with the hull moulded form.

EXAMPLE Figure 41 illustrates waterline curves.

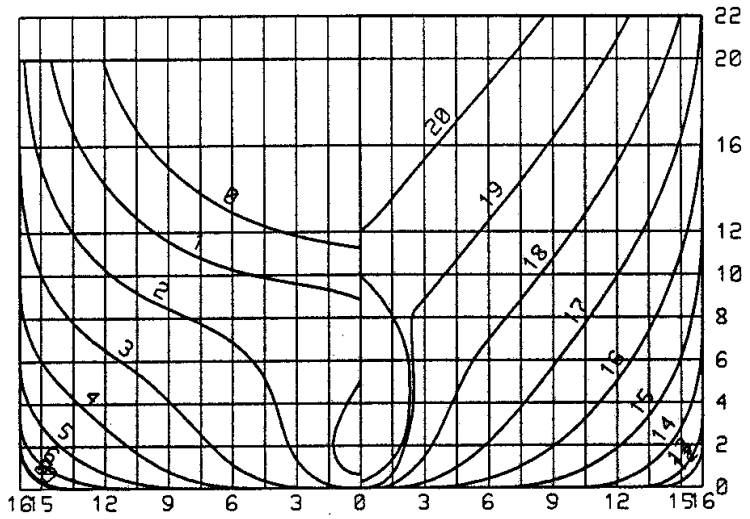


Figure 40 — Station lines

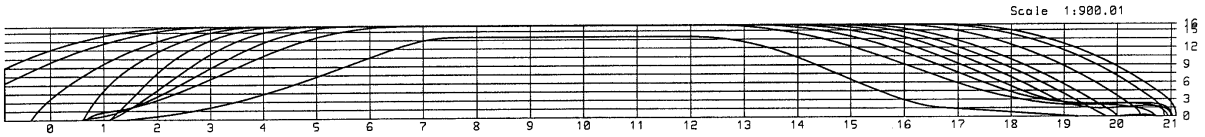


Figure 41 — Waterlines

4.2.89.2 curve_shape

The curve_shape specifies the underlying geometric definition of the Ship_curve (see 4.2.89).

EXAMPLE Wireframe or non-manifold B-spline representations.

4.2.89.3 side_condition

The side_condition specifies the behavior of curves crossing the defined curve at the crossover point.

EXAMPLE A section curve will knuckle at a waterline representing the intersection of the bilge keel and the flat of side.

The side_condition shall be one of the following:

— knuckle;

— smooth;

— tangent;

— unspecified.

4.2.89.3.1 knuckle

The crossing curve has a discontinuous tangent on either side of the crossover point.

4.2.89.3.2 smooth

The crossing curve is smooth on either side of the crossover point.

4.2.89.3.3 tangent

The crossing curve has a specified tangent at the crossover point.

4.2.89.3.4 unspecified

The crossing curve has no known tangency information at the crossover point.

4.2.90 Ship_curve_segment

A Ship_curve_segment defines the curve information for a wireframe representation of the ship or part of the ship. The Ship_curve_segment is one of the bounding curves of a cell in a wireframe with information about the Ship_curve (see 4.2.89) to which the Ship_curve_segment belongs.

NOTE The curve information will support the entity edge_curve as defined in ISO 10303-42.

The data associated with a Ship_curve_segment are the following:

— part_of_ship_curve.

4.2.90.1 part_of_ship_curve

The part_of_ship_curve specifies the Ship_curve (see 4.2.89) that the Ship_curve_segment is part of. See 4.3.88 for the application assertion.

4.2.91 Ship_curve_with_spacing_position

A Ship_curve_with_spacing_position is a type of Ship_curve (see 4.2.89) that has an associated spacing position.

EXAMPLE A waterline that has the geometry defined by a B-spline curve and located at a vertical spacing position.

NOTE A Ship_curve_with_spacing_position can only be defined for a two-dimensional ship curve located in one of the planes of the global coordinate system.

The data associated with a Ship_curve_with_spacing_position are the following:

— location.

4.2.91.1 location

The location specifies a Spacing_position (see 4.2.100) where the ship curve is defined. See 4.3.89 for the application assertion.

4.2.92 Ship_designation

A Ship_designation is a type of General_characteristics_definition (see 4.2.40) 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;

— flag_state;

— local_units;

— port_of_registration;

— ship_identification;

— ship_name;

— description.

4.2.92.1 call_sign

The call_sign specifies a unique, life cycle identifier assigned to the ship by the flag_state for radio communication.

4.2.92.2 flag_state

The flag_state specifies the national authority with which the ship is registered.

4.2.92.3 local_units

The local_units specifies the units that are used by the definition and differ from the units globally defined for the ship. Local_units may be either a Derived_unit (see 4.2.24) or a Named_unit (see 4.2.70). Local_units need not be specified for a particular Ship_designation. There may be more than one local_units for a Ship_designation. See 4.3.91 and 4.3.90 for application assertions.

4.2.92.4 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.

4.2.92.5 ship_identification

The ship_identification specifies a general identifier unique to the ship assigned during the classification process.

4.2.92.6 ship_name

The ship_name specifies the name of the ship assigned by the owner.

4.2.92.7 description

The description specifies more details about the function of the ship and additional information about the cargo carried. If the Shiptype (see 4.2.98) is user_defined, then the description delivers the information for the type of the ship.

4.2.93 Ship_moulded_form

A Ship_moulded_form is a type of Item_structure (see 4.2.54) and a type of Item (see 4.2.52) that groups all Moulded_form (see 4.2.61) objects together.

EXAMPLE 1 A Ship_moulded_form may be a group of individual Moulded_form objects for the ship hull, propeller, rudder, and appendages.

EXAMPLE 2 Figure 42 illustrates moulded forms for ships.

NOTE A Ship_moulded_form shall never be referenced by a Local_co_ordinate_system (see 4.2.56).

Figure 42 — Ship hull, rudder and propeller

The data associated with a Ship_moulded_form are the following:

— items;

— relationships.

4.2.93.1 items

The items specifies all Moulded_form (see 4.2.61) objects that define the Ship_moulded_form. See 4.3.92 for the application assertion.

4.2.93.2 relationships

The relationships specifies the Moulded_form_relationship (see 4.2.66) objects that form the Ship_-moulded_form. See 4.3.93 for the application assertion.

4.2.94 Ship_moulded_form_revision

A Ship_moulded_form_revision is a type of Revision_with_context (see 4.2.84) that details the revision of a Ship_moulded_form (see 4.2.93). It collects all used versions of Moulded_form_design_definition

(see 4.2.64) objects, which are relevant for the definition of a specific ship moulded form.

The data associated with a Ship_moulded_form_revision are the following:

— context_of_revision;

— members.

4.2.94.1 context_of_revision

The context_of_revision specifies the Ship_moulded_form (see 4.2.93) for which the Ship_moulded_-form_revision is being defined. See 4.3.95 for the application assertion.

4.2.94.2 members

The members specify a collection of Moulded_form_design_definition (see 4.2.64) objects that form the Ship_moulded_form_revision. See 4.3.94 for the application assertion.

4.2.95 Ship_overall_dimensions

A Ship_overall_dimensions is a type of Definition (see 4.2.23) that specifies the maximum limits of the Ship_moulded_form (see 4.2.93).

EXAMPLE Figure 43 illustrates ship overall dimensions.

The data associated with a Ship_overall_dimensions are the following:

— defined_for;

— overall_breadth;

— overall_depth;

— overall_length;

— stem_overhang;

— stern_overhang.

4.2.95.1 defined_for

The defined_for specifies the ship for this Ship_overall_dimensions. See 4.3.96 for the application assertion.

4.2.95.2 overall_breadth

The overall_breadth specifies the maximum breadth of the Ship_moulded_form (see 4.2.93).

4.2.95.3 overall_depth

The overall_depth specifies the maximum vertical distance from the baseline to the uppermost deck of the Ship_moulded_form (see 4.2.93).

4.2.95.4 overall_length

The overall_length specifies the length between the extreme forward and aft ends of the Ship_moulded_-form (see 4.2.93).

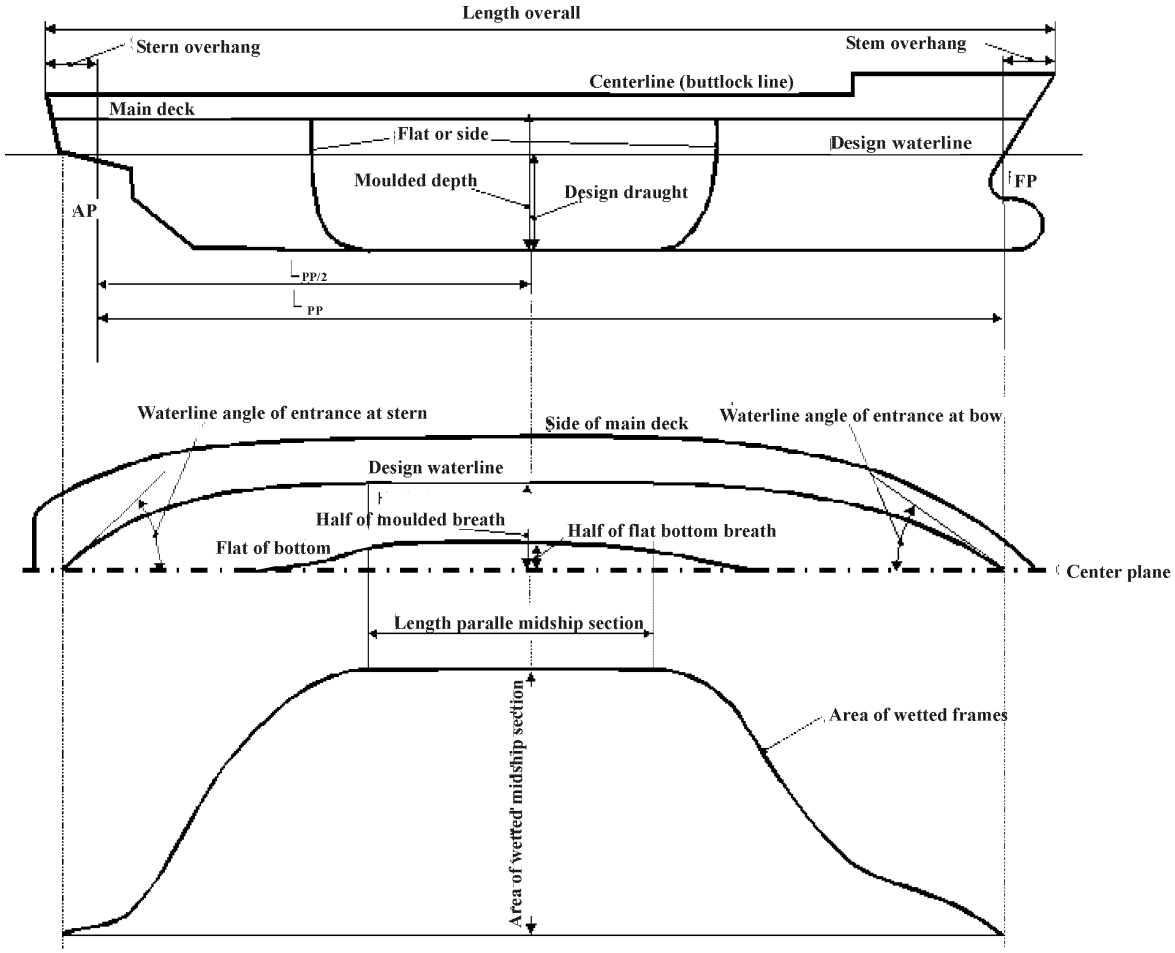


Figure 43 — Ship hull dimensions

4.2.95.5 stem_overhang

The stem_overhang specifies the length from the forward perpendicular to the extreme forward end of the Ship_moulded_form (see 4.2.93).

4.2.95.6 stern_overhang

The stern_overhang specifies the length from the extreme aft end of the Ship_moulded_form (see 4.2.93) to the after perpendicular.

4.2.96 Ship_point

A Ship_point is a type of Moulded_form_representation_item (see 4.2.67) that represents a point with associated naval architectural information.

EXAMPLE Figure 44 illustrates ship points.

A Ship_point may be a knuckle point that has a placement in the global coordinate system.

The data associated with a Ship_point are the following:

— point_class;

— point_shape.

4.2.96.1 point_class

The point_class specifies the naval architectural categories for the Ship_point based on the behaviour of curves passing through the Ship_point.

The point_class shall be one of the following:

— knuckle;

— ordinary;

— tangent;

— unspecified.

4.2.96.1.1 knuckle

The curves passing through the point have a discontinuous tangent on either side of the point.

4.2.96.1.2 ordinary

One or more curves that pass through the point have a smooth tangent.

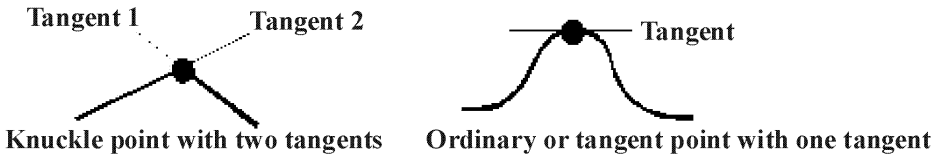


Figure 44 — Ship points

4.2.96.1.3 tangent

The curves passing through the point have a specified tangent.

4.2.96.1.4 unspecified

There is no known tangent information available for the curves passing through the point.

4.2.96.2 point_shape

The point_shape specifies the underlying geometric definition of the Ship_point.

EXAMPLE A type of point_type is offset table representation.

4.2.97 Ship_surface

A Ship_surface is a type of Moulded_form_representation_item (see 4.2.67) that represents a surface with associated naval architectural information.

The data associated with a Ship_surface are the following:

— surface_class;

— surface_shape.

4.2.97.1 surface_class

The surface_class specifies the naval architectural categories for the Ship_surface based on the placement of the surface.

The surface_class shall be one of the following:

— accommodation_area;

— accomodation_deck;

— aft_ship;

— blending_surface;

— bottom;

— bracket;

— bulkhead;

— cargo_area;

— collision_bulkhead;

— cross_tie;

— deck;

— deck_beam;

— deck_house;

— deck_in_superstructure;

— double_bottom;

— double_shell;

— duct_keel;

— engine_area;

— engine_foundation;

— external_surface;

— floor;

— fore_ship;

— frame;

— girder;

— hatch_cover;

— hatchway_coaming;

— hatchway_endcoaming;

— hatchway_sidecoaming;

— hold_bulkhead;

— hopper;

— inner_bottom;

— inner_shell;

— internal_surface;

— keel;

— longitudinal_bulkhead;

— longitudinal_girder;

— lower_boom;

— machinery_casing;

— main_deck;

— mid_ship;

— navigation_deck;

— outer_shell;

— platform_deck;

— plating;

— sheer_strake;

— ship_structure;

— stern_frame;

— stool;

— strength_bulkhead;

— strength_deck;

— stringer;

— superstructure;

— superstructure_aft_bulkhead;

— superstructure_front_bulkhead;

— superstructure_side_bulkhead;

— tank_bottom;

— tank_bulkhead;

— tank_side;

— tank_top;

— transom;

— transversal_bulkhead;

— transverse_floor;

— transverse_web_frame;

— upper_boom;

— user_defined;

— vertical_web_frame;

— wall;

— wash_bulkhead;

— weather_deck;

— web_frame;

— wing_bulkhead.

4.2.97.1.1 accommodation_area

A deckhouse or superstructure where berthing for passengers and crew is located.

4.2.97.1.2 accommodation_deck

A deck located in the accommodation area.

4.2.97.1.3 aft_ship

The portion of the ship which is aft of the midship section.

4.2.97.1.4 blending_surface

A surface that forms a connection between Moulded_form (see 4.2.61) objects.

EXAMPLE A type of blending_surface is a hull and an appendage to the hull (thruster).

4.2.97.1.5 bottom

The bottom of a shell.

4.2.97.1.6 bracket

A flanged plate or a regular plate used to provide rigid connection between two structural members.

4.2.97.1.7 bulkhead

A term applied to any of the partition walls which are used for subdividing the interior of a ship into various compartments.

NOTE bulkhead may be “holding” (watertight), structural (contributing to the strength of the ship), or non-structural (partition in the accommodation area).

4.2.97.1.8 cargo_area

The portion of the ship designated for the storage of the ship cargo, i.e. cargo hold.

4.2.97.1.9 collision_bulkhead

The bulkhead located in the bow area of a ship, which extends from the bottom of the ship to the weather deck, designed to provide watertight integrity in the event of bow damage, and also the foremost athwartship bulkhead in a ship.

4.2.97.1.10 cross_tie

An orthogonal structural member connecting two parallel beams or girders which provides extra strength to the deck.

4.2.97.1.11 deck

A platform in a ship corresponding to a floor in a building. It is the plating, planking, or covering of any tier of beams either in the hull or superstructure of a ship.

4.2.97.1.12 deck_beam

An athwartship or longitudinal horizontal structural member, usually a rolled shape, supporting a deck or flat.

4.2.97.1.13 deck_house

An enclosed structure on or above the weather deck that does not extend from side to side of the ship.

4.2.97.1.14 deck_in_superstructure

A deck in a superstructure.

4.2.97.1.15 double_bottom

A term applied to the space between the inner and outer bottom skins of a vessel, used for ballast water, fuel oil, etc.

NOTE Also applied to indicate that a ship has a complete inner or extra envelop of watertight bottom plating.

4.2.97.1.16 double_shell

A term used to describe a double hull ship, i.e. a damage in the outer hull will not affect the structural integrity of the ship or result in a loss of cargo.

4.2.97.1.17 duct_keel

The area within the double bottom, bounded by watertight longitudinal plate girders in which systems are typically run (also called a pipe tunnel)

4.2.97.1.18 engine_area

The ship engine room or machinery space where the main propulsion machinery is located.

4.2.97.1.19 engine_foundation

A heavy plate stiffened structure that provides structural support to the main propulsion system, typically mounted to the top of the inner bottom structure.

4.2.97.1.20 external_surface

A surface that is part of the outer surface representation of the ship.

4.2.97.1.21 floor

A plate used vertically in the bottom of a ship usually on every frame, to deepen it, and running athwartship from bilge to bilge.

NOTE transverse plate structures typically located in the double bottom, used to provide subdivision of the double bottom and support bottom shell and inner bottom longitudinal members

4.2.97.1.22 fore_ship

A term used indicating portions or part of a ship at or adjacent to the bow. Also applied to that portion and parts of the ship lying between the midship section and bow of the ship.

4.2.97.1.23 frame

A term generally used to designate one of the transverse ribs that make up the skeleton of the ship. The frames act as stiffeners, holding the side shell plating in shape and maintaining the transverse form of the ship.

4.2.97.1.24 girder

A primary structural support member which runs longitudinally in way of decks, shells, and bulkheads.

NOTE a girder provides support and rigidity to smaller and more closely spaced beams.

4.2.97.1.25 hatch_cover

A structural closure provided for large deck openings (hatchway). The hatchways are provided for hoisting freight or machinery in and out, and for access to the hold or machinery spaces.

4.2.97.1.26 hatchway_coaming

The four vertical-web pieces in the frame of a hatchway are collectively called the hatchway coaming.

4.2.97.1.27 hatchway_endcoaming

The athwartship vertical-web pieces at the forward and aft ends of the hatchway that land on the deck beams.

4.2.97.1.28 hatchway_sidecoaming

The longitudinal vertical-web pieces at the sides of the hatchway that rest on the hatch carlings against which the deck beams abut.

4.2.97.1.29 hold_bulkhead

A longitudinal or transverse partition wall, used to partition or subdivide the hold (the space allotted for the stowage of cargo).

4.2.97.1.30 hopper

A dry cargo hold space with sloping bulkhead plates in the bottom on all four sides to facilitate directing the bulk cargoes towards the unloading system.

4.2.97.1.31 inner_bottom

An inner or extra envelope of watertight bottom plate in a ship with double bottom. The plates forming the top of the double bottom are also called tank top.

NOTE a double bottom is usually fitted in large ships extending from bilge to bilge and nearly the whole length fore and aft. Where there are two bottoms, the outer is called the “shell”, the inner the “inner bottom”

4.2.97.1.32 inner_shell

The surfaces which make up the inner plating of a double hull vessel.

4.2.97.1.33 internal_surface

A surface that is part of the inner surface representation of the ship.

EXAMPLE Types of internal_surface are girders or bulkheads

4.2.97.1.34 keel

The principal fore-and-aft component of a ship framing, located along the centerline of the bottom and connected to the stem or stern frames. Floors or bottom transverses are attached to the keel.

NOTE flat plate keel is horizontal, centerline, bottom shell strake constituting the lower flange of the keel. Center vertical keel is vertical centerline web of the keel. Bilge keel is a longitudinal fin fitted at the turn of the bilge to reduce rolling.

4.2.97.1.35 longitudinal_bulkhead

A vertical bulkhead running in fore and aft direction used to partition or subdivide the ship for strength, provide intact stability and separation of spaces.

4.2.97.1.36 longitudinal_girder

A term applied to the fore-and-aft girder in the bottom of a ship. These girders are usually made up from plates and shapes and are sometimes intercostal and sometimes continuous.

4.2.97.1.37 lower_boom

A structure located at a lower level in the ship for cargo handling purpose and works in conjunction with upper boom.

NOTE boom is a long round spar hinged at its lower end, usually to a mast or king post in a ship, and supported by a wire rope or tackle from aloft to the upper end of the boom. Cargo, store, etc., are lifted by tackle leading from the upper end of the boom.

4.2.97.1.38 machinery_casing

The primary function of the machinery casings is to protect the openings which are fitted in the weather deck over the engines and boilers (for access, for light and air, and for the uptakes from the boilers) against the sea entering the ship through these openings in heavy weather. Below the weather deck, the machinery casing separates machinery spaces from accommodation and cargo spaces.

4.2.97.1.39 main_deck

The principal deck of the main hull, usually the highest structure extending from stem to stern and providing strength to the main hull. This structure is also called the strength deck.

4.2.97.1.40 mid_ship

A point located equidistant between the forward and after perpendiculars of the ship.

4.2.97.1.41 navigation_deck

A structure typically located in the deckhouse, it is the deck where ship control/navigation is performed.

4.2.97.1.42 outer_shell

The structure of the outer hull in a double hull vessel that is in contact with the sea.

4.2.97.1.43 platform_deck

A lower deck, usually in the cargo space, which does not contribute to the longitudinal strength of the ship.

4.2.97.1.44 plating

A continuous material used to provide support, subdivision, closure, etc., for bulkheads, decks, deck houses, shells, inner bottoms, etc.

4.2.97.1.45 sheer_strake

The course of shell plating at strength deck level, usually strengthened and free of openings.

4.2.97.1.46 ship_structure

The structure of the whole ship.

4.2.97.1.47 stern_frame

A large casting, forging, or weldment attached to the after end of the keel. Incorporates the rudder gudgeons and in single-screw ships includes the propeller post.

4.2.97.1.48 stool

A structural support located at the base of a corrugated transverse bulkhead providing a transition between bulkhead and inner bottom.

4.2.97.1.49 strength_bulkhead

A bulkhead which contributes to the strength of a vessel.

4.2.97.1.50 strength_deck

The deck that is designed as the uppermost part of the main hull longitudinal strength girder. The bottom shell plating forms the lower most part of this girder.

4.2.97.1.51 stringer

A term applied to a fore-and-aft girder running along the side of a ship at the shell and also to the outboard strake of plating on any deck. Also the side pieces of a ladder or staircase to which the treads and risers are fastened.

4.2.97.1.52 superstructure

A decked-over structure above the upper deck, the outboard sides of which are formed by the shell plating. This structure is distinguished from a deckhouse which does not extend outboard to the ship sides.

4.2.97.1.53 superstructure_aft_bulkhead

The main transverse bulkhead forming the aft boundary of the superstructure house.

4.2.97.1.54 superstructure_front_bulkhead

The main transverse bulkhead forming the forward boundary of the superstructure house.

4.2.97.1.55 superstructure_side_bulkhead

The main longitudinal bulkheads on either side (port and starboard) forming side boundaries of the superstructure house.

4.2.97.1.56 tank_bottom

The lower boundary of the tank; bottom shell, top of inner bottom, or deck which forms the bottom boundary of a tank.

4.2.97.1.57 tank_bulkhead

The longitudinal or transverse vertical stiffened plate structures which are added to form the side and end boundaries of a tank.

4.2.97.1.58 tank_side

The longitudinal stiffened plate structures which are added to form the side boundaries of a tank.

4.2.97.1.59 tank_top

The horizontal stiffened plate structures which are added to form the upper boundary of the tank.

4.2.97.1.60 transom

A square-ended stern used to provide additional hull volume and deck space aft and(or) to decrease resistance in some high speed ships.

4.2.97.1.61 transversal_bulkhead

The vertical stiffened plate structures oriented athwartship and used to partition or subdivide the ship for strength, provide intact stability, and separation of spaces.

4.2.97.1.62 transverse_floor

The vertical transverse plate immediately above the bottom shell plating, often located at every frame, extending from bilge to bilge.

4.2.97.1.63 transverse_web_frame

A built-up frame to provide extra strength, usually consisting of a web plate flanged or otherwise stiffened on its edge, spaced several frame spaces apart, with the smaller, regular frames in between.

4.2.97.1.64 upper_boom

A structure which is located at a high level in a ship for cargo handling and works in conjunction with the lower_boom.

4.2.97.1.65 user_defined

The intended usage of the underlying structure is user defined.

4.2.97.1.66 vertical_web_frame

A type of transverse web frame.

4.2.97.1.67 wall

This term, although not a shipbuilding term, refers to any of the partition walls which are used for subdividing the interior of a ship into various compartments.

4.2.97.1.68 wash_bulkhead

A longitudinal or transverse non-tight bulkhead fitted in a tank to decrease the swashing action of the liquid contents as a ship rolls and pitches at sea.

NOTE it is also called a swash bulkhead

4.2.97.1.69 weather_deck

The uppermost continuous deck with no overhead protection.

4.2.97.1.70 web_frame

A built-up frame to provide extra strength, usually consisting of a web plate flanged or otherwise stiffened on its edge, spaced several frame spaces apart, with the smaller, regular frames in between.

4.2.97.1.71 wing_bulkhead

The outermost longitudinal bulkhead which forms the boundary of a wing tank (port or starboard).

4.2.97.2 surface_shape

The surface_shape specifies the underlying geometric definition of the Ship_surface.

4.2.98 Shiptype

A Shiptype is a type of Functional_definition (see 4.2.39) that is a description of the function, purpose or mission for which a Ship (see 4.2.88) is designed. Each Shiptype is either a Carrier (see 4.2.9), a Navy_ship (see 4.2.71), a Research_ship (see 4.2.82) or a Working_ship (see 4.2.126).

EXAMPLE Figure 45 and Figure 46 illustrates ship types.

Figure 45 — Ship types

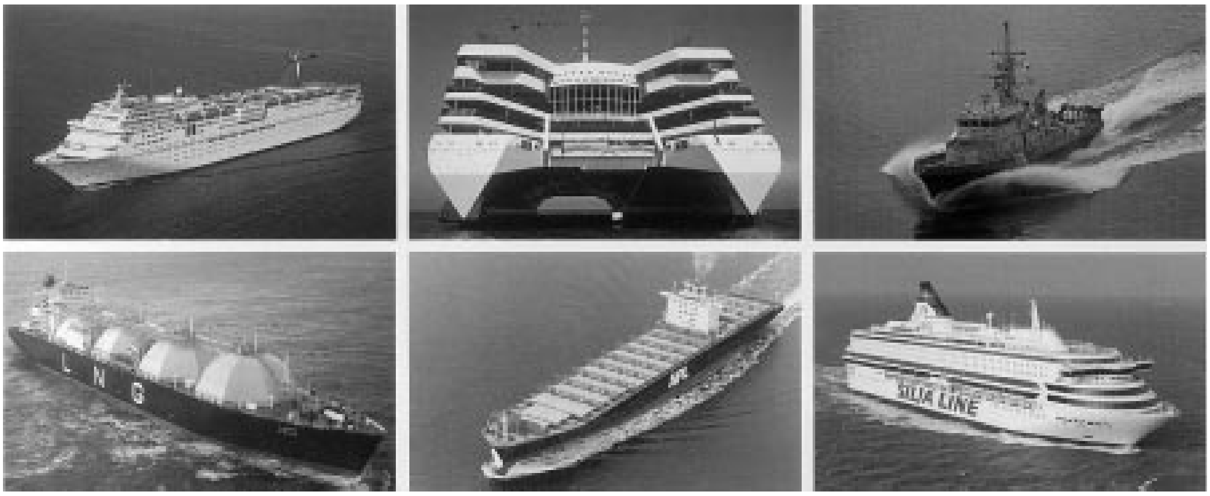


Figure 46 — Ship types

The data associated with a Shiptype are the following:

— defined_for;

— description.

4.2.98.1 defined_for

The defined_for specifies the ship for which the ship type is defined. See 4.3.97 for the application assertion.

4.2.98.2 description

The description specifies additional information about the ship.

4.2.99 Shipyard_designation

A Shipyard_designation is a type of General_characteristics_definition (see 4.2.40) 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.

4.2.99.1 local_units

The local_units specifies the units that are used by the definition and differ from the units globally defined for the ship. Local_units may be either a Derived_unit (see 4.2.24) or a Named_unit (see 4.2.70). Local_units need not be specified for a particular Shipyard_designation. There may be more than one local_units for a Shipyard_designation. See 4.3.98 and 4.3.99 for application assertions.

4.2.99.2 role

The role specifies the contractual obligation the shipyard has in relation to the ship.

The role shall be one of the following:

— prime;

— prime_build;

— prime_design;

— prime_repair;

— subcontractor.

NOTE See 4.2.100.2.1 - 4.2.100.2.5 for the definition of each allowable value for role.

4.2.99.2.1 prime

The shipyard is the prime contractor for the ship.

4.2.99.2.2 prime_build

The shipyard is the prime contractor with contract responsibilities for manufacture of the ship.

4.2.99.2.3 prime_design

The shipyard is the prime contractor with contract responsibility for the design of the ship.

4.2.99.2.4 prime_repair

The shipyard is the prime contractor with contract responsibilities for repair of the ship.

4.2.99.2.5 subcontractor

The shipyard is a secondary contractor, hired by the prime contractor, for the ship.

4.2.99.3 shipyard

The shipyard specifies the name and organizational details of the shipyard.

4.2.99.4 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.

4.2.99.5 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.

4.2.100 Spacing_position

A Spacing_position is a position 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 may be a Spacing_position_with_offset (see 4.2.101), a Longitudinal_position (see 4.2.58), a Transversal_position (see 4.2.112) or a Vertical_position (see 4.2.122).

EXAMPLE Spacing positions are typically specified by Longitudinal frame 123, or Transversal frame 10, 100, 100.1, A. In addition the distance to the global origin is defined for instance by 154.5 meters.

NOTE Figure 47 illustrates longitudinal, transversal, and vertical Spacing_position objects.

The data associated with a Spacing_position are the following:

— name;

— position;

— position_number.

4.2.100.1 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.

4.2.100.2 position

The position specifies the distance to the origin of the global coordinate system of the Ship (see 4.2.88). The axis on which the distance is measured depends on the type of Spacing_position.

4.2.100.3 position_number

The position_number specifies the numerical identification, which is given to the Spacing_position.

4.2.101 Spacing_position_with_offset

A Spacing_position_with_offset is a type of Spacing_position (see 4.2.100) 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.

4.2.101.1 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 (see 4.2.100).

4.2.101.2 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 (see 4.2.100).

4.2.101.3 relating_spacing_position

The relating_spacing_position specifies the spacing position from where the offset is taken to identify the Spacing_position_with_offset. See 4.3.100 for the application assertion.

4.2.102 Spacing_table

A Spacing_table is a type of Definition (see 4.2.23) that is a collection of Spacing_position objects (see 4.2.100) that defines a list of reference points along one of the coordinate axes of the ship. Each Spacing_table may be a Longitudinal_table (see 4.2.59), a Transversal_table (see 4.2.113) or a Buttock_-table (see 4.2.8).

The data associated with a Spacing_table are the following:

— name;

— spacing_table_representations.

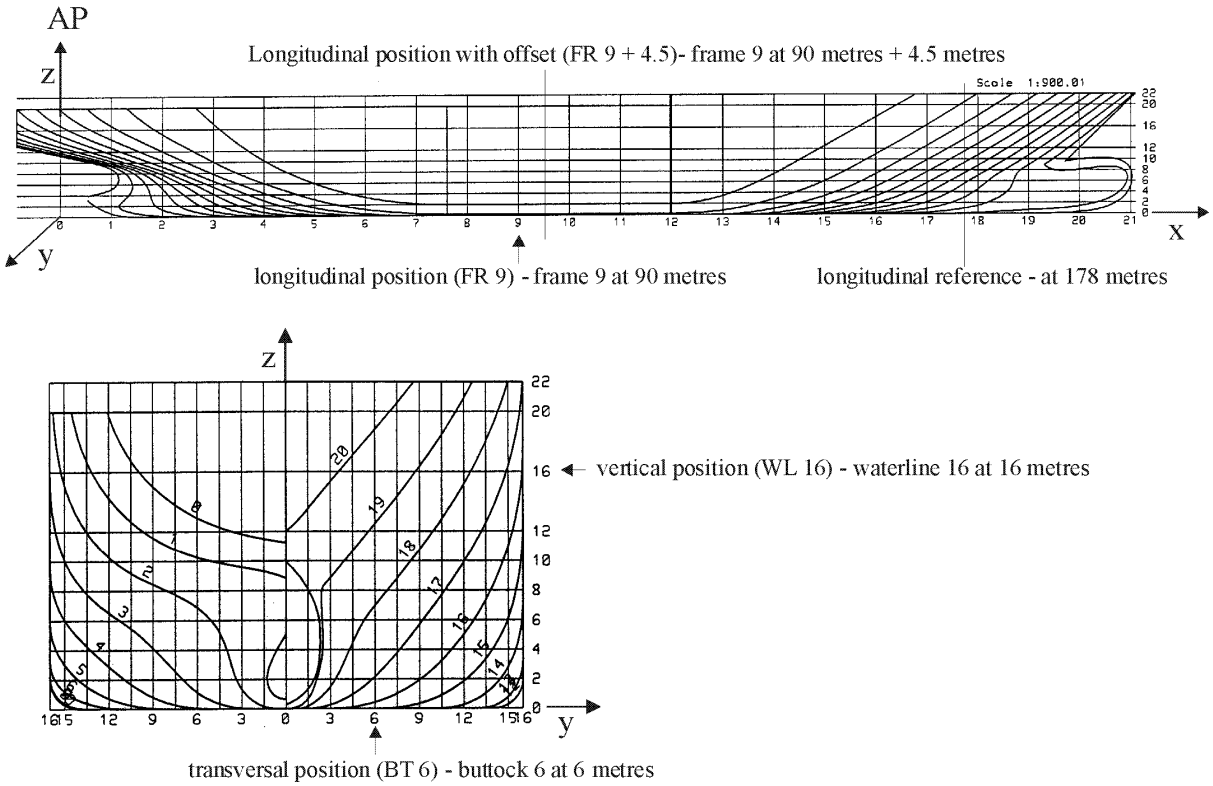


Figure 47 — Spacing position

4.2.102.1 name

The name specifies the context specific identification for the Spacing_table. The name need not be specified for a particular Spacing_table.

4.2.102.2 spacing_table_representations

The spacing_table_representations specifies the positions that make up the table on the coordinate axis that are of interest. See 4.3.101 for the application assertion.

4.2.103 Stability_definition

A Stability_definition is a type of Design_definition (see 4.2.25) 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.

NOTE Damage stability is a special case of Stability_definition.

The data associated with a Stability_definition are the following:

— defined_for;

— representations.

4.2.103.1 defined_for

The defined_for specifies the Ship (see 4.2.88) for which the Stability_definition is defined. See 4.3.102 for the application assertion.

4.2.103.2 representations

The representations specify the Stability_table (see 4.2.106), which the Stability_definition represents. See 4.3.103 for the application assertion.

4.2.104 Stability_properties_for_one_floating_position

A Stability_properties_for_one_floating_position is the collection of data specific to stability calculations for a particular Floating_position (see 4.2.37) for a Ship_moulded_form (see 4.2.93).

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.

4.2.104.1 centre_of_gravity_above_keel

The centre_of_gravity_above_keel specifies the placement for the centre of gravity of a ship. It will be presumed that the placement for the ship's centre of gravity will not change for different heel angles. The X coordinate is measured from the after perpendicular, the Y coordinate is measured from the centreline and the Z coordinate is measured from the baseline. See 4.3.104 for the application assertion.

4.2.104.2 definition_of_starting_floating_position

The definition_of_starting_floating_position specifies the Floating_position (see 4.2.37) 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 4.2.88). See 4.3.105 for the application assertion.

4.2.104.3 related_stability_table

The related_stability_table specifies the Stability_table (see 4.2.106) with which the Stability_-properties_for_one_floating_position are associated. There may be more than one related_stability_table for a Stability_properties_for_one_floating_position. See 4.3.108 for the application assertion.

4.2.104.4 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 Stability_properties_for_one_floating_position. There may be more than one stability_properties_for_different_angles_of_heel for a Stability_properties_for_one_floating_position. See 4.3.106 for the application assertion.

4.2.105 Stability_property

A Stability_property is all the necessary properties for the stability calculations for a Ship_moulded_form (see 4.2.93) with specific loading conditions.

The data associated with a Stability_property are the following:

— angle_of_heel;

— centre_of_buoyancy;

— righting_arm.

4.2.105.1 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.

4.2.105.2 centre_of_buoyancy

The centre_of_buoyancy specifies the placement of the volumetric centre of the submerged Ship_-moulded_form (see 4.2.93) volume. The X coordinate is measured from the after perpendicular, the Y coordinate is measured from the centreline and the Z coordinate is measured from the baseline. See 4.3.107 for the application assertion.

4.2.105.3 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.

4.2.106 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 (see 4.2.37).

EXAMPLE Figure 50 illustrates a stability table.

The data associated with a Stability_table are the following:

— items;

— mean_shell_thickness;

— name.

4.2.106.1 items

The items specifies the stability property that is the righting arm and the centre of buoyancy for a heel angle for one starting Floating_position (see 4.2.37) for which the Stability_table is defined. There may be more than one items for a Stability_table. See 4.3.108 for the application assertion.

4.2.106.2 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.

4.2.106.3 name

The name specifies a user or system defined name for the Stability_table.

4.2.107 Station_table

A Station_table is a type of Longitudinal_table (see 4.2.59) that has positions that reference the placement of stations that are located on 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 ship hull. There are usually 20 stations, but the number can differ from shipyard to shipyard.

4.2.108 Subtraction_of_moulded_form

A Subtraction_of_moulded_form is a type of Displacement_operation (see 4.2.26) that specifies a type of displacement_operation whereby the displacement of the Ship_moulded_form (see 4.2.93) will be reduced by the displacement of the new Moulded_form (see 4.2.61).

4.2.109 Surface_shape_representation

A Surface_shape_representation is a type of Moulded_form_shape_representation (see 4.2.69) that combines topological and geometric information together to describe a surface model as a set of connected surface patches.

EXAMPLE Figure 48 illustrates a surface_shape_representation.

The data associated with a Surface_shape_representation are the following:

— items.

4.2.109.1 items

The items specifies the set of face based surface models that build the Surface_shape_representation.

4.2.110 Symmetry

A Symmetry is either a Planar_symmetry (see 4.2.76) or a Rotational_symmetry (see 4.2.85), and provides the symmetry information for any part of the ship.

4.2.111 Thruster_moulded_form_design_parameter

A Thruster_moulded_form_design_parameter is a type of Moulded_form_characteristics_definition (see 4.2.63) that contains the dimensions and ratios of the thruster tunnel and propeller.

The data associated with a Thruster_moulded_form_design_parameter are the following:

— geometric_thruster_location;

— thruster_location;

— thruster_propeller_parameter;

— thruster_tunnel_diameter;

— thruster_tunnel_max_length;

— thruster_tunnel_min_length.

4.2.111.1 geometric_thruster_location

The geometric_thruster_location specifies the placement of the centreline of the thruster tunnel. The X coordinate is measured from the after perpendicular, the Y coordinate is measured from the centreline and the Z coordinate is measured from the baseline. See 4.3.109 for the application assertion.

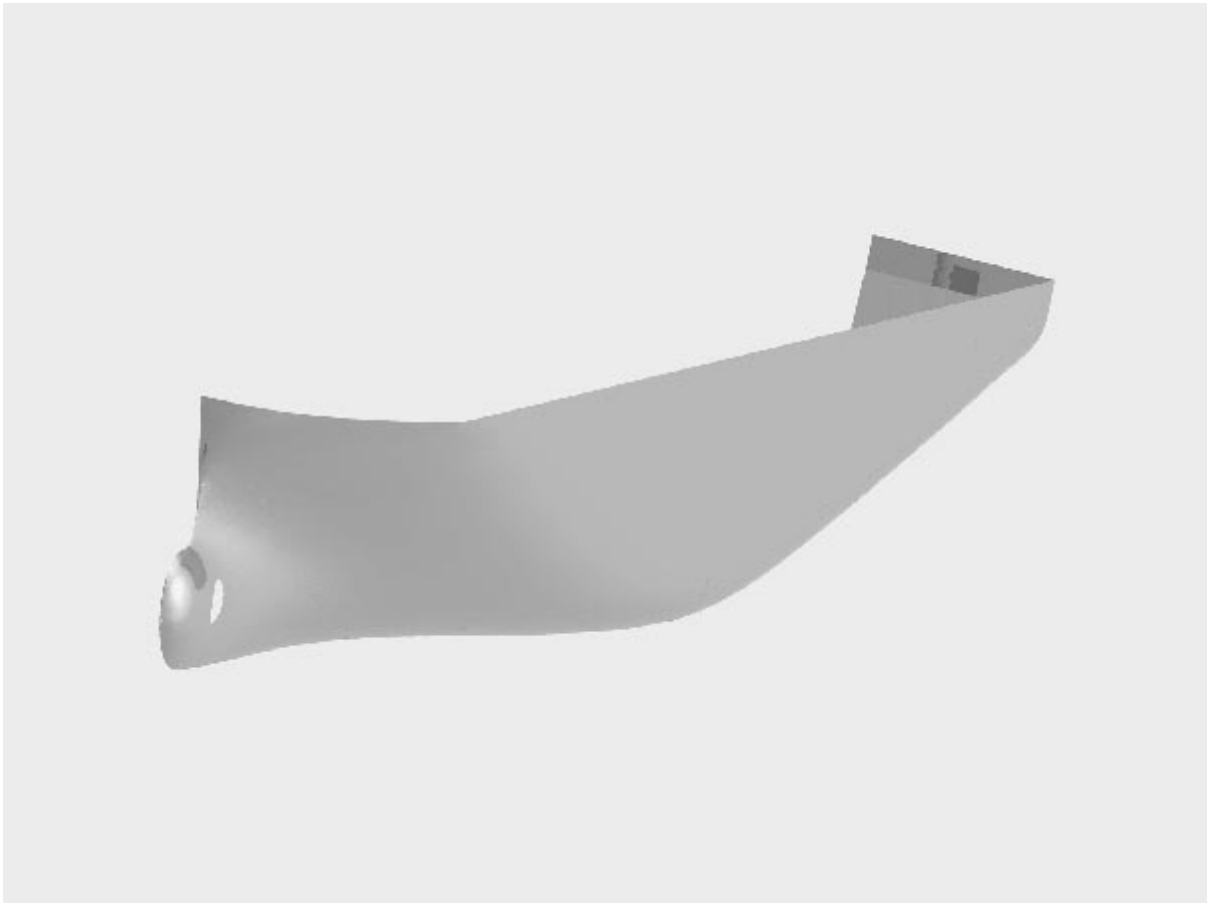


Figure 48 — Surface representation

4.2.111.2 thruster_location

The thruster_location specifies whether the thruster is located at the bow or at the stern.

The thruster_location shall be one of the following:

— bow;

— stern.

NOTE See 4.2.111.2.1 - 4.2.111.2.2 for definition of each allowable value for thruster_location.

4.2.111.2.1 bow

The placement is at the bow of the ship hull.

4.2.111.2.2 stern

The placement is at the stern of the ship hull.

4.2.111.3 thruster_propeller_parameter

The thruster_propeller_parameter specifies the dimensions and ratios of the propeller in the thruster tunnel. See 4.3.110 for the application assertion.

EXAMPLE Figure 49 illustrates a thruster propeller.

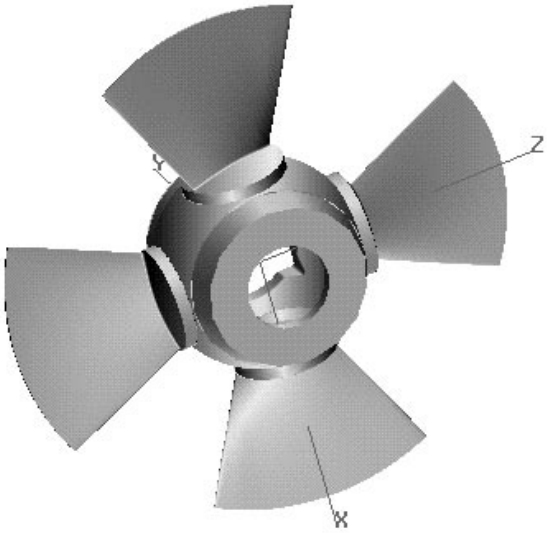


Figure 49 — Thruster propeller

4.2.111.4 thruster_tunnel_diameter

The thruster_tunnel_diameter specifies the diameter of the thruster tunnel.

4.2.111.5 thruster_tunnel_max_length

The thruster_tunnel_max_length specifies the maximum length of the thruster tunnel when it intersects with the ship hull.

4.2.111.6 thruster_tunnel_min_length

The thruster_tunnel_min_length specifies the minimum length of the thruster tunnel when it intersects with the ship hull.

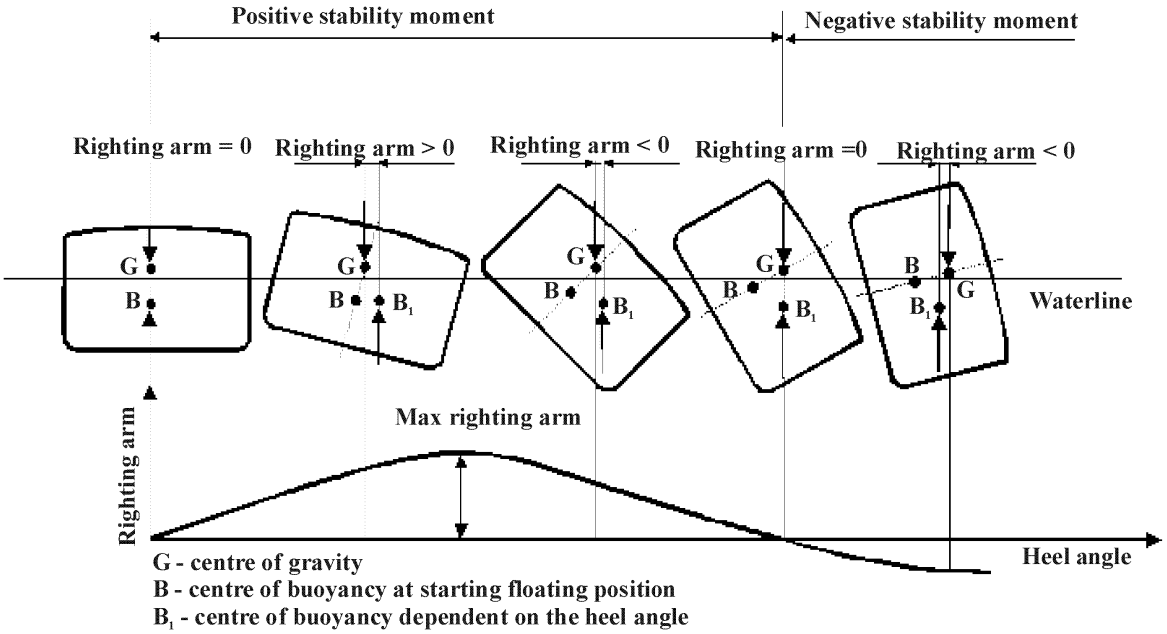


Figure 50 — Form stability

4.2.112 Transversal_position

A Transversal_position is a type of Spacing_position (see 4.2.100) that it is located on the global Y-axis.

4.2.113 Transversal_table

A Transversal_table is a type of Spacing_table (see 4.2.102) that has positions that are along the transverse axis of the global coordinate system that is the global Y-axis. A Transversal_table may be a Buttock_table (see 4.2.8).

The data associated with a Transversal_table are the following:

— spacing_table_representations.

4.2.113.1 spacing_table_representations

The spacing_table_representations specifies the transversal position for which the Transversal_table is defined. There may be more than one transversal position for a Transversal_table. See 4.3.111 for the application assertion.

4.2.114 Universal_resource_locator

A Universal_resource_locator is the address of an electronic data source. The electronic data soure is an internet address. The structure of an Universal_resource_locator may exactly follow the specification of the Uniform Resource Identifier (URI) as described in RFC 2396.

EXAMPLE A type of Universal_resource_locator is: http://www.w3.org.

The data associated with an Universal_resource_locator are the following:

— location.

4.2.114.1 location

The location specifies the storage place where a document or other information is located in the web.

NOTE Such a location is typically composed of several components, like protocol type, machine address (name or internet protocol number), an optional port number and a local path. Further components are possible.

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.

4.2.115 Version_creation

A Version_creation is a type of Versionable_object_change_event (see 4.2.121) that represents the event leading to a new Definition (see 4.2.23), Item_structure (see 4.2.54), or Item_relationship (see 4.2.53).

The data associated with a Version_creation are the following:

— base;

— subject.

4.2.115.1 base

The base specifies the Versionable_object (see 4.2.120) from which the subject is derived. The base shall exist if the object pointed to by the subject attribute is in some way derived from an existing Versionable_object. 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 (see 4.2.52), or to any Versionable_object of another Item which contributes to the creation of the subject. There may be more than one base for a Version_creation. See 4.3.112 for the application assertion.

4.2.115.2 subject

The subject specifies the Versionable_object (see 4.2.120) created by the change event. See 4.3.112 for the application assertion

4.2.116 Version_deletion

A Version_deletion is a type of Versionable_object_change_event (see 4.2.121) that is the event leading to the deletion of a Definition (see 4.2.23), an Item_structure (see 4.2.54), or an Item_relationship (see 4.2.53).

The data associated with a Version_deletion are the following:

— subject.

4.2.116.1 subject

The subject specifies the object deleted or to be deleted by the change event. See 4.3.113 for the application assertion.

4.2.117 Version_history

A Version_history is the identification of Versionable_object (see 4.2.120) objects and their Version_-relationship objects (see 4.2.119) in terms of their role as predecessors, successors and with respect to each other. The Version_history shall be a directed acyclic graph.

NOTE Consequently, the Version_history may contain Versionable_object objects considered alternatives with respect to each other (a Versionable_object having more than one successor), and merged Versionable_object objects (a Versionable_object having more than one predecessor).

The data associated with a Version_history are the following:

— current_version;

— relationships;

— versions.

4.2.117.1 current_version

The current_version specifies the Versionable_object (see 4.2.120) that plays the role of the current -version in this Version_history. See 4.3.115 for the application assertion.

4.2.117.2 relationships

The relationships specifies the Version_relationship (see 4.2.119) for which the Version_history is defined. There may be more than one Version_relationship for a Version_history. See 4.3.114 for the application assertion.

4.2.117.3 versions

The versions specifies the Versionable_object (see 4.2.120) for which the Version_history is defined. There may be more than one Versionable_object for a Version_history. See 4.3.115 for the application assertion.

4.2.118 Version_modification

A Version_modification is a type of Versionable_object_change_event (see 4.2.121) that represents the event leading to a change of a Versionable_object (see 4.2.120). 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 (see 4.2.117) of the same Item (see 4.2.52).

EXAMPLE A type of 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.

4.2.118.1 base

The base specifies the Versionable_object (see 4.2.120) from which the subject is derived. There may be more than one base for a Version_modification. See 4.3.116 for the application assertion.

4.2.118.2 subject

The subject specifies the Versionable_object (see 4.2.120) modified or to be modified by the change event. See 4.3.116 for the application assertion.

4.2.119 Version_relationship

A Version_relationship defines the relationship of two Versionable_object (see 4.2.120) objects of the same type in terms of a Version_history (see 4.2.117).

The data associated with a Version_relationship are the following:

— predecessor;

— reason;

— successor.

4.2.119.1 predecessor

The predecessor specifies the Versionable_object (see 4.2.120) from which the successor is derived. See 4.3.115 for the application assertion.

4.2.119.2 reason

The reason specifies why a certain person created the new version, at a certain time.

4.2.119.3 successor

The successor specifies the Versionable_object (see 4.2.120) of which the predecessor is the preceding version. See 4.3.115 for the application assertion.

4.2.120 Versionable_object

A Versionable_object is an object that may be versioned. Each Versionable_object is either a Definition (see 4.2.23), a Document (see 4.2.27), a Revision (see 4.2.83), an Item_relationship (see 4.2.53) or an Item_structure (see 4.2.54).

The data associated with a Versionable_object are the following:

— description;

— version_id.

4.2.120.1 description

The description specifies additional information that identifies or describes the object. The description need not be specified for a particular Versionable_object.

4.2.120.2 version_id

The version_id specifies the version identifier of the Versionable_object.

4.2.121 Versionable_object_change_event

A Versionable_object_change_event is a type of Event (see 4.2.33) that is the generalization of the Event objects effectively changing a Definition (see 4.2.23), Item_structure (see 4.2.54), or Item_relationship (see 4.2.53). A Versionable_object_change_event may be a Envisaged_version_creation (see 4.2.32), a Version_creation (see 4.2.115), a Version_deletion (see 4.2.116), or a Version_modification (see 4.2.118).

4.2.122 Vertical_position

A Vertical_position is a type of Spacing_position (see 4.2.100) that is located on the global Z-axis.

4.2.123 Vertical_table

A Vertical_table is a type of Spacing_table (see 4.2.102) 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 4.2.124).

The data associated with a Vertical_table are the following:

— spacing_table_representations.

4.2.123.1 spacing_table_representations

The spacing_table_representations specify the vertical positions that make up the vertical table. See 4.3.118 for the application assertion.

4.2.124 Waterline_table

A Waterline_table is a type of Vertical_table (see 4.2.123) whose positions are a reference for the placement of waterlines and are located on the global Z-axis.

4.2.125 Wireframe_shape_representation

A Wireframe_shape_representation is a type of Moulded_form_shape_representation (see 4.2.69) that defines the wireframe representation of the ship or part of the ship. The wireframe representation is defined by a list of Ship_curve (see 4.2.89) objects. Most of the Ship_curve objects are running through the entire wireframe from one end to the other end.

EXAMPLE A typical Wireframe_shape_representation can be formed by a list of station and waterline curves .

EXAMPLE Figure 51 illustrates a Wireframe_shape_representation.

The data associated with a Wireframe_shape_representation are the following:

— items.

4.2.125.1 items

The items specifies the list of Ship_curve (see 4.2.89) objects that build the Wireframe_shape_-representation. See 4.3.119 for the application assertion.

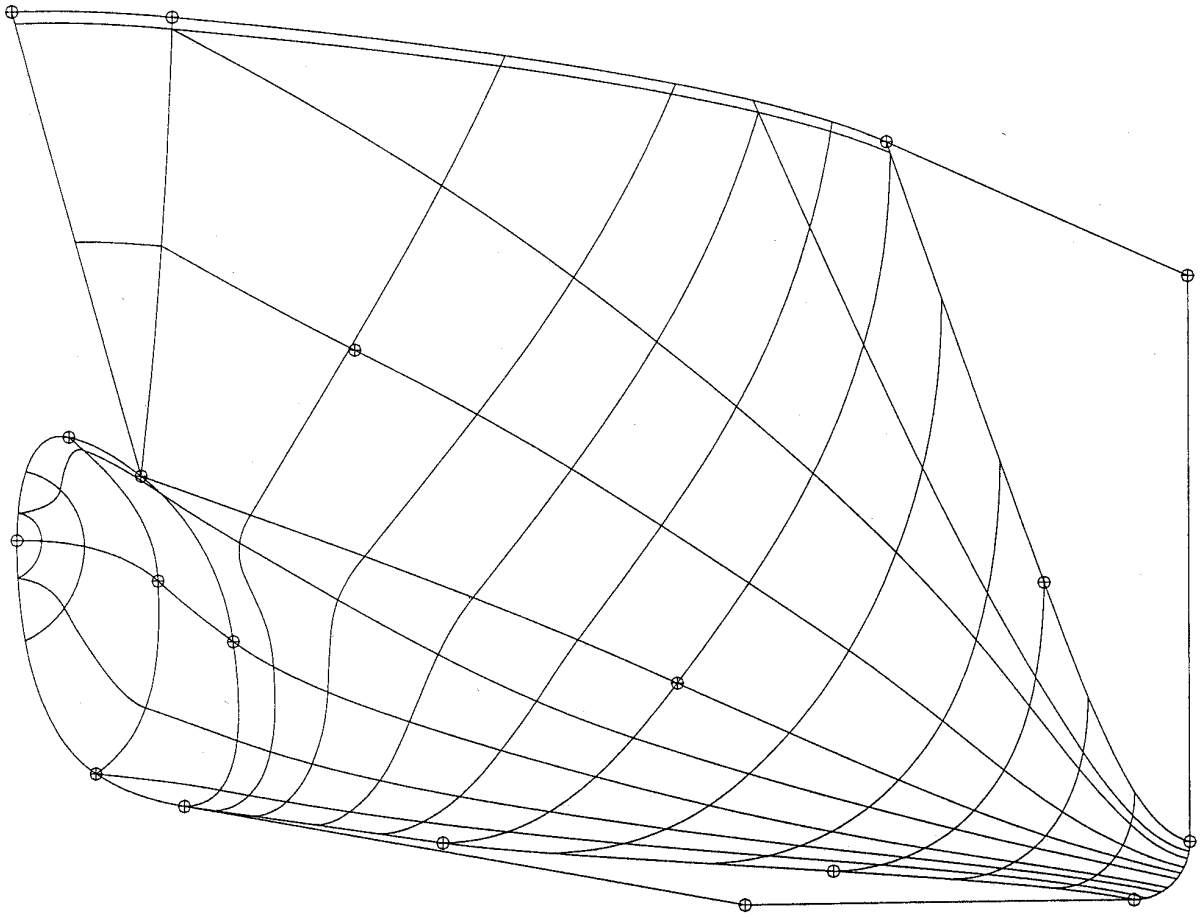


Figure 51 — Wireframe

4.2.126 Working_ship

A Working_ship is a type of Shiptype (see 4.2.98) that is a ship that is constructed to perform specific tasks.

The data associated with a Working_ship are the following:

— has_type.

4.2.126.1 has_type

The has_type specifies the type of the working ship.

The has_type shall be 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 4.2.126.1.1 - 4.2.126.1.25 for the definition of each allowable value for has_type.

4.2.126.1.1 crane_vessel

A Working_ship constructed for lifting purposes.

4.2.126.1.2 dredger

A Working_ship constructed for dredging channels or harbor entrances.

4.2.126.1.3 drilling_vessel

A Working_ship constructed for drilling purposes.

4.2.126.1.4 fire_fighter

A Working_ship constructed for fire fighting.

4.2.126.1.5 fishing_vessel

A Working_ship constructed for fishing.

4.2.126.1.6 floating_dock

A Working_ship constructed for lifting ships for repair.

4.2.126.1.7 floating_hotel

A Working_ship constructed for use as a hotel.

4.2.126.1.8 fpgo

A floating platform used for production and storage of gas.

4.2.126.1.9 fpso

A floating platform used for production and storage of oil.

4.2.126.1.10 ice_breaker

A Working_ship constructed for breaking ice.

4.2.126.1.11 offshore_supply_vessel

A Working_ship constructed for supplying offshore platforms.

4.2.126.1.12 oil_production_and_storage_vessel

A Working_ship used for production and storage of oil.

4.2.126.1.13 oil_production_vessel

A Working_ship constructed for production of oil.

4.2.126.1.14 oil_storage_vessel

A Working_ship constructed for storage of oil.

4.2.126.1.15 pilot_boat

A Working_ship constructed for carrying the pilot to the ship that he navigates in and out of the harbor.

4.2.126.1.16 pipe_laying_vessel

A Working_ship constructed for the purpose of laying pipe.

4.2.126.1.17 pusher

A Working_ship constructed to push other unpropelled ships

4.2.126.1.18 reefer

A Working_ship constructed for transporting refrigerated cargo.

4.2.126.1.19 sealer

A Working_ship constructed for the purpose of seal hunting.

4.2.126.1.20 shuttle_tanker

A Working_ship equipped for offshore oil loading.

NOTE It may be used for transporting oil from a platform to shore.

4.2.126.1.21 stern_trawler

A Working_ship constructed for trawling the sea from the stern of the ship.

4.2.126.1.22 supply_vessel

A Working_ship constructed for the purpose of transporting supplies.

EXAMPLE A type of supply_vessel is an offshore supply vessel.

4.2.126.1.23 tug

A Working_ship constructed for towing purposes.

4.2.126.1.24 user_defined

The ship type is not one of the pre-defined types defined here.

NOTE Details may be found in the Shiptype (see 4.2.99) description attribute.

4.2.126.1.25 well_stimulation_vessel

A Working_ship constructed for the purposes of stimulating a well.

4.3 Application assertions

This subclause specifies the application assertions for the ship moulded forms 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.

NOTE There are several assertions below which specify a cardinality of zero for the relationship between a subtype of Definition and Derived_unit (or Named_unit). This occurs because the local_units attribute of Definition is redeclared in the subtype to be an empty set.

4.3.1 Alternative_version_relationship to Versionable_object

Each Alternative_version_relationship has alternative_1 defined by exactly one Versionable_object. 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. Each Versionable_object defines alternative_2 for zero, one, or many Alternative_version_relationship objects.

4.3.2 Approval_history to Approval_event

Each Approval_history has approvals defined by a sequential list of one or more Approval_event objects. Each Approval_event defines the approval_reference for exactly one Approval_history.

4.3.3 Approval_history to Definition

Each Approval_history has subject defined by exactly one Definition. Each Definition defines the subject for zero, one or many Approval_history objects.

4.3.4 Change_definition to Change

Each Change_definition has defined_for defined by one or more Change objects. Each Change defines defined_for for zero, one, or many Change_definition objects.

4.3.5 Change_definition to Derived_unit

Each Change_definition has units defined by exactly zero Derived_unit. Each Derived_unit object defines units for exactly zero Change_definition.

NOTE This assertion is established through the unit select.

4.3.6 Change_definition to Named_unit

Each Change_definition has units defined by exactly zero Named_unit. Each Named_unit object defines units for exactly zero Change_definition.

NOTE This assertion is established through the unit select.

4.3.7 Change_impact to Versionable_object_change_event

Each Change_impact has impact defined by one or more Versionable_object_change_event objects. Each Versionable_object_change_event defines the impact for zero, one, or many Change_impact objects.

4.3.8 Change_plan to Change_impact

Each Change_plan has planned_impact defined by exactly one Change_impact. Each Change_impact defines the planned_impact for zero, one, or many Change_plan objects.

4.3.9 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.

4.3.10 Change_plan to Change_request

Each Change_plan has chosen_solution_for defined by exactly one Change_request. Each Change_-request defines the chosen_solution_for zero, one, or many Change_plan objects.

4.3.11 Change_realization to Change_impact

Each Change_realization has impact defined by exactly one Change_impact. Each Change_impact defines the impact for zero, one, or many Change_realization objects.

4.3.12 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.

4.3.13 Change_realization to Change_plan

Each Change_realization has realization_of defined by exactly one Change_plan. Each Change_plan defines the realization_of for zero, one, or many Change_realization objects.

4.3.14 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.

4.3.15 Class_and_statutory_designation to Class_notation

Each Class_and_statutory_designation has the_class defined by exactly one Class_notation. Each Class_-notation defines the_class for zero, one, or many Class_and_statutory_designation objects.

4.3.16 Class_and_statutory_designation to Named_unit

Each Class_and_statutory_designation has units defined by exactly zero Named_unit. Each Named_unit defines units for exactly zero Class_and_statutory_designation.

NOTE This assertion is established through the unit select.

4.3.17 Class_and_statutory_designation to Derived_unit

Each Class_and_statutory_designation has units defined by exactly zero Derived_unit. Each Derived_unit defines units for exactly zero Class_and_statutory_designation.

NOTE This assertion is established through the unit select.

4.3.18 Class_and_statutory_designation to Regulation

Each Class_and_statutory_designation has the_statutory defined by exactly one Regulation. Each Regulation defines the_statutory for zero, one, or many Class_and_statutory_designation objects.

4.3.19 Deck_moulded_form_design_parameter to Moulded_form

Each Deck_moulded_form_design_parameter has defined_for defined by one or more Moulded_form objects. Each Moulded_form defines the defined_for for zero, one, or many Deck_moulded_form_-design_parameter objects.

4.3.20 Definable_object to Global_id

Each Definable_object has id defined by exactly one Global_id. Each Global_id defines the id for zero, one, or many Definable_object objects.

4.3.21 Definition to Definable_object

Each Definition has defined_for defined by one or more Definable_object objects. Each Definable_object defines the definitions for zero, one, or more Definition objects.

4.3.22 Definition to Derived_unit

Each Definition has units defined by zero, one, or more Derived_unit objects.Each Derived_unit defines units for zero, one, or many Definition objects.

NOTE This assertion is established through the unit select.

4.3.23 Definition to Global_id

Each Definition has id defined by exactly one Global_id. Each Global_id defines the id for zero, one, or many Definition objects.

4.3.24 Definition to Named_unit

Each Definition has units defined by zero, one, or more Named_unit objects. Each Named_unit defines units for zero, one, or many Definition objects.

NOTE This assertion is established through the unit select.

4.3.25 Displacement_operation to Moulded_form

Each Displacement_operation has displacement_of_moulded_form_to_add_or_subtract defined by exactly one Moulded_form. Each Moulded_form defines the displacement_of_moulded_form_to_add_-or_subtract for zero, one, or many Displacement_operation objects.

4.3.26 Document_portion to Document

Each Document_portion has source defined by exactly one Document. Each Document defines the source for zero, one, or many Document_portion objects.

4.3.27 Document_reference to Document

Each Document_reference has the assigned_documnent defined by zero or one Document objects. Each Document defines the assigned_documnent for zero, one, or many Document_reference objects.

NOTE This assertion is established through the document_reference select.

4.3.28 Document_reference to Document_portion

Each Document_reference has the assigned_documnent defined by zero or one Document_portion objects. Each Document_portion defines the assigned_documnent for zero, one, or many Document_reference objects.

NOTE This assertion is established through the document_reference select.

4.3.29 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.

4.3.30 External_instance_reference to Global_id

Each External_instance_reference has target_guid defined by exactly one Global_id. Each Global_id defines the target_guid for zero, one, or many External_instance_reference objects.

4.3.31 External_reference to External_storage

Each External_reference has location defined by zero or one External_storage objects. Each External_storage defines the location of zero, one, or many External_reference objects.

NOTE This assertion is established through the any_address select.

4.3.32 External_reference to Universal_resource_locator

Each External_reference has location defined by zero or one Universal_resource_locator objects. Each Universal_resource_locator defines the location of zero, one, or many External_reference objects.

NOTE This assertion is established through the any_address select.

4.3.33 Functional_definition to Named_unit

Each Functional_definition has units defined by exactly zero Named_unit. Each Named_unit defines units for exactly zero Functional_definition.

NOTE This assertion is established through the unit select.

4.3.34 Functional_definition to Derived_unit

Each Functional_definition has units defined by exactly zero Derived_unit. Each Derived_unit defines units for exactly zero Functional_definition.

NOTE This assertion is established through the unit select.

4.3.35 General_characteristics_definition to Ship

Each General_characteristics_definition has defined_for defined by one or more Ship objects. Each Ship defines the defined_for, for zero, one, or many General_characteristics_definition objects.

4.3.36 Hull_applicability to Definition

Each Hull_applicability has definitions_for_hulls defined by zero, one, or many Definition objects. Each Definition defines the definitions_for_hulls for zero, one, or many Hull_applicability objects.

4.3.37 Hull_applicability to Item

Each Hull_applicability has items_for_hulls defined by zero, one, or many Item objects. Each Item defines the definitions_for_hulls for zero, one, or many Hull_applicability objects.

4.3.38 Hull_moulded_form_design_parameter to Midship_tumble

Each Hull_moulded_form_design_parameter has midship_tumble_data defined by zero or one Midship_-tumble objects. Each Midship_tumble defines the midship_tumble_data for zero, one, or many Hull_-moulded_form_design_parameter objects.

4.3.39 Hydrostatic_definition to Displacement_operation

Each Hydrostatic_definition has a sequential list of displacement_changes defined by zero, one, or many Displacement_operation objects. Each Displacement_operation defines the displacement_changes for zero, one, or many Hydrostatic_definition objects.

4.3.40 Hydrostatic_definition to Hydrostatic_table

Each Hydrostatic_definition has representations defined by one or more Hydrostatic_table objects. Each Hydrostatic_table define the representations for zero, one, or many Hydrostatic_definition objects.

4.3.41 Hydrostatic_definition to Ship

Each Hydrostatic_definition has defined_for defined by one or more Ship objects. Each Ship defines the defined_for for zero, one, or many Hydrostatic_definition objects.

4.3.42 Hydrostatic_position_value to Centre_location

Each Hydrostatic_position_value has position defined by exactly one Centre_location. Each Centre_location defines the position for zero, one, or many Hydrostatic_position_value objects.

4.3.43 Hydrostatic_properties_for_constant_floating_position to Floating_-position

Each Hydrostatic_properties_for_constant_floating_position has definition_of_floating_position defined by exactly one Floating_position. Each Floating_position defines the definition_of_floating_position for zero, one, or many Hydrostatic_properties_for_constant_floating_position objects.

4.3.44 Hydrostatic_properties_for_constant_floating_position to Hydrostatic_property_value

Each Hydrostatic_properties_for_constant_floating_position has hydrostatic_property_values defined by one or more Hydrostatic_property_value objects. Each Hydrostatic_property_value defines the hydrostatic_property_values for zero, one, or many Hydrostatic_properties_for_constant_floating_-position objects.

4.3.45 Hydrostatic_property to Centre_location

Each Hydrostatic_property has property_measure defined by exactly one Centre_location. Each Centre_location defines the property_measure for zero, one, or many Hydrostatic_property objects.

4.3.46 Hydrostatic_table to Hydrostatic_properties_for_constant_floating_-position

Each Hydrostatic_table has items defined by one or more Hydrostatic_properties_for_constant_floating_-position objects. Each Hydrostatic_properties_for_constant_floating_position defines the related_-hydrostatic_table for zero, one, or many Hydrostatic_table objects.

4.3.47 Hydrostatic_table to Hydrostatic_property

Each Hydrostatic_table has properties_in_the_hydrostatic_table defined by one or more Hydrostatic_-property objects. Each Hydrostatic_property define the properties_in_the_hydrostatic_table for zero, one, or many Hydrostatic_table objects.

4.3.48 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.

4.3.49 Item to Ship

Each Item has ship_context defined by zero or one Ship objects. Each Ship defines the ship_items for zero, one or many Item objects.

4.3.50 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.

4.3.51 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.

4.3.52 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.

4.3.53 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.

4.3.54 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.

4.3.55 Knot to Ship_curve

Each Knot has intersecting_ship_curves defined by a sequential list of two or more Ship_curve objects. Each Ship_curve defines the intersecting_ship_curves for zero, one, or many Knot objects.

4.3.56 Local_co_ordinate_system to Global_axis_placement

Each Local_co_ordinate_system has the parent defined by zero or one Global_axis_placement objects. Each Global_axis_placement defines the parent for zero, one, or many Local_co_ordinate_system objects.

NOTE This assertion is established through the co_ordinate_system select.

4.3.57 Local_co_ordinate_system to Local_co_ordinate_system

Each Local_co_ordinate_system has the parent defined by zero or one Local_co_ordinate_system objects. Each Local_co_ordinate_system defines the parent for zero, one, or many Local_co_ordinate_-system objects.

NOTE This assertion is established through the co_ordinate_system select.

4.3.58 Local_co_ordinate_system_with_position_reference to Spacing_position

Each Local_co_ordinate_system_with_position_reference has longitudinal_ref defined by exactly one Spacing_position. 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 exactly one Spacing_position. 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 exactly one Spacing_position. Each Spacing_position defines the vertical_ref for zero, one, or many Local_co_-ordinate_system_with_position_reference objects.

NOTE This assertion is established through the local_reference select.

4.3.59 Longitudinal_table to Longitudinal_position

Each Longitudinal_table has spacing_table_representations defined by zero, one, or many Longitudinal_-position objects. Each Longitudinal_position defines the spacing_table_representations for zero, one, or many Longitudinal_table objects.

4.3.60 Moulded_form_boundary_relationship to Moulded_form

Each Moulded_form_boundary_relationship has item_1 defined by exactly one Moulded_form. Each Moulded_form defines the item_1 for zero, one, or many Moulded_form_boundary_relationship objects.

4.3.61 Moulded_form_boundary_relationship to Moulded_form

Each Moulded_form_boundary_relationship has item_2 defined by exactly one Moulded_form. Each Moulded_form defines the item_2 for zero, one, or many Moulded_form_boundary_relationship objects.

4.3.62 Moulded_form_characteristics_definition to Moulded_form

Each Moulded_form_characteristics_definition has defined_for defined by one or more Moulded_form objects. Each Moulded_form defines the defined_for for zero, one, or many Moulded_form_-characteristics_definition objects.

4.3.63 Moulded_form_design_definition to Edge_based_wireframe_shape

Each Moulded_form_design_definition has moulded_surface defined by zero or one Edge_based_wireframe_shape objects. Each Edge_based_wireframe_shape defines the moulded_surface for zero, one, or more Moulded_form_design_definition objects.

NOTE This assertion is established through the surface_geometry and surface_representation selects.

Each Moulded_form_design_definition has representations defined by zero, one, or more Edge_based_wireframe_shape objects. Each Edge_based_wireframe_shape defines the representations for zero, one, or more Moulded_form_design_definition objects.

NOTE This assertion is established through the surface_representation select.

4.3.64 Moulded_form_design_definition to Non_manifold_surface_shape

Each Moulded_form_design_definition has moulded_surface defined by zero or one Non_manifold_surface_shape objects. Each Non_manifold_surface_shape defines moulded_surface for zero, one, or more Moulded_form_design_definition objects.

NOTE This assertion is established through the surface_geometry and surface_representation selects.

Each Moulded_form_design_definition has representations defined by zero, one, or more Moulded_form_design_definition objects. Each Moulded_form_design_definition defines the representations for zero, one, or more Moulded_form_design_definition objects.

NOTE This assertion is established through the surface_representation select.

4.3.65 Moulded_form_design_definition to Moulded_form

Each Moulded_form_design_definition has defined_for defined by one or more Moulded_form objects. Each Moulded_form defines the defined_for for zero, one, or many Moulded_form_design_definition objects.

4.3.66 Moulded_form_design_definition to Moulded_form_boundary_-relationship

Each Moulded_form_design_definition has borders defined by zero, one, or many Moulded_form_-boundary_relationship objects. Each Moulded_form_boundary_relationship defines the border for zero, one, or many Moulded_form_design_definition objects.

NOTE This assertion is established through the moulded_form_border select.

4.3.67 Moulded_form_design_definition to Moulded_form_shape_-representation

Each Moulded_form_design_definition has a moulded surface of zero or one Moulded_form_shape_-representation objects. Each Moulded_form_shape_representation is the moulded surface for zero, one, or many Moulded_form_design_definition objects.

NOTE This assertion is established through the surface_geometry select.

4.3.68 Moulded_form_design_definition to Ship_curve

Each Moulded_form_design_definition has borders defined by zero, one, or many Ship_curve objects. Each Ship_curve defines the border for zero, one, or many Moulded_form_design_definition objects.

NOTE This assertion is established through the moulded_form_border select.

4.3.69 Moulded_form_design_definition to Ship_surface

Each Moulded_form_design_definition has a moulded surface of zero or one Ship_surface objects. Each Ship_surface is the moulded surface for zero, one, or many Moulded_form_design_definition objects.

NOTE This assertion is established through the surface_geometry select.

4.3.70 Moulded_form_design_definition to Spacing_position

Each Moulded_form_design_definition has borders defined by zero, one, or many Spacing_position objects. Each Spacing_position defines the border for zero, one, or many Moulded_form_design_definition objects.

NOTE This assertion is established through the moulded_form_border and planar_boundary selects.

4.3.71 Moulded_form_functional_definition to Moulded_form

Each Moulded_form_functional_definition has defined_for defined by one or more Moulded_form objects. Each Moulded_form define the defined_for for zero, one, or more Moulded_form_functional_-definition objects.

4.3.72 Moulded_form_relationship to Moulded_form

Each Moulded_form_relationship has item_1 defined by exactly one Moulded_form. Each Moulded_-form defines the item_1 for zero, one, or many Moulded_form_relationship objects.

Each Moulded_form_relationship has item_2 defined by exactly one Moulded_form. Each Moulded_-form defines the item_2 for zero, one, or many Moulded_form_relationship objects.

4.3.73 Moulded_form_representation_relationship to Moulded_form_-shape_representation

Each Moulded_form_representation_relationship has rep_1 defined by exactly one Moulded_form_-shape_representation. Each Moulded_form_shape_representation defines the rep_1 for zero, one, or many Moulded_form_representation_relationship objects.

Each Moulded_form_representation_relationship has rep_2 defined by exactly one Moulded_form_-shape_representation. Each Moulded_form_shape_representation defines the rep_2 for zero, one, or many Moulded_form_representation_relationship objects.

4.3.74 Moulded_form_shape_representation to Symmetry

Each Moulded_form_shape_representation has moulded_form_symmetry defined by zero or one Symmetry objects. Each Symmetry defines the moulded_form_symmetry for zero, one, or many Moulded_form_shape_representation objects.

4.3.75 Offset_point_table_model to Section_of_offset_point_table

Each Offset_point_table_model has offset_point_table_sections defined by one or more Section_of_-offset_point_table objects. Each Section_of_offset_point_table objects defines the for_table for one or many Offset_point_table_model objects.

4.3.76 Offset_table_shape_representation to Offset_point_table_model

Each Offset_table_shape_representation has items defined by one or more Offset_point_table_model objects. Each Offset_point_table_model defines the items for zero, one, or many Offset_table_shape_-representation objects.

4.3.77 Owner_designation to Derived_unit

Each Owner_designation has units defined by exactly zero Derived_unit. Each Derived_unit defines units for exactly zero Owner_designation.

NOTE This assertion is established through the unit select.

4.3.78 Owner_designation to Named_unit

Each Owner_designation has units defined by exactly zero Named_unit. Each Named_unit defines units for exactly zero Owner_designation.

NOTE This assertion is established through the unit select.

4.3.79 Propeller_location to Centre_location

Each Propeller_location has propeller_location defined by exactly one Centre_location. Each Centre_location defines the propeller_location for zero, one, or many Propeller_location objects.

Each Propeller_location has shaft_line_location defined by exactly one Centre_location. Each Centre_location defines the shaft_line_location for zero, one, or many Propeller_location objects

4.3.80 Propeller_moulded_form_design_parameter to Propeller_location

Each Propeller_moulded_form_design_parameter has location_of_the_propeller_at_the_ship_hull defined by zero or one Propeller_location objects. Each Propeller_location defines the location_of_the _propeller_at_the_ship_hull for zero, one, or many Propeller_moulded_form_design_parameter objects.

4.3.81 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.

4.3.82 Revision to Versionable_object

Each Revision has members defined by one or more Versionable_object objects. Each Versionable_-object defines the members for zero, one, or many Revision objects.

4.3.83 Revision_with_context to Definable_object

Each Revision_with_context has context_of_revision defined by exactly one Definable_object. Each Definable_object defines the context_of_revision for zero, one, or many Revision_with_context objects.

4.3.84 Rudder_moulded_form_design_parameter to Centre_location

Each Rudder_moulded_form_design_parameter has rudder_location defined by exactly one Centre_location. Each Centre_location defines the rudder_location for zero, one, or many Rudder_moulded_form_design_parameter
objects.

4.3.85 Section_of_offset_point_table to Ship_point

Each Section_of_offset_point_table has section_points defined by a sequential list of one or more Ship_-point objects. Each Ship_point defines the section_points for zero, one, or many Section_of_offset_-point_table objects.

4.3.86 Ship to Derived_unit

Each Ship has units defined by zero, one, or more Derived_unit objects. Each Derived_unit defines units for zero, one, or many Ship objects.

NOTE This assertion is established through the unit select.

4.3.87 Ship to Named_unit

Each Ship has units defined by zero, one, or more Named_unit objects. Each Named_unit defines units for zero, one, or many Ship objects.

NOTE This assertion is established through the unit select.

4.3.88 Ship_curve_segment to Ship_curve

Each Ship_curve_segment has part_of_ship_curve defined by exactly one Ship_curve. Each Ship_curve defines the part_of_ship_curve for zero, one, or many Ship_curve_segment objects.

4.3.89 Ship_curve_with_spacing_position to Spacing_position

Each Ship_curve_with_spacing_position has location defined by exactly one Spacing_position. Each Spacing_position defines the location for zero, one, or many Ship_curve_with_spacing_position objects.

4.3.90 Ship_designation to Named_unit

Each Ship_designation has units defined by exactly zero Named_unit. Each Named_unit defines units for exactly zero Ship_designation.

NOTE This assertion is established through the unit select.

4.3.91 Ship_designation to Derived_unit

Each Ship_designation has units defined by exactly zero Derived_unit. Each Derived_unit defines units for exactly zero Ship_designation.

NOTE This assertion is established through the unit select.

4.3.92 Ship_moulded_form to Moulded_form

Each Ship_moulded_form has items defined by one or more Moulded_form objects. Each Moulded_form defines the items for zero, one, or many Ship_moulded_form objects.

4.3.93 Ship_moulded_form to Moulded_form_relationship

Each Ship_moulded_form has relationships defined by zero, one, or many Moulded_form_relationship objects. Each Moulded_form_relationship defines the relationships for zero, one, or many Ship_-moulded_form objects.

4.3.94 Ship_moulded_form_revision to Moulded_form_design_definition

Each Ship_moulded_form_revision has members defined by one or more Moulded_form_design_-definition objects. Each Moulded_form_design_definition is a member for zero, one, or more Ship_-moulded_form_revision objects.

4.3.95 Ship_moulded_form_revision to Ship_moulded_form

Each Ship_moulded_form_revision has context_of_revision defined by exactly one Ship_moulded_form. Each Ship_moulded_form defines the context_of_revision for zero, one, or more Ship_moulded_form_-revision objects.

4.3.96 Ship_overall_dimensions to Ship

Each Ship_overall_dimensions has defined_for defined by one or more Ship objects. Each Ship defines the defined_for for zero, one, or many Ship_overall_dimensions objects.

4.3.97 Shiptype to Ship

Each Shiptype has defined_for defined by one or more Ship objects. Each Ship defines the defined_for for zero, one, or many Shiptype objects.

4.3.98 Shipyard_designation to Named_unit

Each Shipyard_designation has units defined by exactly zero Named_unit. Each Named_unit defines units for exactly zero Shipyard_designation.

NOTE This assertion is established through the unit select.

4.3.99 Shipyard_designation to Derived_unit

Each Shipyard_designation has units defined by exactly zero Derived_unit. Each Derived_unit defines units for exactly zero Shipyard_designation.

NOTE This assertion is established through the unit select.

4.3.100 Spacing_position_with_offset to Spacing_position

Each Spacing_position_with_offset has relating_spacing_position defined by exactly one Spacing_-position. Each Spacing_position defines the relating_spacing_position for zero, one, or many Spacing_-position_with_offset objects.

4.3.101 Spacing_table to Spacing_position

Each Spacing_table has spacing_table_representations defined by zero or more Spacing_position objects. Each Spacing_position defines the spacing_table_representations for zero, one, or many Spacing_table objects.

4.3.102 Stability_definition to Ship

Each Stability_definition has defined_for defined by one or more Ship objects. Each Ship defines the defined_for for zero, one, or many Stability_definition objects.

4.3.103 Stability_definition to Stability_table

Each Stability_definition has representations defined by one or more Stability_table objects. Each Stability_table defines the representations for zero, one, or many Stability_definition objects.

4.3.104 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. Each Centre_location defines the centre_of_gravity_above_keel for zero, one, or many Stability_properties_for_one_floating_position objects.

4.3.105 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. Each Floating_position defines the definition_of_starting_floating_-position for zero, one, or many Stability_properties_for_one_floating_position objects.

4.3.106 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 more Stability_property objects. Each Stability_property define the stability_-properties_for_different_angles_of_heel for zero, one, or many Stability_properties_for_one_floating_-position objects.

4.3.107 Stability_property to Centre_location

Each Stability_property has centre_of_buoyancy defined by exactly one Centre_location. Each Centre_location defines the centre_of_buoyancy for zero, one, or many Stability_property objects.

4.3.108 Stability_table to Stability_properties_for_one_floating_position

Each Stability_table has items defined by one or more Stability_properties_for_one_floating_position objects. Each Stability_properties_for_one_floating_position defines a sequential list of related_-stability_table for one or more Stability_table objects.

4.3.109 Thruster_moulded_form_design_parameter to Centre_location

Each Thruster_moulded_form_design_parameter has geometric_thruster_location defined by exactly one Centre_location. Each Centre_location defines the geometric_thruster_location for zero, one, or many Thruster_moulded_form_design_parameter objects.

4.3.110 Thruster_moulded_form_design_parameter to Propeller_moulded_-form_design_parameter

Each Thruster_moulded_form_design_parameter has thruster_propeller_parameter defined by exactly one Propeller_moulded_form_design_parameter. Each Propeller_moulded_form_design_parameter defines the thruster_propeller_parameter for zero, one, or many Thruster_moulded_form_design_-parameter objects.

4.3.111 Transversal_table to Transversal_position

Each Transversal_table has spacing_table_representations defined by zero, one, or many Transversal_-position objects. Each Transversal_position defines the spacing_table_representations for zero, one, or many Transversal_table objects.

4.3.112 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. Each Versionable_object defines the subject for zero, one, or many Version_creation objects.

4.3.113 Version_deletion to Versionable_object

Each Version_deletion has subject defined by exactly one Versionable_object. Each Versionable_object defines the subject for zero, one, or many Version_deletion objects.

4.3.114 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.

4.3.115 Version_history to Versionable_object

Each Version_history has versions defined by one or more Versionable_object objects. Each Versionable_object define the versions for zero, one, or many Version_history objects.

Each Version_history has current_version defined by exactly one Versionable_object. Each Versionable_object defines the current_version for zero, one, or many Version_history objects.

4.3.116 Version_modification to Versionable_object

Each Version_modification has base defined by one or more 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. Each Version_object defines the subject for zero, one, or many Version_modification objects.

4.3.117 Version_relationship to Versionable_object

Each Version_relationship has predecessor defined by exactly one Versionable_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. Each Versionable_-object defines the successor for zero, one, or many Version_relationship objects.

4.3.118 Vertical_table to Vertical_position

Each Vertical_table has spacing_table_representations defined by zero, one, or many Vertical_position objects. Each Vertical_position defines the spacing_table_representations for zero, one, or many Vertical_table objects.

4.3.119 Wireframe_shape_representation to Ship_curve

Each Wireframe_shape_representation has items defined by one or more Ship_curve objects. Each Ship_curve defines the items for zero, one, or more Wireframe_shape_representation 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 5.1.1 below;

-- the text following is a comment (normally a clause reference).

Table 1 shows the key mappings for this part of 10303.

Table 1 — Key mappings for AP216

AP216 key ARM entities	Mapping to Part 40 entities
Shiptype	product_related_product_category
Ship	product

AP216 key ARM entities	Mapping to Part 40 entities
Ship_moulded_form Moulded_form Principal_characteristics Class_and_statutory_designation Class_parameters Ship_designation Shipyard_designation Owner_designation Global_axis_placement	product_definition
Moulded_form_relationship	product_definition_relationship
Moulded_form_functional_definition Class_notation Regulation Displacement_operation Hydrostatic_definition Stability_definition Moulded_form_characteristics Definition (and Subtypes) Local_co_ordinate_system Local_co_ordinate_system_with_position_reference Spacing_table	property_definition
Moulded_form_design_definition	product_definition_shape
Wireframe_shape_representation Surface_shape_representation	shape_representation non_manifold_surface_shape_representation
Hydrostatic_table Stability_table Midship_tumble Propeller_location	representation
Offset_table_shape_representation	shape_representation
Hydrostatic_property Hydrostatic_property_for_constant_floating_position Floating_position Stability_properties_for_one_floating_position Stability_property Offset_point_table_model Section_of_offset_point_table Ship_point Ship_curve Ship_surface Spacing_position	compound_representation_item

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 template 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 template 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 to 5.1.16. Each such template application is a reference to the template definition, based on the pattern established by the template signature, and supplies the value parameters that are to be 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 template 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 template signature or the value parameters in a template application;

, separates formal parameters in a template 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.

EXAMPLE An illustration of the usage of APPROVES to assign an approval to a product_definition_shape in the role of 'subject':

```
approval <-
/APPROVES(product_definition_shape,'subject')/
```

Signature:

/APPROVES(ENTITY, ARM_ROLE)/

Parameter definitions:

ENTITY: the identifier of the AIM entity data type to which an approval is assigned

ARM_ROLE: the value of the name attribute for object_role

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.

EXAMPLE An illustration of the usage of CLASS to assign a group with name 'plate' to a product_defintion:

/CLASS(product_definition, 'plate', 'structural part')/

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_classification ->
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**.

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_classification ->
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.

EXAMPLE An illustration of the usage of CLASS_ID that verifies the product instance has a class name ‘ship’:

action_item = product
product
{/CLASS_ID(product, ‘ship’)/}

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_classification ->
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.

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.

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 a **description_attribute** is assigned to an ENTITY that contains a derived description attribute. The derivation for the description attribute is based on the Basic_attribute_schema of ISO 10303-41.

NOTE The following entity data types use 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.

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 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.

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.

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.

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. 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.

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 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.

EXAMPLE An illustration of the usage of GROUPS to add a product_definition to the group with the role of 'item structure' :

group <-
/GROUPS(product_definition, 'item structure')/

Signature:

/GROUPS(ENTITY, ARM_ROLE)/

Parameter definitions:

ENTITY: the entities being grouped that play an ARM_ROLE

ARM_ROLE: the value of the **name** attribute for **object_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

5.1.1.13 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.

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
```

5.1.1.14 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.

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
```

5.1.1.15 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.

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_attributeid_attribute.attribute_value

5.1.1.16 ID_ROLE

The ID_ROLE mapping template specifies a reference path constraint in which an assignment of an **identification_assignment** has the role ROLE.

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]}
```

5.1.1.17 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.

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
```

5.1.1.18 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 4.2.52) and Item_structure (see 4.2.54).

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’
```

5.1.1.19 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.

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
```

5.1.1.20 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.

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'}
```

5.1.1.21 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**.

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
```


5.1.1.22 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.

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
```

5.1.1.23 PDR_NAME

The PDR_NAME mapping template specifies a reference path constraint in which a **property_-definition_representation.name** is assigned the value NAME.

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
```

5.1.1.24 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.

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
```

5.1.1.25 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.

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
```

5.1.1.26 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.

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}
```

5.1.1.27 PROD_DEF_PRODUCT

The PROD_DEF_PRODUCT mapping template specifies a reference path constraint in which a **product_definition** is linked with a **product**.

Signature:

/PROD_DEF_PRODUCT/

Parameter definitions:

none

Template body:

```
product_definition
product_definition.formation ->
product_definition_formation
product_definition_formation.of_product ->
product
```

5.1.1.28 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**.

Signature:

/PROD_DEF_PROP_DEF/

Parameter definitions:

none

Template body:

```
PROD_DEF_PROP_DEF_HELP
property_definition
```

5.1.1.29 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.

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
```

5.1.1.30 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_representaion** 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.

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)
```

5.1.1.31 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.

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)
```

5.1.1.32 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 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. PROD_DEF_TO_SPECIAL_VAL_REP_ITEM is intended to be used by other mapping templates.

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)
```

5.1.1.33 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.

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)
```

5.1.1.34 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.

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)
```

5.1.1.35 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.

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')]}
```

5.1.1.36 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_representaion** that has an attribute **name** with the specific value ID1.

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
```

5.1.1.37 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**.

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
```

5.1.1.38 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.

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)
```

5.1.1.39 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_-representaion** whose attribute **name** has the value ID. The property_definition_representation refers to the **global_unit_assigned_context** via a **representation** instance.

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
```

5.1.1.40 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.

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}
```

5.1.1.41 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**.

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
```

5.1.1.42 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’.

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')})
```

5.1.1.43 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'.

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')}
```

5.1.1.44 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'.

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')}
```

5.1.1.45 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'. This mapping template can be understood as the "inverse" mapping template to VERSION_ID(T).

Signature:

/RELATE_ID_2_VO(T)/

Parameter definitions:

T: type if object that has an **identification_assignment**

Template body:

```
identification_assignment
{identification_assignment.role ->
identification_role
[identification_role.name = 'version identifier']} =>
applied_identification_assignment
applied_identification_assignment.items[i] ->
identification_item = &T
{CLASS_ID(&T, 'versionable object')}
```

5.1.1.46 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.

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}
```

5.1.1.47 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.

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 =
derived_unit
[NAME_ASSGN_WITH_VAL(derived_unit,'&DER_UNIT_NAME')]}
representation.items [i] ->
representation_item =>
{representation_item.name = &ID}
value_representation_item
value_representation_item.value_component -> measure_value
{measure_value = context_dependent_measure }
```

5.1.1.48 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.

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}
```

5.1.1.49 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.

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

5.1.1.50 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.

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_classification ->
group =>
{group.name = &ID}
class
```

5.1.1.51 SDR_NAME

The SDR_NAME mapping template specifies a reference path constraint in which a **shape_definition_representation**.**name** is assigned the value NAME.

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
```

5.1.1.52 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.

Template signature:

/SUBTYPE(application_object)/

Parameter definition:

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.

5.1.1.53 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.

Template 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.

5.1.1.54 VERSION_ID

The VERSION_ID mapping template specifies a reference path constraint in which an entity of type T is given a version identifier.

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 basic_geometry UoF

5.1.2.1 SHIP_CURVE

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: {/ROOT_CLASS(compound_representation_item, 'ship curve')/}

5.1.2.1.1 curve_class

AIM element: representation_item.name
Source: ISO 10303-43
Reference path: compound_representation_item <=
representation_item
{(representation_item.name = 'buttock line')
(representation_item.name = 'centreline')
(representation_item.name = 'camber')
(representation_item.name = 'sheer')
(representation_item.name = 'flat of bottom')
(representation_item.name = 'flat of side')
(representation_item.name = 'intersection line')
(representation_item.name = 'station line')
(representation_item.name = 'bounding line')
(representation_item.name = 'rise of floor')
(representation_item.name = 'trace line')
(representation_item.name = 'waterline')
(representation_item.name = 'unspecified')}

5.1.2.1.2 side_condition

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.135
Reference path: /COMPOUND('side condition')/
descriptive_representation_item
{(descriptive_representation_item.description = 'knuckle')
(descriptive_representation_item.description = 'smooth')
(descriptive_representation_item.description = 'tangent')
(descriptive_representation_item.description = 'unspecified')}

5.1.2.1.3 ship_curve to curve (as curve_shape)

AIM element: PATH
Rules: 5.2.4.114, 5.2.4.135
Reference path: /COMPOUND('curve shape')/
geometric_representation_item =>
edge =>
(edge_curve)
(oriented_edge)

5.1.2.2 SHIP_CURVE_WITH_SPACING_POSITION

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: {/ROOT_CLASS(compound_representation_item, 'ship curve with spacing position')/}

5.1.2.2.1 curve_class

NOTE Attribute inherited from supertype Ship_curve (see 5.1.2.1).

AIM element: PATH
Reference path: /SUPERTYPE(Ship_curve)/ (see 5.1.2.1.1)

5.1.2.2.2 side_condition

NOTE Attribute inherited from supertype Ship_curve (see 5.1.2.1).

AIM element: PATH
Reference path: /SUPERTYPE(Ship_curve)/ (see 5.1.2.1.2)

5.1.2.2.3 ship_curve_with_spacing_position to curve (as curve_shape)

NOTE Attribute inherited from supertype Ship_curve (see 5.1.2.1).

AIM element: PATH
Reference path: /SUPERTYPE(Ship_curve)/ (see 5.1.2.1.3)

5.1.2.2.4 ship_curve_with_spacing_position to spacing_position (as location)

AIM element: PATH
Rules: 5.2.4.137
Reference path: /COMPOUND('location')/
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘spacing position’)/}

5.1.2.3 SHIP_POINT

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: {/ROOT_CLASS(compound_representation_item, 'ship point')/}

5.1.2.3.1 point_class

AIM element: representation_item.name
Source: ISO 10303-43
Reference path: compound_representation_item <=
representation_item
{(representation_item.name = 'ordinary')
(representation_item.name = 'tangent')
(representation_item.name = 'knuckle')
(representation_item.name = 'unspecified')}

5.1.2.3.2 ship_point to point (as point_shape)

AIM element: PATH
Rules: 5.2.4.114, 5.2.4.141
Reference path: /COMPOUND('point shape')/
geometric_representation_item =>
vertex_point

5.1.2.4 SHIP_SURFACE

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: {/ROOT_CLASS(compound_representation_item, 'ship surface')/}

5.1.2.4.1 surface_class

AIM element: representation_item.name
Source: ISO 10303-43
Reference path: compound_representation_item <=
representation_item
{(representation_item.name = 'accomodation area')
(representation_item.name = 'accomodation deck')
(representation_item.name = 'aft ship')
(representation_item.name = 'blending surface')
(representation_item.name = 'bottom')
(representation_item.name = 'bracket')
(representation_item.name = 'bulkhead')
(representation_item.name = 'cargo area')
(representation_item.name = 'collision bulkhead')
(representation_item.name = 'cross tie')
(representation_item.name = 'deck')

(representation_item.name = 'deck beam')
(representation_item.name = 'deck house')
(representation_item.name = 'deck in superstructure')
(representation_item.name = 'double bottom')
(representation_item.name = 'double shell')
(representation_item.name = 'duct keel')
(representation_item.name = 'engine area')
(representation_item.name = 'engine foundation')
(representation_item.name = 'external surface')
(representation_item.name = 'floor')
(representation_item.name = 'fore ship')
(representation_item.name = 'frame')
(representation_item.name = 'girder')
(representation_item.name = 'hatch cover')
(representation_item.name = 'hatchway coaming')
(representation_item.name = 'hatchway endcoaming')
(representation_item.name = 'hatchway sidecoaming')
(representation_item.name = 'hold bulkhead')
(representation_item.name = 'hopper')
(representation_item.name = 'inner bottom')
(representation_item.name = 'inner shell')
(representation_item.name = 'internal surface')
(representation_item.name = 'keel')
(representation_item.name = 'longitudinal bulkhead')
(representation_item.name = 'longitudinal girder')
(representation_item.name = 'lower boom')
(representation_item.name = 'machinery casing')
(representation_item.name = 'main deck')
(representation_item.name = 'mid ship')
(representation_item.name = 'navigation deck')
(representation_item.name = 'outer shell')
(representation_item.name = 'platform deck')
(representation_item.name = 'plating')
(representation_item.name = 'sheer strake')
(representation_item.name = 'ship structure')
(representation_item.name = 'stern frame')
(representation_item.name = 'stool')
(representation_item.name = 'strength bulkhead')
(representation_item.name = 'strength deck')
(representation_item.name = 'stringer')
(representation_item.name = 'superstructure')
(representation_item.name = 'superstructure aft bulkhead')
(representation_item.name = 'superstructure front bulkhead')
(representation_item.name = 'superstructure side bulkhead')
(representation_item.name = 'tank bottom')
(representation_item.name = 'tank bulkhead')
(representation_item.name = 'tank side')
(representation_item.name = 'tank top')
(representation_item.name = 'transom')
(representation_item.name = 'transversal bulkhead')

(representation_item.name = 'transverse floor')
(representation_item.name = 'transverse web frame')
(representation_item.name = 'upper boom')
(representation_item.name = 'user defined')
(representation_item.name = 'vertical web frame')
(representation_item.name = 'wall')
(representation_item.name = 'wash bulkhead')
(representation_item.name = 'weather deck')
(representation_item.name = 'web frame')
(representation_item.name = 'wing bulkhead')}

5.1.2.4.2 ship_surface to surface (as surface_shape)

AIM element: PATH
Rules: 5.2.4.114, 5.2.4.142
Reference path: /COMPOUND('surface shape')/
topological_representation_item =>
face=>
(face_surface)
(subface)
(oriented_face)

5.1.3 configuration_management UoF

5.1.3.1 ALTERNATIVE_VERSION_RELATIONSHIP

AIM element: identification_assignment_relationship
Source: ISO 10303-41
Rules: 5.2.4.5, 5.2.4.6
Reference path: /ROOT_CLASS(identification_assignment_relationship, 'alternative version relationship')/

5.1.3.1.1 reason

AIM element: identification_assignment_relationship.description
Source: ISO 10303-41
Rules: 5.2.4.4

5.1.3.1.2 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)/)

5.1.3.1.3 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)/)

5.1.3.2 APPROVAL_EVENT

AIM element: approval
Source: ISO 10303-41
Reference path: {[/CLASS(approval, 'approval event', 'event')/]
[/ROOT_CLASS(approval, 'event')/]}

5.1.3.2.1 caused_by

NOTE Attribute inherited from supertype Event (see 5.1.3.12).

AIM element: person_and_organization
Source: ISO 10303-41
Rules: 5.2.4.7
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

5.1.3.2.2 caused_when

NOTE Attribute inherited from supertype Event (see 5.1.3.12).

AIM element: approval_date_time.date_time
Source: ISO 10303-41
Rules: 5.2.4.8
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)}

5.1.3.2.3 description

NOTE Attribute inherited from supertype Event (see 5.1.3.12).

AIM element: approval.level
Source: ISO 10303-41

5.1.3.2.4 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')}

5.1.3.2.5 user_defined_result

AIM element: approval_status.name
Source: ISO 10303-41
Reference path: approval
approval.status ->
approval_status
approval_status.name

5.1.3.2.6 approval_event to approval_history (as approval_reference)

NOTE Attribute is inverse relationship for Approval_history attribute approvals (see 5.1.3.3.2).

AIM element: PATH
Rules: 5.2.4.53
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')/}

5.1.3.3 APPROVAL_HISTORY

AIM element: group
Source: ISO 10303-41
Reference path: /ROOT_CLASS(group, 'approval history')/

5.1.3.3.1 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')}

5.1.3.3.2 approval_history to approval_event (as approvals)

AIM element: PATH
Rules: 5.2.4.10, 5.2.4.149, 5.2.4.11
Reference path: group <-
/GROUPS(approval, 'approvals')/
approval

5.1.3.3.3 approval_history to definition (as subject)

AIM element: PATH
Rules: 5.2.4.9
Reference path: group <-
/GROUPS(approval, 'approvals')/
approval <-
(/APPROVES(product_definition, 'subject')/
product_definition
{/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’)/})

5.1.3.4 CHANGE

AIM element: action
Source: ISO 10303-41
Reference path: {[/CLASS(action,’change’,’item’)/]
[/CLASS(action,’item’,’definable object’)/]
[/ROOT_CLASS(action, ‘definable object’)/]}

5.1.3.4.1 description

NOTE Attribute inherited from supertype Item (see 5.1.8.3).

AIM element: action.description
Source: ISO 10303-41

5.1.3.4.2 name

NOTE Attribute inherited from supertype Item (see 5.1.8.3).

AIM element: action.name
Source: ISO 10303-41

5.1.3.4.3 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_classification ->
group

5.1.3.4.4 change to external_reference (as documentation)

NOTE Attribute inherited from supertype Item (see 5.1.8.3).

#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’)/}

5.1.3.4.5 change to global_id (as id)

NOTE Attribute inherited from supertype Item (see 5.1.8.3), which inherits from supertype Definition (see 5.1.4.1).

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.3
Reference path: action
identification_item = action
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.3.4.6 change to ship (as ship_context)

NOTE Attribute inherited from supertype Item (see 5.1.8.3).

AIM element:	PATH
Source:	ISO 10303-41
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')/}

5.1.3.5 CHANGE_DEFINITION

	#1: Change_definition is a change_request #2: Change_definition is a change_plan #3: Change_definition is a change_realization
AIM element:	#1: (version_action_request) #2: (action_request_solution) #3: (executed_action)
Source:	ISO 10303-41 ISO 10303-41 ISO 10303-41

5.1.3.5.1 author

AIM element:	#1: /SUBTYPE(Change_request)/ (see 5.1.3.9.2) #2: /SUBTYPE(Change_plan)/ (see 5.1.3.7.1) #3: /SUBTYPE(Change_realization)/ (see 5.1.3.8.1)

5.1.3.5.2 date_time

AIM element:	#1: /SUBTYPE(Change_request)/ (see 5.1.3.9.3) #2: /SUBTYPE(Change_plan)/ (see 5.1.3.7.2) #3: /SUBTYPE(Change_realization)/ (see 5.1.3.8.2)

5.1.3.5.3 description

AIM element:	#1: /SUBTYPE(Change_request)/ (see 5.1.3.9.4) #2: /SUBTYPE(Change_plan)/ (see 5.1.3.7.3) #3: /SUBTYPE(Change_realization)/ (see 5.1.3.8.3)

5.1.3.5.4 version_id

AIM element: #1: /SUBTYPE(Change_request)/ (see 5.1.3.9.8)
#2: /SUBTYPE(Change_plan)/ (see 5.1.3.7.4)
#3: /SUBTYPE(Change_realization)/ see(5.1.3.8.4)

5.1.3.5.5 change_definition to change (as defined_for)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: #1: /SUBTYPE(Change_request)/ (see 5.1.3.9.9)
#2: /SUBTYPE(Change_plan)/ (see 5.1.3.7.7)
#3: /SUBTYPE(Change_realization)/ (see 5.1.3.8.6)

5.1.3.5.6 change_definition to global_id (as id)

AIM element: #1: /SUBTYPE(Change_request)/ (see 5.1.3.9.10)
#2: /SUBTYPE(Change_plan)/ (see 5.1.3.7.8)
#3: /SUBTYPE(Change_realization)/ (see 5.1.3.8.7)

5.1.3.6 CHANGE_IMPACT

AIM element: applied_action_request_assignment
Source: ISO 10303-216
Reference path: {/ROOT_CLASS(applied_action_request_assignment, ‘change impact’)/}

5.1.3.6.1 change_impact to versionable_object_change_event (as impact)

AIM element: PATH
Rules: 5.2.4.26
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’)/}

5.1.3.7 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’)/]}

5.1.3.7.1 author

NOTE Attribute inherited from supertype Change_definition (see 5.1.3.5).

AIM element: applied_person_and_organization_assignment.assigned_person_and_-organization
Source: ISO 10303-216
Rules: 5.2.4.12
Reference path: /PERS_ORG_ASSGN(action_request_solution, 'author')/

5.1.3.7.2 date_time

NOTE Attribute inherited from supertype Change_definition (see 5.1.3.5).

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-216
Rules: 5.2.4.1
Reference path: /DAT_TIME_ASSGN(action_request_solution,'date time')/

5.1.3.7.3 description

NOTE Attribute inherited from supertype Item (see 5.1.8.3).

AIM element: action_request_solution.description
Source: ISO 10303-41
Reference path: /DESCRIPTION_ASSGN(action_request_solution)/

5.1.3.7.4 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17) through supetype Change_definition (see 5.1.3.5), which inherits from supertype Definition (see 5.1.4.1).

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-216
Rules: 5.2.4.163
Reference path: /VERSION_ID(action_request_solution)/

5.1.3.7.5 change_plan to check (as checks)

AIM element: PATH
Reference path: action_request_solution
action_item = action_request_solution
action_item <-
applied_action_assignment.items[i]
applied_action_assignment <=
action_assignment
action_assignment.assigned_action ->
action
{/CLASS_ID(action, ‘check’)}

5.1.3.7.6 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’)/}

5.1.3.7.7 change_plan to change (as defined_for)

NOTE Attribute inherited from supertype Change_definition (see 5.1.3.5).

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‘)/}

5.1.3.7.8 change_plan to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Change_definition (see 5.1.3.5).

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.2
Reference path: action_request_solution
identification_item = action_request_solution
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.3.7.9 change_plan to change_impact (as planned_impact)

AIM element: PATH
Rules: 5.2.4.27
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’)/}

5.1.3.8 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’)/]}

5.1.3.8.1 author

NOTE Attribute inherited from supertype Change_definition (see 5.1.3.5).

AIM element: applied_person_and_organization_assignment.assigned_person_and_-organization
Source: ISO 10303-216
Rules: 5.2.4.13
Reference path: /PERS_ORG_ASSGN(executed_action, 'author')/

5.1.3.8.2 date_time

NOTE Attribute inherited from supertype Change_definition (see 5.1.3.5).

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-216
Rules: 5.2.4.36
Reference path: /DAT_TIME_ASSGN(executed_action,'date time')/

5.1.3.8.3 description

NOTE Attribute inherited from supertype Versionable_object (see 5.1.8.3).

AIM element: executed_action.description
Source: ISO 10303-41

5.1.3.8.4 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17) through supetype Change_definition (see 5.1.3.5), which inherits from supertype Definition (see 5.1.4.1).

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-216
Rules: 5.2.4.163
Reference path: /VERSION_ID(executed_action)/

5.1.3.8.5 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’)/}

5.1.3.8.6 change_realization to change (as defined_for)

NOTE Attribute inherited from supertype Change_definition (see 5.1.3.5).

AIM element: PATH
Reference path: executed_action <=
action <-
action_relationship.related_action
action_relationship
action_relationship.relating_action ->
action
{/CLASS_ID(action, ‘change‘)/}

5.1.3.8.7 change_realization to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Change_definition (see 5.1.3.5).

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.41
Reference path: executed_action
identification_item = executed_action
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.3.8.8 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’)/}

5.1.3.8.9 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’)/}

5.1.3.9 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’)/]}

5.1.3.9.1 addressee

AIM element: applied_person_and_organization_assignment.assigned_person_and_-organization
Source: ISO 10303-216
Reference path: /PERS_ORG_ASSGN(versioned_action_request, 'addressee')/

5.1.3.9.2 author

NOTE Attribute inherited from supertype Change_definition (see 5.1.3.5).

AIM element: applied_person_and_organization_assignment.assigned_person_and_-organization
Source: ISO 10303-216
Rules: 5.2.4.14
Reference path: /PERS_ORG_ASSGN(versioned_action_request, 'author')/

5.1.3.9.3 date_time

NOTE Attribute inherited from supertype Change_definition (see 5.1.3.5).

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-216
Rules: 5.2.4.35
Reference path: /DAT_TIME_ASSGN(versioned_action_request,'date time')/

5.1.3.9.4 description

NOTE Attribute inherited from supertype Versionable_object (see 5.1.8.3).

AIM element: versioned_action_request.description
Source: ISO 10303-41

5.1.3.9.5 initiator

AIM element: applied_person_and_organization_assignment.assigned_person_and_-organization
Source: ISO 10303-216
Rules: 5.2.4.50
Reference path: /PERS_ORG_ASSGN(versioned_action_request, 'initiator')/

5.1.3.9.6 problem

AIM element: versioned_action_request.purpose
Source: ISO 10303-41

5.1.3.9.7 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)/

5.1.3.9.8 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17) through supetype Change_definition (see 5.1.3.5), which inherits from supertype Definition (see 5.1.4.1).

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-216
Rules: 5.2.4.163
Reference path: /VERSION_ID(versioned_action_request)/

5.1.3.9.9 change_request to change (as defined_for)

NOTE Attribute inherited from supertype Change_definition (see 5.1.3.5).

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‘)/}

5.1.3.9.10 change_request to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Change_definition (see 5.1.3.5).

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.164
Reference path: versioned_action_request
identification_item = versioned_action_request
identification_item <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.3.9.11 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’)/}

5.1.3.10 CHECK

AIM element: action
Source: ISO 10303-41
Reference path: {[/CLASS(action, ‘check’, ‘event’)/]
[/ROOT_CLASS(action, ‘event’)/]}

5.1.3.10.1 caused_by

NOTE Attribute inherited from supertype Event (see 5.1.3.12).

AIM element: applied_person_and_organization_assignment.assigned_person_and_-
organization
Source: ISO 10303-216
Rules: 5.2.4.15
Reference path: /PERS_ORG_ASSGN(action, 'caused by')/

5.1.3.10.2 caused_when

NOTE Attribute inherited from supertype Event (see 5.1.3.12).

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-216
Rules: 5.2.4.20
Reference path: /DAT_TIME_ASSGN(action, 'caused when')/

5.1.3.10.3 description

NOTE Attribute inherited from supertype Event (see 5.1.3.12).

AIM element: action.description
Source: ISO 10303-41

5.1.3.11 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’)/]}

5.1.3.11.1 category

AIM element: action.name
Source: ISO 10303-41

5.1.3.11.2 caused_by

NOTE Attribute inherited from supertype Event (see 5.1.3.12), through supetype Versionable_object_change_event (see 5.1.3.20).

AIM element: applied_person_and_organization_assignment.assigned_person_and_-organization
Source: ISO 10303-41
Rules: 5.2.4.16
Reference path: /PERS_ORG_ASSGN(action, 'caused by')/

5.1.3.11.3 caused_when

NOTE Attribute inherited from supertype Event (see 5.1.3.12), through supetype Versionable_object_change_event (see 5.1.3.20).

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-216
Rules: 5.2.4.21
Reference path: /DAT_TIME_ASSGN(action,'caused when')/

5.1.3.11.4 description

NOTE Attribute inherited from supertype Event (see 5.1.3.12), through supetype Versionable_object_change_event (see 5.1.3.20).

AIM element: action.description
Source: ISO 10303-41
Rules: 5.2.4.40

5.1.3.11.5 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(product_definition)/)
(/RELATE_ACT_2_VO(property_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(document)/)

5.1.3.12 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

5.1.3.12.1 caused_by

AIM element: /SUBTYPE(Versionable_object_change_event)/ (see 5.1.3.20)
/SUBTYPE(Approval_event)/ (see 5.1.3.2)
/SUBTYPE(Check)/ (see 5.1.3.10)

5.1.3.12.2 caused_when

AIM element: /SUBTYPE(Versionable_object_change_event)/ (see 5.1.3.20)
/SUBTYPE(Approval_event)/ (see 5.1.3.2)
/SUBTYPE(Check)/ (see 5.1.3.10)

5.1.3.12.3 description

AIM element: /SUBTYPE(Versionable_object_change_event)/ (see 5.1.3.20)
/SUBTYPE(Approval_event)/ (see 5.1.3.2)
/SUBTYPE(Check)/ (see 5.1.3.10)

5.1.3.13 REVISION

AIM element: group
Source: ISO 10303-41
Rules: 5.2.4.52
Reference path: {[/CLASS(group, ‘revision’, ‘versionable object’)/]
[/ROOT_CLASS(group, ‘versionable object’)/]}

5.1.3.13.1 description

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: group.description
Source: ISO 10303-41
Rules: 5.2.4.133

5.1.3.13.2 name

AIM element: group.name
Source: ISO 10303-41

5.1.3.13.3 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-216
Rules: 5.2.4.163
Reference path: /VERSION_ID(group)/

5.1.3.13.4 revision to versionable_object (as members)

AIM element: PATH
Rules: 5.2.4.52
Reference path: group <-
(/RELATE_GROUP_2_VO(product_definition, 'members')/)
(/RELATE_GROUP_2_VO(property_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(document, 'members')/)

5.1.3.14 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’)/]}

5.1.3.14.1 description

NOTE Attribute inherited from supertype Revision (see 5.1.3.13).

AIM element: group.description
Source: ISO 10303-41
Rules: 5.2.4.133

5.1.3.14.2 name

NOTE Attribute inherited from supertype Revision (see 5.1.3.13).

AIM element: group.name
Source: ISO 10303-41

5.1.3.14.3 version_id

NOTE Attribute inherited from supertype Revision (see 5.1.3.13), which inherits from supertype Versionable_object (see 5.1.9.17).

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-216
Rules: 5.2.4.163
Reference path: /VERSION_ID(group)/

5.1.3.14.4 revision_with_context to definable_object (as context_of_revision)

AIM element: PATH
Rules: 5.2.4.134
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’) /)

5.1.3.14.5 revision_with_context to versionable_object (as members)

NOTE Attribute inherited from supertype Revision (see 5.1.3.13).

AIM element: /SUPERTYPE(revision)/ (see 5.1.3.13)

5.1.3.15 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’)/]}

5.1.3.15.1 caused_by

NOTE Attribute inherited from supertype Event (see 5.1.3.12), through supetype Versionable_object_change_event (see 5.1.3.20).

AIM element: applied_person_and_organization_assignment.assigned_person_and_-organization
Source: ISO 10303-216
Rules: 5.2.4.17
Reference path: /PERS_ORG_ASSGN(action, 'caused by')/

5.1.3.15.2 caused_when

NOTE Attribute inherited from supertype Event (see 5.1.3.12), through supetype Versionable_object_change_event (see 5.1.3.20).

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-216
Rules: 5.2.4.22
Reference path: /DAT_TIME_ASSGN(action, 'caused when')/

5.1.3.15.3 description

NOTE Attribute inherited from supertype Event (see 5.1.3.12), through supetype Versionable_object_change_event (see 5.1.3.20).

AIM element: action.description
Source: ISO 10303-41
Rules: 5.2.4.153

5.1.3.15.4 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(product_definition)/)
(/RELATE_ACT_2_VO(property_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(document)/)

5.1.3.15.5 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(product_definition)/)
(/RELATE_ACT_2_VO(property_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(document)/)

5.1.3.16 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')/]}

5.1.3.16.1 caused_by

NOTE Attribute inherited from supertype Event (see 5.1.3.12), through supetype Versionable_object_change_event (see 5.1.3.20).

AIM element: applied_person_and_organization_assignment.assigned_person_and_-organization
Source: ISO 10303-216
Rules: 5.2.4.18
Reference path: /PERS_ORG_ASSGN(action, 'caused by')/

5.1.3.16.2 caused_when

NOTE Attribute inherited from supertype Event (see 5.1.3.12), through supetype Versionable_object_change_event (see 5.1.3.20).

AIM element: applied_date_and_time_assignment.assigned_date_and_time
Source: ISO 10303-216
Rules: 5.2.4.24
Reference path: /DAT_TIME_ASSGN(action, ‘caused when')/

5.1.3.16.3 description

NOTE Attribute inherited from supertype Event (see 5.1.3.12), through supetype Versionable_object_change_event (see 5.1.3.20).

AIM element: action.description
Source: ISO 10303-41
Rules: 5.2.4.154

5.1.3.16.4 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(product_definition)/)
(/RELATE_ACT_2_VO(property_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(document)/)

5.1.3.17 VERSION_HISTORY

AIM element: group
Source: ISO 10303-41
Rules: 5.2.4.158
Reference path: /ROOT_CLASS(group, 'version history')/

5.1.3.17.1 version_history to versionable_object (as current_version)

AIM element: PATH
Rules: 5.2.4.157, 5.2.4.155
Reference path: group <-
(/RELATE_GROUP_2_VO(product_definition, 'versions')/)
(/RELATE_GROUP_2_VO(property_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(document, 'versions')/)

5.1.3.17.2 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')/}

5.1.3.17.3 version_history to versionable_object (as versions)

AIM element: PATH
Rules: 5.2.4.165
Reference path: group <-
(/RELATE_GROUP_2_VO(product_definition, 'versions')/)
(/RELATE_GROUP_2_VO(property_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(document, 'versions')/)

5.1.3.18 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’)/]}

5.1.3.18.1 caused_by

NOTE Attribute inherited from supertype Event (see 5.1.3.12), through supetype Versionable_object_change_event (see 5.1.3.20).

AIM element:	applied_person_and_organization_assignment.assigned_person_and_-organization
Source:	ISO 10303-216
Rules:	5.2.4.19
Reference path:	/PERS_ORG_ASSGN(action, 'caused by')/

5.1.3.18.2 caused_when

NOTE Attribute inherited from supertype Event (see 5.1.3.12), through supetype Versionable_object_change_event (see 5.1.3.20).

AIM element:	applied_date_and_time_assignment.assigned_date_and_time
Source:	ISO 10303-216
Rules:	5.2.4.23
Reference path:	/DAT_TIME_ASSGN(action,'caused when')/

5.1.3.18.3 description

NOTE Attribute inherited from supertype Event (see 5.1.3.12), through supetype Versionable_object_change_event (see 5.1.3.20).

AIM element:	action.description
Source:	ISO 10303-41
Rules:	5.2.4.159

5.1.3.18.4 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(product_definition)/) (/RELATE_ACT_2_VO(property_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(document)/)

5.1.3.18.5 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(product_definition)/)
(/RELATE_ACT_2_VO(property_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(document)/)

5.1.3.19 VERSION_RELATIONSHIP

AIM element: identification_assignment_relationship
Source: ISO 10303-41
Rules: 5.2.4.160, 5.2.4.162
Reference path: /ROOT_CLASS(identification_assignment_relationship, 'version relationship')/

5.1.3.19.1 reason

AIM element: identification_assignment_relationship.description
Source: ISO 10303-41
Rules: 5.2.4.161

5.1.3.19.2 version_relationship to versionable_object (as predecessor)

AIM element: PATH
Reference path: identification_assignment_relationship
identification_assignment_relationship.related_assignment ->
(/RELATE_ID_2_VO(product_definition)/)
(/RELATE_ID_2_VO(property_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(document)/)
(/RELATE_ID_2_VO(group)/)

5.1.3.19.3 version_relationship to versionable_object (as successor)

AIM element: PATH
Reference path: identification_assignment_relationship
identification_assignment_relationship.relating_assignment ->
(/RELATE_ID_2_VO(product_definition)/)
(/RELATE_ID_2_VO(property_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(document)/)
(/RELATE_ID_2_VO(group)/)

5.1.3.20 VERSIONABLE_OBJECT_CHANGE_EVENT

#1: Versionable_object_change_event is an envisaged_version_creation
#2: Versionable_object_change_event is a version_creation
#3: Versionable_object_change_event is a version_modification
#4: Versionable_object_change_event is a version_deletion

AIM element: #1: (action)
#2: (action)
#3: (action)
#4: (action)

Source: ISO 10303-41

5.1.3.20.1 caused_by

NOTE Attribute inherited from supertype Event (see 5.1.3.12).

AIM element: #1: /SUBTYPE(Envisaged_version_creation)/ (see 5.1.3.11.2)
#2: /SUBTYPE(Version_creation)/ (see 5.1.3.15.1)
#3: /SUBTYPE(Version_modification)/ (see 5.1.3.18.1)
#4: /SUBTYPE(Version_deletion)/ (see 5.1.3.16.1)

5.1.3.20.2 caused_when

NOTE Attribute inherited from supertype Event (see 5.1.3.12).

AIM element: #1: /SUBTYPE(Envisaged_version_creation)/ (see 5.1.3.11.3)
#2: /SUBTYPE(Version_creation)/ (see 5.1.3.15.2)
#3: /SUBTYPE(Version_modification)/ (see 5.1.3.18.2)
#4: /SUBTYPE(Version_deletion)/ (see 5.1.3.16.2)

5.1.3.20.3 description

NOTE Attribute inherited from supertype Event (see 5.1.3.12).

AIM element:
#1: /SUBTYPE(Envisaged_version_creation)/ (see 5.1.3.11.4)
#2: /SUBTYPE(Version_creation)/ (see 5.1.3.15.3)
#3: /SUBTYPE(Version_modification)/ (see 5.1.3.18.3)
#4: /SUBTYPE(Version_deletion)/ (see 5.1.3.16.3)

5.1.4 definitions UoF

5.1.4.1 DEFINITION

#1: definition is a change_definition
#2: definition is a design_definition
#3: definition is a functional_definition
#4: definition is a general_characteristics_definition
#5: definition is a local_coordinate_system
#6: definition is a moulded_form_characteristics_definition
#7: definition is a spacing_table

AIM element:
#1: /SUBTYPE(change_definition)/ (see 5.1.3.5)
#2: /SUBTYPE(design_definition)/ (see 5.1.4.2)
#3: /SUBTYPE(functional_definition)/ (see 5.1.4.3)
#4: /SUBTYPE(general_characteristics_definition)/ (see 5.1.4.4)
#5: /SUBTYPE(local_co_ordinate_system)/ (see 5.1.9.4)
#6: /SUBTYPE(moulded_form_characteristics_definition)/ (see 5.1.11.7)
#7: /SUBTYPE(spacing_table)/ (see 5.1.9.10)

5.1.4.1.1 description

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element:
#1: /SUBTYPE(change_definition)/ (see 5.1.3.5.3)
#2: /SUBTYPE(design_definition)/ (see 5.1.4.2.1)
#3: /SUBTYPE(functional_definition)/ (see 5.1.4.3.1)
#4: /SUBTYPE(general_characteristics_definition)/ (see 5.1.4.4.1)
#5: /SUBTYPE(local_co_ordinate_system)/ (see 5.1.9.4.1)
#6: /SUBTYPE(moulded_form_characteristics_definition)/ (see 5.1.11.7.1)
#7: /SUBTYPE(spacing_table)/ (see 5.1.9.10.1)

5.1.4.1.2 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element:
#1: /SUBTYPE(change_definition)/ (see 5.1.3.5.4)
#2: /SUBTYPE(design_definition)/ (see 5.1.4.2.3)
#3: /SUBTYPE(functional_definition)/ (see 5.1.4.3.2)
#4: /SUBTYPE(general_characteristics_definition)/ (see 5.1.4.4.2)

#5: /SUBTYPE(local_co_ordinate_system)/ (see 5.1.9.4.2)
#6: /SUBTYPE(moulded_form_characteristics_definition)/ (see 5.1.11.7.4)
#7: /SUBTYPE(spacing_table)/ (see 5.1.9.10.3)

5.1.4.1.3 definition to definable_object (as defined_for)

AIM element:
#1: /SUBTYPE(change_definition)/ (see 5.1.3.5.5)
#2: /SUBTYPE(design_definition)/ (see 5.1.4.2.4)
#3: /SUBTYPE(functional_definition)/ (see 5.1.4.3.4)
#4: /SUBTYPE(general_characteristics_definition)/ (see 5.1.4.4.3)
#5: /SUBTYPE(local_co_ordinate_system)/ (see 5.1.9.4.3)
#6: /SUBTYPE(moulded_form_characteristics_definition)/ (see 5.1.11.7.5)
#7: /SUBTYPE(spacing_table)/ (see 5.1.9.10.4)

5.1.4.1.4 definition to global_id (as id)

AIM element:
#1: /SUBTYPE(change_definition)/ (see 5.1.3.5.6)
#2: /SUBTYPE(design_definition)/ (see 5.1.4.2.5)
#3: /SUBTYPE(functional_definition)/ (see 5.1.4.3.5)
#4: /SUBTYPE(general_characteristics_definition)/ (see 5.1.4.4.4)
#5: /SUBTYPE(local_co_ordinate_system)/ (see 5.1.9.4.4)
#6: /SUBTYPE(moulded_form_characteristics_definition)/ (see 5.1.11.7.6)
#7: /SUBTYPE(spacing_table)/ (see 5.1.9.10.5)

5.1.4.1.5 definition to derived_unit (as local_units)

AIM element:
#2: /SUBTYPE(design_definition)/ (see 5.1.4.2.6)
#4: /SUBTYPE(general_characteristics_definition)/ (see 5.1.4.4.5)
#5: /SUBTYPE(local_co_ordinate_system)/ (see 5.1.9.4.5)
#6: /SUBTYPE(moulded_form_characteristics_definition)/ (see 5.1.11.7.7)
#7: /SUBTYPE(spacing_table)/ (see 5.1.9.10.6)

5.1.4.1.6 definition to named_unit (as local_units)

AIM element:
#2: /SUBTYPE(design_definition)/ (see 5.1.4.2.7)
#4: /SUBTYPE(general_characteristics_definition)/ (see 5.1.4.4.6)
#5: /SUBTYPE(local_co_ordinate_system)/ (see 5.1.9.4.6)
#6: /SUBTYPE(moulded_form_characteristics_definition)/ (see 5.1.11.7.8)
#7: /SUBTYPE(spacing_table)/ (see 5.1.9.10.7)

5.1.4.2 DESIGN_DEFINITION

#1: if design_definition is a hydrostatic_definition
#2: if design_definition is a moulded_form_design_definition
#3: if design_definition is a stability_definition

AIM element:
#1: /SUBTYPE(hydrostatic_definition)/ (see 5.1.7.4)
#2: /SUBTYPE(moulded_form_design_definition)/ (see 5.1.14.3)

#3: /SUBTYPE(stability_definition)/ (see 5.1.7.11)

5.1.4.2.1 description

AIM element: #1: /SUBTYPE(hydrostatic_definition)/ (see 5.1.7.4.1)
#2: /SUBTYPE(moulded_form_design_definition)/ (see 5.1.14.3.1)
#3: /SUBTYPE(stability_definition)/ (see 5.1.7.11.1)

5.1.4.2.2 representations

AIM element: #1: /SUBTYPE(hydrostatic_definition)/ (see 5.1.7.4.8)
#2: /SUBTYPE(moulded_form_design_definition)/ (see 5.1.14.3.15, 5.1.14.3.16)
#3: /SUBTYPE(stability_definition)/ (see 5.1.7.11.7)

5.1.4.2.3 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17) through Definition (see 5.1.4.1).

AIM element: #1: /SUBTYPE(hydrostatic_definition)/ (see 5.1.7.4.2)
#2: /SUBTYPE(moulded_form_design_definition)/ (see 5.1.14.3.2)
#3: /SUBTYPE(stability_definition)/ (see 5.1.7.11.2)

5.1.4.2.4 design_definition to definable_object (as defined_for)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: #1: /SUBTYPE(hydrostatic_definition)/ (see 5.1.7.4.3)
#2: /SUBTYPE(moulded_form_design_definition)/ (see 5.1.14.3.7)
#3: /SUBTYPE(stability_definition)/ (see 5.1.7.11.3)

5.1.4.2.5 design_definition to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: #1: /SUBTYPE(hydrostatic_definition)/ (see 5.1.7.4.5)
#2: /SUBTYPE(moulded_form_design_definition)/ (see 5.1.14.3.8)
#3: /SUBTYPE(stability_definition)/ (see 5.1.7.11.4)

5.1.4.2.6 design_definition to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: #1: /SUBTYPE(hydrostatic_definition)/ (see 5.1.7.4.6)
#2: /SUBTYPE(moulded_form_design_definition)/ (see 5.1.14.3.9)
#3: /SUBTYPE(stability_definition)/ (see 5.1.7.11.5)

5.1.4.2.7 design_definition to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: #1: /SUBTYPE(hydrostatic_definition)/ (see 5.1.7.4.7)
#2: /SUBTYPE(moulded_form_design_definition)/ (see 5.1.14.3.10)
#3: /SUBTYPE(stability_definition)/ (see 5.1.7.11.6)

5.1.4.3 FUNCTIONAL_DEFINITION

#1: functional definition is a moulded_form_functional_definition
#2: functional definition is a Shiptype

AIM element: #1: /SUBTYPE(moulded_form_functional_definition)/ (see 5.1.14.4)
#2: /SUBTYPE(Shiptype)/ (see 5.1.12.11)

5.1.4.3.1 description

AIM element: #1: /SUBTYPE(moulded_form_functional_definition)/ (see 5.1.14.4.1)
#2: /SUBTYPE(Shiptype)/ (see 5.1.12.11.1)

5.1.4.3.2 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17) through Definition (see 5.1.4.1).

AIM element: #1: /SUBTYPE(moulded_form_functional_definition)/ (see 5.1.14.4.4)
#2: /SUBTYPE(Shiptype)/ (see 5.1.12.11.3)

5.1.4.3.3 user_def_function

AIM element: #1: /SUBTYPE(moulded_form_functional_definition)/ (see 5.1.14.4.3)
#2: /SUBTYPE(Shiptype)/ (see 5.1.12.11.2)

5.1.4.3.4 functional_definition to definable_object (as defined_for)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: #1: /SUBTYPE(moulded_form_functional_definition)/ (see 5.1.14.4.5)
#2: /SUBTYPE(Shiptype)/ (see 5.1.12.11.4)

5.1.4.3.5 functional_definition to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: #1: /SUBTYPE(moulded_form_functional_definition)/ (see 5.1.14.4.6)
#2: /SUBTYPE(Shiptype)/ (see 5.1.12.11.5)

5.1.4.4 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 global_axis_placement
#4: if general_characteristics_definition is an owner_designation
#5: if general_characteristics_definition is a principal_characteristics
#6: if general_characteristics_definition is a ship_designation
#7: if general_characteristics_definition is a shipyard_designation
#8: if general_characteristics_definition is a ship_overall_dimensions

AIM element:
#1: /SUBTYPE(class_and_statutory_designation)/ (see 5.1.12.2)
#2: /SUBTYPE(class_parameters)/ (see 5.1.12.4)
#3: /SUBTYPE(global_axis_placement)/ (see 5.1.9.3)
#4: /SUBTYPE(owner_designation)/ (see 5.1.12.6)
#5: /SUBTYPE(principal_characteristics)/ (see 5.1.12.7)
#6: /SUBTYPE(ship_designation)/ (see 5.1.12.10)
#7: /SUBTYPE(shipyard_designation)/ (see 5.1.12.12)
#8: /SUBTYPE(ship_overall_dimensions)/ (see 5.1.11.11)

5.1.4.4.1 description

AIM element:
#1: /SUBTYPE(class_and_statutory_designation)/ (see 5.1.12.2.2)
#2: /SUBTYPE(class_parameters)/ (see 5.1.12.4.2)
#3: /SUBTYPE(global_axis_placement)/ (see 5.1.9.3.2)
#4: /SUBTYPE(owner_designation)/ (see 5.1.12.6.1)
#5: /SUBTYPE(principal_characteristics)/ (see 5.1.12.7.2)
#6: /SUBTYPE(ship_designation)/ (see 5.1.12.10.2)
#7: /SUBTYPE(shipyard_designation)/ (see 5.1.12.12.1)
#8: /SUBTYPE(ship_overall_dimensions)/ (see 5.1.11.11.1)

5.1.4.4.2 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17) through Definition (see 5.1.4.1).

AIM element:
#1: /SUBTYPE(class_and_statutory_designation)/ (see 5.1.12.2.3)
#2: /SUBTYPE(class_parameters)/ (see 5.1.12.4.8)
#3: /SUBTYPE(global_axis_placement)/ (see 5.1.9.3.4)
#4: /SUBTYPE(owner_designation)/ (see 5.1.12.6.6)
#5: /SUBTYPE(principal_characteristics)/ (see 5.1.12.7.12)
#6: /SUBTYPE(ship_designation)/ (see 5.1.12.10.7)
#7: /SUBTYPE(shipyard_designation)/ (see 5.1.12.12.6)
#8: /SUBTYPE(ship_overall_dimensions)/ (see 5.1.11.11.7)

5.1.4.4.3 general_characteristics_definition to ship (as defined_for)

AIM element:
#1: /SUBTYPE(class_and_statutory_designation)/ (see 5.1.12.2.4)
#2: /SUBTYPE(class_parameters)/ (see 5.1.12.4.9)
#3: /SUBTYPE(global_axis_placement)/ (see 5.1.9.3.5)
#4: /SUBTYPE(owner_designation)/ (see 5.1.12.6.7)
#5: /SUBTYPE(principal_characteristics)/ (see 5.1.12.7.13)
#6: /SUBTYPE(ship_designation)/ (see 5.1.12.10.8)
#7: /SUBTYPE(shipyard_designation)/ (see 5.1.12.12.7)
#8: /SUBTYPE(ship_overall_dimensions)/ (see 5.1.11.11.8)

5.1.4.4.4 general_characteristics_definition to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element:
#1: /SUBTYPE(class_and_statutory_designation)/ (see 5.1.12.2.5)
#2: /SUBTYPE(class_parameters)/ (see 5.1.12.4.10)
#3: /SUBTYPE(global_axis_placement)/ (see 5.1.9.3.6)
#4: /SUBTYPE(owner_designation)/ (see 5.1.12.6.8)
#5: /SUBTYPE(principal_characteristics)/ (see 5.1.12.7.14)
#6: /SUBTYPE(ship_designation)/ (see 5.1.12.10.9)
#7: /SUBTYPE(shipyard_designation)/ (see 5.1.12.12.8)
#8: /SUBTYPE(ship_overall_dimensions)/ (see 5.1.11.11.9)

5.1.4.4.5 general_characteristics_definition to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element:
#1: /SUBTYPE(class_and_statutory_designation)/ (see note)
#2: /SUBTYPE(class_parameters)/ (see 5.1.12.4.11)
#3: /SUBTYPE(global_axis_placement)/ (see 5.1.9.3.7)
#4: /SUBTYPE(owner_designation)/ (see note)
#5: /SUBTYPE(principal_characteristics)/ (see 5.1.12.7.15)
#6: /SUBTYPE(ship_designation)/ (see note)
#7: /SUBTYPE(shipyard_designation)/ (see note)
#8: /SUBTYPE(ship_overall_dimensions)/ (see 5.1.11.11.10)

NOTE For this subtype the attribute has been re-declared in the ARM to be a set of zero, which is interpreted as the attribute is not required

5.1.4.4.6 general_characteristics_definition to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: #1: /SUBTYPE(class_and_statutory_designation)/ (see note)
#2: /SUBTYPE(class_parameters)/ (see 5.1.12.4.12)
#3: /SUBTYPE(global_axis_placement)/ (see 5.1.9.3.8)
#4: /SUBTYPE(owner_designation)/ (see note)
#5: /SUBTYPE(principal_characteristics)/ (see 5.1.12.7.16)
#6: /SUBTYPE(ship_designation)/ (see note)
#7: /SUBTYPE(shipyard_designation)/ (see note)
#8: /SUBTYPE(ship_overall_dimensions)/ (see 5.1.11.11.11)

NOTE For this subtype the attribute has been re-declared in the ARM to be a set of zero, which is interpreted as the attribute is not required

5.1.5 external_references UoF

5.1.5.1 DOCUMENT

AIM element: document
Source: ISO 10303-41
Rules: 5.2.4.38
Reference path: {[/CLASS(document,‘document’, ‘versionable object’)/]
[/ROOT_CLASS(document,‘versionable object’)/]}

5.1.5.1.1 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 10303-216
#2: ISO 10303-216
#3: ISO 10303-216

Reference path: #1: /PERS_ASSGN(document,‘author’)/
#2: /ORG_ASSGN(document,‘author’)/
#3: /PERS_ORG_ASSGN(document,‘author’)/

5.1.5.1.2 description

AIM element: document.description
Source: ISO 10303-41

5.1.5.1.3 source_type

AIM element: document_representation_type.name
Source: ISO 10303-41
Rules: 5.2.4.37
Reference path: document <-
document_representation_type.represented_document
document_representation_type
document_representation_type.name

5.1.5.1.4 title

AIM element: document.name
Source: ISO 10303-41

5.1.5.1.5 version_id

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-216
Rules: 5.2.4.163
Reference path: /VERSION_ID(document)/

5.1.5.2 DOCUMENT_PORTION

AIM element: document_usage_constraint
Source: ISO 10303-41

5.1.5.2.1 element_type

AIM element: document_usage_constraint.subject_element
Source: ISO 10303-41

5.1.5.2.2 element_value

AIM element: document_usage_constraint.subject_element_value
Source: ISO 10303-41

5.1.5.2.3 document_portion to document (as source)

AIM element: document_usage_constraint.source
Reference path: document_usage_constraint
document_usage_constraint.source ->
document

5.1.5.3 DOCUMENT_REFERENCE

AIM element: document
Source: ISO 10303-41
Reference path: {/ROOT_CLASS(document,‘document reference’)/}

5.1.5.3.1 document_reference to document (as assigned_document)

AIM element: PATH
Reference path: document

5.1.5.3.2 document_reference to document_portion (as assigned_document)

AIM element: PATH
Reference path: document <-
document_usage_constraint.source
document_usage_constraint

5.1.5.4 DOCUMENT_REFERENCE_WITH_ADDRESS

AIM element: document
Source: ISO 10303-41
Rules: 5.2.4.39
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’)/]}

5.1.5.4.1 description

NOTE Attribute inherited from supertype External_reference (see 5.1.5.6).

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

5.1.5.4.2 document_reference_with_address to document (as assigned_-document)

NOTE Attribute inherited from supertype Document_reference (see 5.1.5.3).

AIM element: PATH
Reference path: document

5.1.5.4.3 document_reference_with_address to document_portion (as assigned_document)

NOTE Attribute inherited from supertype Document_reference (see 5.1.5.3).

AIM element: PATH
Reference path: document <-
document_usage_constraint.source
document_usage_constraint

5.1.5.4.4 document_reference_with_address to external_storage (as location)

NOTE Attribute inherited from supertype External_reference (see 5.1.5.6).

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'}]

5.1.5.4.5 document_reference_with_address to universal_resource_locator (as location)

NOTE Attribute inherited from supertype External_reference (see 5.1.5.6).

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'}]

5.1.5.5 EXTERNAL_INSTANCE_REFERENCE

AIM element: applied_external_identification_assignment
Source: ISO 10303-216
Reference path: applied_external_identification_assignment <=
external_identification_assignment <=
{/ID_ROLE('external instance reference')/}

5.1.5.5.1 entity_type

AIM element: external_source.source_id
Source: ISO 10303-41
Rules: 5.2.4.51
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}

5.1.5.5.2 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}

5.1.5.5.3 external_instance_reference to global_id (as target_guid)

AIM element: PATH
Rules: 5.2.4.42
Reference path: applied_external_identification_assignment <=
external_identification_assignment <=
identification_assignment
identification_assignment.assigned_id

5.1.5.6 EXTERNAL_REFERENCE

AIM element: applied_external_identification_assignment
Source: ISO 10303-216
Reference path: applied_external_identification_assignment <=
external_identification_assignment <=
identification_assignment
identification_assignment.role ->
identificaton_role
{identification_role.name ='external reference'}

5.1.5.6.1 description

AIM element: identification_role.description
Source: ISO 10303-41
Rules: 5.2.4.49
Reference path: applied_external_identification_assignment <=
external_identification_assignment <=
identification_assignment
identification_assignment.role ->
identificaton_role
identification_role.description

5.1.5.6.2 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')/}

5.1.5.6.3 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')/}

5.1.5.7 EXTERNAL_STORAGE

AIM element: external_source
Source: ISO 10303-41
Reference path: {/ROOT_CLASS(external_source, 'external storage')/}

5.1.5.7.1 location

AIM element: external_source.source_id
Source: ISO 10303-41
Reference path: external_source
external_source.source_id
{-> source_item
source_item = identifier}

5.1.5.8 UNIVERSAL_RESOURCE_LOCATOR

AIM element: external_source
Source: ISO 10303-41
Reference path: {/ROOT_CLASS(external_source, 'universal resource locator')/}

5.1.5.8.1 location

AIM element: external_source.source_id
Source: ISO 10303-41
Reference path: external_source
external_source.source_id
{-> source_item
source_item = identifier}

5.1.6 Hull_class_applicability UoF

5.1.6.1 HULL_APPLICABILITY

AIM element: serial_numbered_effectivity
Source: ISO 10303-41

5.1.6.1.1 end_hull

AIM element: serial_numbered_effectivity.effectivity_end_id
Source: ISO 10303-41

5.1.6.1.2 start_hull

AIM element: serial_numbered_effectivity.effectivity_start_id
Source: ISO 10303-41

5.1.6.1.3 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')/})

5.1.6.1.4 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’)/})

5.1.7 hydrostatics UoF

5.1.7.1 ADDITION_OF_MOULDED_FORM

AIM element: property_definition
Source: ISO 10303-41
Reference path: /ROOT_CLASS(property_definition, 'addition of moulded form')/

5.1.7.1.1 addition_of_moulded_form to moulded_form (as displacement_of_-moulded_form_to_add_or_subtract)

NOTE Attribute inherited from supertype Displacement_operation (see 5.1.7.2).

AIM element: /SUPERTYPE(Displacement_operation)/ (see 5.1.7.2)

5.1.7.2 DISPLACEMENT_OPERATION

#1: displacement_operation is an Addition_of_moulded_form
#2: displacement_operation is a Subtraction_of_moulded_form

AIM element: #1: /SUBTYPE(Addition_of_moulded_form)/ (see 5.1.7.1)
#2: /SUBTYPE(Subtraction_of_moulded_form)/ (see 5.1.7.15)

5.1.7.2.1 displacement_operation to moulded_form (as displacement_of_-moulded_form_to_add_or_subtract)

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/
{/CLASS_ID(product_definition, 'moulded_form')/}

5.1.7.3 FLOATING_POSITION

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: /ROOT_CLASS(compound_representation_item, 'floating position')/

5.1.7.3.1 angle_of_heel

AIM element: plane_angle_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.43
Reference path: /COMPOUND('angle of heel')/
value_representation_item
value_representation_item.value_component
{-> measure_value = plane_angle_measure}

5.1.7.3.2 angle_of_trim

AIM element: plane_angle_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.43
Reference path: /COMPOUND('angle of trim')/
value_representation_item
value_representation_item.value_component
{-> measure_value = plane_angle_measure}

5.1.7.3.3 breadth_of_waterline

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.43
Reference path: /COMPOUND('breadth of waterline')/
value_representation_item
value_representation_item.value_component
{-> measure_value = positive_length_measure}

5.1.7.3.4 draught_at_amidships

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.43
Reference path: /COMPOUND('draught at amidships')/
value_representation_item
value_representation_item.value_component
{-> measure_value = positive_length_measure}

5.1.7.3.5 length_of_waterline

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.43
Reference path: /COMPOUND('length of waterline')/
value_representation_item
value_representation_item.value_component
{-> measure_value = positive_length_measure}

5.1.7.3.6 moulded_form_displacement

AIM element: volume_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.43
Reference path: /COMPOUND('moulded form displacement')/
value_representation_item
value_representation_item.value_component
{-> measure_value = volume_measure}

5.1.7.4 HYDROSTATIC_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Reference path: {[/CLASS(property_definition, 'hydrostatic definition', 'design definition')/]
[/CLASS(property_definition, 'design definition', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.7.4.1 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.7.4.2 version_id

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17) through supetype Design_definition (see 5.1.4.2), which inherits from supertype Definition (see 5.1.4.1).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(property_definition)/

5.1.7.4.3 hydrostatic_definition to ship (as defined_for)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Design_definition (see 5.1.4.2).

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/
/PROD_DEF_PRODUCT/
{/CLASS_ID(product, 'ship')/}

5.1.7.4.4 hydrostatic_definition to displacement_operation (as displacement_changes)

AIM element: PATH
Reference path: property_definition <-
property_definition_relationship.related_property_definition
property_definition_relationship
{property_definition_relationship.name = 'displacement change'}
property_definition_relationship.relating_property_definition ->
property_definition
{(/CLASS_ID(property_definition, 'addition of moulded form')/)
(/CLASS_ID(property_definition, 'subtraction of moulded form')/)}

5.1.7.4.5 hydrostatic_definition to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Design_definition (see 5.1.4.2).

AIM element: PATH
Source: ISO 10303-41
Rules: 5.2.4.45, 5.2.4.89
Reference path: property_definition
identification_item = property_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.7.4.6 hydrostatic_definition to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Design_definition (see 5.1.4.2).

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.7.4.7 hydrostatic_definition to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Design_definition (see 5.1.4.2).

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.7.4.8 hydrostatic_definition to hydrostatic_table (as representations)

NOTE Attribute inherited from supertype Design_definition (see 5.1.4.2).

AIM element: PATH
Rules: 5.2.4.59
Reference path: /PROP_DEF_TO_REP/
{/CLASS_ID(representation, 'hydrostatic table')/}

5.1.7.5 HYDROSTATIC_POSITION_VALUE

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: /CLASS_ID(compound_representation_item, ‘centre location’)/

5.1.7.5.1 hydrostatic_position_value to centre_location (as position)

AIM element: IDENTICAL MAPPING

5.1.7.6 HYDROSTATIC_PROPERTIES_FOR_CONSTANT_FLOATING_-POSITION

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: /ROOT_CLASS(compound_representation_item, 'hydrostatic properties for constant floating position')/

5.1.7.6.1 hydrostatic_properties_for_constant_floating_position to floating_-position (as definition_of_floating_position)

AIM element: PATH
Rules: 5.2.4.47
Reference path: /COMPOUND('definition of floating position')/
compound_representation_item
{/CLASS_ID(compound_representation_item, 'floating position')/}

5.1.7.6.2 hydrostatic_properties_for_constant_floating_position to hydrostatic_property_value (as hydrostatic_property_values)

AIM element: PATH
Rules: 5.2.4.31
Reference path: /COMPOUND('hydrostatic property value')/
(compound_representation_item
{/CLASS_ID(compound_representation_item, ‘centre_location)/})
(value_representation_item)

5.1.7.6.3 hydrostatic_properties_for_constant_floating_position to hydrostatic_table (as related_hydrostatic_table)

NOTE attribute is inverse relationship for Hydrostatic_table attribute items (see 5.1.7.10.3).

AIM element: PATH
Reference path: compound_representation_item <=
representation_item <-
representation.items [i]
representation
{/CLASS_ID(representation, 'hydrostatic table')/}

5.1.7.7 HYDROSTATIC_PROPERTY

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: /ROOT_CLASS(compound_representation_item, 'hydrostatic property')/

5.1.7.7.1 property_measure

#1: property_measure is an area_measure
#2: property_measure is an inertia_moment_measure
#3: property_measure is a length_measure
#4: property_measure is a moment_measure
#5: property_measure is a centre_location

AIM element: #1: area_measure
#2: context_dependent_measure
#3: length_measure
#4: moment_measure
#5: compound_representation_item

Source: ISO 10303-43

Reference path: #1: /COMPOUND('property measure')/
value_representation_item
{value_representation_item.value_component ->
measure_value = area_measure}

#2: /COMPOUND('property measure')/
value_representation_item
{value_representation_item.value_component ->
measure_value = context_dependent_measure}

#3: /COMPOUND('property measure')/
value_representation_item
{value_representation_item.value_component ->
measure_value = length_measure}

#4: /COMPOUND('property measure')/
value_representation_item
{value_representation_item.value_component ->
measure_value = moment_measure}

#5: /COMPOUND('property measure')/
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘centre_location’)/}

5.1.7.7.2 property_type

AIM element: descriptive_representation_item
Source: ISO 10303-45
Rules: 5.2.4.48
Reference path: /COMPOUND('property type')/
descriptive_representation_item
{(descriptive_representation_item.description = 'waterplane area')
(descriptive_representation_item.description = 'midship section area')
(descriptive_representation_item.description = 'wetted surface area')
(descriptive_representation_item.description = 'moment for unit change of trim')
(descriptive_representation_item.description = 'longitudinal second moment of area of waterplane')
(descriptive_representation_item.description = 'longitudinal metacentric height')
(descriptive_representation_item.description = 'transverse second moment of area of waterplane')
(descriptive_representation_item.description = 'transverse metacentric height')
(descriptive_representation_item.description = 'centre of buoyancy')
(descriptive_representation_item.description = 'centre of flotation')
(descriptive_representation_item.description = 'centre of lateral resistance')}

5.1.7.8 HYDROSTATIC_PROPERTY_VALUE

AIM element: (compound_representation_item)
(value_representation_item.value_component)
Source: ISO 10303-43
Reference path: /SUBTYPE(Hydrostatic_position_value)/ (see 5.1.7.5)
/SUBTYPE(Hydrostatic_scalar_value)/ (see 5.1.7.9)

5.1.7.9 HYDROSTATIC_SCALAR_VALUE

AIM element: value_representation_item.value_component
Source: ISO 10303-43
Reference path: {value_representation_item
value_representation_item.value_component ->
measure_value = ratio_measure}

5.1.7.9.1 scalar

AIM element: IDENTICAL MAPPING

5.1.7.10 HYDROSTATIC_TABLE

AIM element: representation
Source: ISO 10303-43
Reference path: /ROOT_CLASS(representation, 'hydrostatic table')/

5.1.7.10.1 name

AIM element: representation.name
Source: ISO 10303-43

5.1.7.10.2 mean_shell_thickness

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.99
Reference path: /REP_TO_VAL_REP_ITEM('mean shell thickness', positive_length_measure)/

5.1.7.10.3 hydrostatic_table to hydrostatic_properties_for_constant_-floating_position (as items)

AIM element: PATH
Rules: 5.2.4.97
Reference path: representation
representation.items [i] ->
representation_item =>
compound_representation_item
{/CLASS_ID(compound_representation_item, 'hydrostatic properties for constant floating position')/}

5.1.7.10.4 hydrostatic_table to hydrostatic_property (as properties_in_the_-hydrostatic_table)

AIM element: PATH
Rules: 5.2.4.98
Reference path: representation
representation.items [i] ->
representation_item =>
compound_representation_item
{/CLASS_ID(compound_representation_item, 'hydrostatic property')/}

5.1.7.11 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')/]}

5.1.7.11.1 description

AIM element: property_definition.description
Source: ISO 10303-41

5.1.7.11.2 version_id

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17) through supetype Design_definition (see 5.1.4.2), which inherits from supertype Definition (see 5.1.4.1).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(property_definition)/

5.1.7.11.3 stability_definition to ship (as defined_for)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Design_definition (see 5.1.4.2).

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/
/PROD_DEF_PRODUCT/
{/CLASS_ID(product, 'ship')/}

5.1.7.11.4 stability_definition to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Design_definition (see 5.1.4.2).

AIM element: PATH
Source: ISO 10303-41
Rules: 5.2.4.45, 5.2.4.89
Reference path: property_definition
identification_item = property_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.7.11.5 stability_definition to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Design_definition (see 5.1.4.2).

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.7.11.6 stability_definition to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Design_definition (see 5.1.4.2).

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.7.11.7 stability_definition to stability_table (as representations)

NOTE Attribute inherited from supertype Design_definition (see 5.1.4.2).

AIM element: PATH
Rules: 5.2.4.66
Reference path: /PROP_DEF_TO_REP/
{/CLASS_ID(representation, 'stability table')/}

5.1.7.12 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')/

5.1.7.12.1 stability_properties_for_one_floating_position to centre_location (as centre_of_gravity_above_keel)

AIM element: PATH
Rules: 5.2.4.114, 5.2.4.147
Reference path: /COMPOUND('centre of gravity above keel')/
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘centre location’)/}

5.1.7.12.2 stability_properties_for_one_floating_position to floating_position (as definition_of_starting_floating_position)

AIM element: PATH
Rules: 5.2.4.47
Reference path: /COMPOUND('definition of starting floating position')/
compound_representation_item
{/CLASS_ID(compound_representation_item, 'floating position')/}

5.1.7.12.3 stability_properties_for_constant_floating_position to stability_-table (as related_stability_table)

NOTE attribute is inverse relationship for Stability_table attribute items (see 5.1.7.14.3).

AIM element: PATH
Reference path: compound_representation_item <=
representation_item <-
representation.items [i]
representation
{/CLASS_ID(representation, 'stability table')/}

5.1.7.12.4 stability_properties_for_one_floating_position to stability_-property (as stability_properties_for_different_angles_of_heel)

AIM element: PATH
Rules: 5.2.4.146
Reference path: /COMPOUND('stability properties for different angles of heel')/
compound_representation_item
{/CLASS_ID(compound_representation_item, 'stability property')/}

5.1.7.13 STABILITY_PROPERTY

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: /ROOT_CLASS(compound_representation_item, 'stability property')/

5.1.7.13.1 angle_of_heel

AIM element: plane_angle_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.148
Reference path: /COMPOUND('angle of heel')/
value_representation_item
{value_representation_item.value_component ->
measure_value = plane_angle_measure}

5.1.7.13.2 righting_arm

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.148
Reference path: /COMPOUND('righting arm')/
value_representation_item
{value_representation_item.value_component ->
measure_value = positive_length_measure}

5.1.7.13.3 stability_property to centre_location (as centre_of_buoyancy)

AIM element: PATH
Rules: 5.2.4.114, 5.2.4.148
Reference path: /COMPOUND('centre of buoyancy')/
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘centre location’)/}

5.1.7.14 STABILITY_TABLE

AIM element: representation
Source: ISO 10303-41
Reference path: /ROOT_CLASS(representation, 'stability table')/

5.1.7.14.1 mean_shell_thickness

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.111
Reference path: /REP_TO_VAL_REP_ITEM('mean shell thickness', positive_length_measure)/

5.1.7.14.2 name

AIM element: representation.name
Source: ISO 10303-43

5.1.7.14.3 stability_table to stability_properties_for_one_floating_position (as items)

AIM element: PATH
Rules: 5.2.4.110
Reference path: representation
representation.items [i] ->
representation_item =>
compound_representation_item
{/CLASS_ID(compound_representation_item, 'stability properties for one floating position')/}

5.1.7.15 SUBTRACTION_OF_MOULDED_FORM

AIM element: property_definition
Source: ISO 10303-41
Reference path: /ROOT_CLASS(property_definition, 'subtraction of moulded form')/

5.1.7.15.1 subtraction_of_moulded_form to moulded_form (as displacement_of_moulded_form_to_add_or_subtract)

NOTE Attribute inherited from supertype Displacement_operation (see 5.1.7.2).

AIM element: /SUPERTYPE(Displacement_operation)/ (see 5.1.7.2)

5.1.8 items UoF

5.1.8.1 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 5.1.8.3)
#2: /SUBTYPE(item_relationship)/ (see 5.1.8.4)
#3: /SUBTYPE(item_structure)/ (see 5.1.8.5)

5.1.8.1.1 definable_object to global_id (as id)

AIM element: #1: /SUBTYPE(item)/ (see 5.1.8.3)
#2: /SUBTYPE(item_relationship)/ (see 5.1.8.4)
#3: /SUBTYPE(item_structure)/ (see 5.1.8.5)

5.1.8.2 GLOBAL_ID

AIM element: applied_identification_assignment
Source: ISO 10303-216
Reference path: applied_identification_assignment <=
identification_assignment
{identification_assignment.role->
identification_role
identification_role.name = 'globally unambiguous identifier'}

5.1.8.2.1 id

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Reference path: applied_identification_assignment <=
identification_assignment
identification_assignment.assigned_id

5.1.8.3 ITEM

#1: if item is a change
#2: if item is a moulded_form
#3: if item is a ship
#3: if item is a ship_moulded_form

AIM element: #1: /SUBTYPE(change)/ (see 5.1.3.4)
#2: /SUBTYPE(moulded_form)/ (see 5.1.14.1)
#3: /SUBTYPE(ship)/ (see 5.1.8.6)
#4: /SUBTYPE(ship_moulded_form)/ (see 5.1.14.11)

5.1.8.3.1 description

AIM element: #1: /SUBTYPE(change)/ (see 5.1.3.4.1)
#2: /SUBTYPE(moulded_form)/ (see 5.1.14.1.1)
#3: /SUBTYPE(ship)/ (see 5.1.8.6.1)
#4: /SUBTYPE(ship_moulded_form)/ (see 5.1.14.11.1)

5.1.8.3.2 name

AIM element: #1: /SUBTYPE(change)/ (see 5.1.3.4.2)
#2: /SUBTYPE(moulded_form)/ (see 5.1.14.1.2)
#3: /SUBTYPE(ship)/ (see 5.1.8.6.2)
#4: /SUBTYPE(ship_moulded_form)/ (see 5.1.14.11.2)

5.1.8.3.3 item to external_reference (as documentation)

AIM element: #1: /SUBTYPE(change)/ (see 5.1.3.4.4)
#2: /SUBTYPE(moulded_form)/ (see 5.1.14.1.3)
#3: /SUBTYPE(ship)/ (see 5.1.8.6.3)
#4: /SUBTYPE(ship_moulded_form)/ (see 5.1.14.11.4)

5.1.8.3.4 item to global_id (as id)

NOTE Attribute inherited from supertype Definable_object(see 5.1.8.1).

AIM element: #1: /SUBTYPE(change)/ (see 5.1.3.4.5)
#2: /SUBTYPE(moulded_form)/ (see 5.1.14.1.4)
#3: /SUBTYPE(ship)/ (see 5.1.8.6.4)
#4: /SUBTYPE(ship_moulded_form)/ (see 5.1.14.11.7)

5.1.8.3.5 item to ship (as ship_context)

AIM element: #1: /SUBTYPE(change)/ (see 5.1.3.4.6)
#2: /SUBTYPE(moulded_form)/ (see 5.1.14.1.5)
#4: /SUBTYPE(ship_moulded_form)/ (see 5.1.14.11.10)

5.1.8.4 ITEM_RELATIONSHIP

AIM element: /SUBTYPE(moulded_form_relationship)/ (see 5.1.14.5)

5.1.8.4.1 description

AIM element: /SUBTYPE(moulded_form_relationship)/ (see 5.1.14.5.1)

5.1.8.4.2 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: /SUBTYPE(moulded_form_relationship)/ (see 5.1.14.5.2)

5.1.8.4.3 item_relationship to external_instance_reference (as external_item_-1)

AIM element: /SUBTYPE(moulded_form_relationship)/ (see 5.1.14.5.3)

5.1.8.4.4 item_relationship to external_instance_reference (as external_item_-2)

AIM element: /SUBTYPE(moulded_form_relationship)/ (see 5.1.14.5.4)

5.1.8.4.5 item_relationship to global_id (as id)

NOTE Attribute inherited from supertype 5.1.8.1

AIM element: /SUBTYPE(moulded_form_relationship)/ (see 5.1.14.5.5)

5.1.8.4.6 item_relationship to item (as item_1)

AIM element: /SUBTYPE(moulded_form_relationship)/ (see 5.1.14.5.6)

5.1.8.4.7 item_relationship to item (as item_2)

AIM element: /SUBTYPE(moulded_form_relationship)/ (see 5.1.14.5.7)

5.1.8.5 ITEM_STRUCTURE

AIM element: /SUBTYPE(ship_moulded_form)/ (see 5.1.14.11)

5.1.8.5.1 description

AIM element: /SUBTYPE(ship_moulded_form)/ (see 5.1.14.11.1)

5.1.8.5.2 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: /SUBTYPE(ship_moulded_form)/ (see 5.1.14.11.3)

5.1.8.5.3 item_structure to external_instance_reference (as external_items)

AIM element: /SUBTYPE(ship_moulded_form)/ (see 5.1.14.11.5)

5.1.8.5.4 item_structure to external_instance_reference (as external_-relationships)

AIM element: /SUBTYPE(ship_moulded_form)/ (see 5.1.14.11.6)

5.1.8.5.5 item_structure to global_id(as id)

NOTE Attribute inherited from supertype Definable_object(see 5.1.8.1).

AIM element: /SUBTYPE(ship_moulded_form)/ (see 5.1.14.11.7)

5.1.8.5.6 item_structure to item (as items)

AIM element: /SUBTYPE(ship_moulded_form)/ (see 5.1.14.11.8)

5.1.8.5.7 item_structure to item_relationship (as relationships)

AIM element: /SUBTYPE(ship_moulded_form)/ (see 5.1.14.11.9)

5.1.8.6 SHIP

AIM element: product
Source: ISO 10303-41
Reference path: {[/CLASS(product, 'ship', 'item')/]
[/CLASS(product, 'item', 'definable object')/]
[/ROOT_CLASS(product, 'definable object')/]}

5.1.8.6.1 description

NOTE Attribute inherited from supertype Item (see 5.1.8.3).

AIM element: product.description
Source: ISO 10303-41

5.1.8.6.2 name

NOTE Attribute inherited from supertype Item (see 5.1.8.3).

AIM element: product.name
Source: ISO 10303-41

5.1.8.6.3 ship to external_reference (as documentation)

NOTE Attribute inherited from supertype Item (see 5.1.8.3).

#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')/}

5.1.8.6.4 ship to global_id (as id)

NOTE Attribute inherited from supertype Definable_object (see 5.1.8.1).

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

5.1.8.6.5 ship to item (as ship_items)

#1: if the item is a moulded_form or ship_moulded_form
#2: if the 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

5.1.8.6.6 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’)}

5.1.8.6.7 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

5.1.8.6.8 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

5.1.9 location_concepts UoF

5.1.9.1 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')/]}

5.1.9.1.1 description

NOTE Attribute inherited from supertype Spacing_table(see 5.1.9.10).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.1)

5.1.9.1.2 name

NOTE Attribute inherited from supertype Spacing_table(see 5.1.9.10).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.2)

5.1.9.1.3 version_id

NOTE Attribute inherited from supertype Versionable_object)see 5.1.9.17).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.3)

5.1.9.1.4 buttock_table to definable_object (as defined_for)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.4)

5.1.9.1.5 buttock_table to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.5)

5.1.9.1.6 buttock_table to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.6)

5.1.9.1.7 buttock_table to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.7)

5.1.9.1.8 buttock_table to transversal_position (as spacing_table_-representations)

NOTE Attribute inherited from supertype Transversal_table (see 5.1.9.13).

AIM element: /SUPERTYPE(transversal_table)/ (see 5.1.9.13.8)

5.1.9.2 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’)/]}

5.1.9.2.1 description

NOTE Attribute inherited from supertype Spacing_table (see 5.1.9.10).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.7.1)

5.1.9.2.2 name

NOTE Attribute inherited from supertype Spacing_table (see 5.1.9.10).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.7.2)

5.1.9.2.3 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.7.3)

5.1.9.2.4 frame_table to definable_object (as defined_for)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.7.4)

5.1.9.2.5 frame_table to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.7.5)

5.1.9.2.6 frame_table to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.7.6)

5.1.9.2.7 frame_table to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.7.7)

5.1.9.2.8 frame_table to longitudinal_position (as spacing_table_-representations)

NOTE Attribute inherited from supertype Longitudinal_table (see 5.1.9.7).

AIM element: /SUPERTYPE(longitudinal_table)/ (see 5.1.9.7.8)

5.1.9.3 GLOBAL_AXIS_PLACEMENT

AIM element: [product_definition]
[axis2_placement_3d]
Source: ISO 10303-41, ISO 10303-42
Rules: 5.2.4.44, 5.2.4.95, 5.2.4.114
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

5.1.9.3.1 after_perpendicular_offset

AIM element: value_representation_item.value_component
Source: ISO 10303-41
Rules: 5.2.4.95, 5.2.4.114
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)/

5.1.9.3.2 description

NOTE Attribute inherited from supertype Item (see 5.1.8.3)

AIM element: product_definition.description
Source: ISO 10303-41

5.1.9.3.3 orientation

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.95, 5.2.4.114
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 pointing')
(descriptive_representation_item.description = 'aft pointing')}

5.1.9.3.4 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(product_definition)/

5.1.9.3.5 global_axis_placement to ship (as defined_for)

NOTE Attribute inherited from supertype General_characteristics_definition (see 5.1.4.4).

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product, 'ship')/}

5.1.9.3.6 global_axis_placement to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.70
Reference path: product_definition
identification_item = product_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.9.3.7 global_axis_placement to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS('local_units')/
unit
unit = derived_unit
derived_unit

5.1.9.3.8 global_axis_placement to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS('local_units')/
unit
unit = named_unit
nameed_unit

5.1.9.4 LOCAL_CO_ORDINATE_SYSTEM

AIM element: [property_definition]
[axis2_placement_3d]

Source: ISO 10303-41, ISO 10303-42
Rules: 5.2.4.114, 5.2.4.82, 5.2.4.100
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

5.1.9.4.1 description

NOTE Attribute inherited from supertype Item (see 5.1.8.3)

AIM element: property_definition.description
Source: ISO 10303-41

5.1.9.4.2 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(property_definition)/

5.1.9.4.3 local_co_ordinate_system to definable_object (as defined_for)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/

5.1.9.4.4 local_co_ordinate_system to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.89
Reference path: property_definition
identification_item = property_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.9.4.5 local_co_ordinate_system to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local_units')/
unit
unit = derived_unit
derived_unit

5.1.9.4.6 local_co_ordinate_system to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local_units')/
unit
unit = named_unit
named_unit

5.1.9.4.7 local_co_ordinate_system to global_axis_placement (as parent)

AIM element: PATH
Rules: 5.2.4.125
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

5.1.9.4.8 local_co_ordinate_system to local_co_ordinate_system (as parent)

AIM element: PATH
Rules: 5.2.4.125
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

5.1.9.4.9 local_co_ordinate_system to local_co_ordinate_system_with_-position_reference (as parent)

AIM element: PATH
Rules: 5.2.4.125
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

5.1.9.5 LOCAL_CO_ORDINATE_SYSTEM_WITH_POSITION_-REFERENCE

AIM element: [property_definition]
[axis2_placement_3d]
Source: ISO 10303-41, ISO 10303-42
Rules: 5.2.4.114, 5.2.4.83, 5.2.4.101
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

5.1.9.5.1 description

NOTE Attribute inherited from supertype Item (see 5.1.8.3)

AIM element: /SUPERTYPE(local_co_ordinate_system)/ (see 5.1.9.4.1)

5.1.9.5.2 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: /SUPERTYPE(local_co_ordinate_system)/ (see 5.1.9.4.2)

5.1.9.5.3 local_co_ordinate_system_with_position_reference to definable_-object (as defined_for)

NOTE Attribute inherited from supertype 5.1.4.1

AIM element: /SUPERTYPE(local_co_ordinate_system)/ (see 5.1.9.4.3)

5.1.9.5.4 local_co_ordinate_system_with_position_reference to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(local_co_ordinate_system)/ (see 5.1.9.4.4)

5.1.9.5.5 local_co_ordinate_system_with_position_reference to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(local_co_ordinate_system)/ (see 5.1.9.4.5)

5.1.9.5.6 local_co_ordinate_system_with_position_reference to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(local_co_ordinate_system)/ (see 5.1.9.4.6)

5.1.9.5.7 local_co_ordinate_system_with_position_reference to length_-measure (as longitudinal_ref)

AIM element: length_measure
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}

5.1.9.5.8 local_co_ordinate_system_with_position_reference to spacing_-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}}

5.1.9.5.9 local_co_ordinate_system_with_position_reference to global_axis_-placement (as parent)

AIM element: PATH
Rules: 5.2.4.125
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 with position reference 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

5.1.9.5.10 local_co_ordinate_system_with_position_reference to local_co_-ordinate_system (as parent)

AIM element: /SUPERTYPE(local_co_ordinate_system)/ (see 5.1.9.4.8)

5.1.9.5.11 local_co_ordinate_system_with_position_reference to local_co_-ordinate_system_with_position_reference (as parent)

AIM element: /SUPERTYPE(local_co_ordinate_system)/ (see 5.1.9.4.9)

5.1.9.5.12 local_co_ordinate_system_with_position_reference to length_-measure (as transversal_ref)

AIM element: length_measure
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}

5.1.9.5.13 local_co_ordinate_system_with_position_reference to spacing_-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}}

5.1.9.5.14 local_co_ordinate_system_with_position_reference to length_-measure (as vertical_ref)

AIM element: length_measure
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}

5.1.9.5.15 local_co_ordinate_system_with_position_reference to spacing_-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}}

5.1.9.6 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}

5.1.9.6.1 name

NOTE Attribute inherited from supertype Spacing_position (see 5.1.9.8).

AIM element: /SUPERTYPE(spacing_position)/ (see 5.1.9.8.1)
Reference path: see supertype mappings

5.1.9.6.2 position

NOTE Attribute inherited from supertype Spacing_position (see 5.1.9.8).

AIM element: /SUPERTYPE(spacing_position)/ (see 5.1.9.8.2)

5.1.9.6.3 position_number

NOTE Attribute inherited from supertype Spacing_position (see 5.1.9.8).

AIM element: /SUPERTYPE(spacing_position)/ (see 5.1.9.8.3)

5.1.9.7 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')/]}

5.1.9.7.1 description

NOTE Attribute inherited from supertype Spacing_position (see 5.1.9.10).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.1)

5.1.9.7.2 name

NOTE Attribute inherited from supertype Spacing_position (see 5.1.9.10).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.2)

5.1.9.7.3 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.3)

5.1.9.7.4 longitudinal_table to definable_object (as defined_for)

NOTE Attribute inherited from supertype 5.1.4.1

AIM element: /SUPERTYPE(spacing_table/ (see 5.1.9.10.4)

5.1.9.7.5 longitudinal_table to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.5)

5.1.9.7.6 longitudinal_table to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.6)

5.1.9.7.7 longitudinal_table to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.7)

5.1.9.7.8 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')/}

5.1.9.8 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}

5.1.9.8.1 name

AIM element: compound_representation_item.name
Source: ISO 10303-43

5.1.9.8.2 position

AIM element: length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.143
Reference path: /COMPOUND('position')/
value_representation_item
{value_representation_item.value_component ->
measure_value = length_measure}

5.1.9.8.3 position_number

AIM element: count_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.143
Reference path: /COMPOUND('position number')/
value_representation_item
{value_representation_item.value_component ->
measure_value = count_measure}

5.1.9.9 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}

5.1.9.9.1 name

NOTE Attribute inherited from supertype Spacing_position (see 5.1.9.8).

AIM element: compound_representation_item.name
Source: ISO 10303-43

5.1.9.9.2 offset

AIM element: length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.145
Reference path: /COMPOUND(‘offset‘)/
value_representation_item
{value_representation_item.value_component ->
measure_value = length_measure}

5.1.9.9.3 position

AIM element: length_measure
Source: ISO 10303-43
Reference path: /COMPOUND(‘position‘)/
value_representation_item
{value_representation_item.value_component ->
measure_value = length_measure}

5.1.9.9.4 position_number

NOTE Attribute inherited from supertype Spacing_position (see 5.1.9.8).

AIM element: length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.143
Reference path: /COMPOUND(‘position number‘)/
value_representation_item
{value_representation_item.value_component ->
measure_value = count_measure}

5.1.9.9.5 spacing_position_with_offset to spacing_position (as relating_spacing_position)

AIM element: PATH
Rules: 5.2.4.144
Reference path: /COMPOUND(‘relating spacing position‘)/
{/CLASS_HELP(compound_representation_item)/
(group.name = ‘longitudinal position’)
(group.name = ‘transversal position’)
(group.name = ‘vertical position’)
class}}

5.1.9.10 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’)/]}

5.1.9.10.1 description

AIM element: property_definition.description
Source: ISO 10303-43

5.1.9.10.2 name

AIM element: property_definition.name
Source: ISO 10303-43

5.1.9.10.3 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(property_definition)/

5.1.9.10.4 spacing_table to definable_object (as defined_for)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: property_definition
property_defintion.definition ->
characterized_definition
characterized_definition = characterized_product_definition
characterized_product_definition = product_definition
product_definition
product_definition.formation ->
product_definition_formation
product_definition_formation.of_product ->
product
{/CLASS_ID(product, 'ship')}

5.1.9.10.5 spacing_table to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.89
Reference path: property_definition
identification_item = property_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.9.10.6 spacing_table to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.9.10.7 spacing_table to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.9.10.8 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}}

5.1.9.11 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’)/]}

5.1.9.11.1 description

NOTE Attribute inherited from supertype Spacing_table (see 5.1.9.10).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.7.1)

5.1.9.11.2 name

NOTE Attribute inherited from supertype Spacing_table (see 5.1.9.10).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.7.2)

5.1.9.11.3 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.3)

5.1.9.11.4 station_table to definable_object (as defined_for)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.4)

5.1.9.11.5 station_table to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.7.5)

5.1.9.11.6 station_table to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.7.6)

5.1.9.11.7 station_table to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.7.7)

5.1.9.11.8 station_table to longitudinal_position (as spacing_table_-representations)

NOTE Attribute inherited from supertype Longitudinal_table (see 5.1.9.7).

AIM element: /SUPERTYPE(longitudinal_table)/ (see 5.1.9.7.8)

5.1.9.12 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}

5.1.9.12.1 name

NOTE Attribute inherited from supertype Spacing_position (see 5.1.9.8).

AIM element: /SUPERTYPE(spacing_position)/ (see 5.1.9.8.1)

5.1.9.12.2 position

NOTE Attribute inherited from supertype Spacing_position (see 5.1.9.8).

AIM element: /SUPERTYPE(spacing_position)/ (see 5.1.9.8.2)

5.1.9.12.3 position_number

NOTE Attribute inherited from supertype Spacing_position (see 5.1.9.8).

AIM element: /SUPERTYPE(spacing_position)/ (see 5.1.9.8.3)

5.1.9.13 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’)/]}

5.1.9.13.1 description

NOTE Attribute inherited from supertype Spacing_table (see 5.1.9.10).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.1)

5.1.9.13.2 name

NOTE Attribute inherited from supertype Spacing_table (see 5.1.9.10).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.2)

5.1.9.13.3 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.3)

5.1.9.13.4 transversal_table to definable_object (as defined_for)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.4)

5.1.9.13.5 transversal_table to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.5)

5.1.9.13.6 transversal_table to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.6)

5.1.9.13.7 transversal_table to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.7)

5.1.9.13.8 transversal_table to transversal_position (as spacing_table_-representations)

AIM element: PATH
Reference path: property_definition<=
/PROP_DEF_TO_REP/
representation.items [i] ->
representation_item =>
compound_representation_item
{/CLASS_ID(compound_representation_item, 'transversal position')/}

5.1.9.14 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}

5.1.9.14.1 name

NOTE Attribute inherited from supertype 5.1.9.8

AIM element: /SUPERTYPE(spacing_position)/ (see 5.1.9.8.1)

5.1.9.14.2 position

NOTE Attribute inherited from supertype 5.1.9.8

AIM element: /SUPERTYPE(spacing_position)/ (see 5.1.9.8.2)

5.1.9.14.3 position_number

NOTE Attribute inherited from supertype 5.1.9.8

AIM element: /SUPERTYPE(spacing_position)/ (see 5.1.9.8.3)

5.1.9.15 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')/]}

5.1.9.15.1 description

NOTE Attribute inherited from supertype 5.1.9.10

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.1)

5.1.9.15.2 name

NOTE Attribute inherited from supertype 5.1.9.10

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.2)

5.1.9.15.3 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.3)

5.1.9.15.4 vertical_table to definable_object (as defined_for)

NOTE Attribute inherited from supertype 5.1.4.1

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.4)

5.1.9.15.5 vertical_table to global_id (as id)

NOTE Attribute inherited from supertype 5.1.4.1

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.5)

5.1.9.15.6 vertical_table to derived_unit (as local_units)

NOTE Attribute inherited from supertype 5.1.4.1

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.6)

5.1.9.15.7 vertical_table to named_unit (as local_units)

NOTE Attribute inherited from supertype 5.1.4.1

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.7)

5.1.9.15.8 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')/}

5.1.9.16 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')/]}

5.1.9.16.1 description

NOTE Attribute inherited from supertype 5.1.9.10

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.1)

5.1.9.16.2 name

NOTE Attribute inherited from supertype 5.1.9.10

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.2)

5.1.9.16.3 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.3)

5.1.9.16.4 waterline_table to definable_object (as defined_for)

NOTE Attribute inherited from supertype 5.1.4.1

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.4)

5.1.9.16.5 waterline_table to global_id (as id)

NOTE Attribute inherited from supertype 5.1.4.1

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.5)

5.1.9.16.6 waterline_table to derived_unit (as local_units)

NOTE Attribute inherited from supertype 5.1.4.1

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.6)

5.1.9.16.7 waterline_table to named_unit (as local_units)

NOTE Attribute inherited from supertype 5.1.4.1

AIM element: /SUPERTYPE(spacing_table)/ (see 5.1.9.10.7)

5.1.9.16.8 waterline_table to vertical_position (as spacing_table_-representations)

NOTE Attribute inherited from supertype 5.1.9.15

AIM element: /SUPERTYPE(vertical_table)/ (see 5.1.9.15.8)

5.1.9.17 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 5.1.4.1)
#2: /SUBTYPE(document)/ (see 5.1.5.1)
#3: /SUBTYPE(item_relationship)/ (see 5.1.8.4)
#4: /SUBTYPE(item_structure)/ (see 5.1.8.5)
#5: /SUBTYPE(revision)/ (see 5.1.3.13)

5.1.9.17.1 version_id

AIM element: #1: /SUBTYPE(definition)/ (see 5.1.4.1.2)
#2: /SUBTYPE(document)/ (see 5.1.5.1.5)
#3: /SUBTYPE(item_relationship)/ (see 5.1.8.4.2)
#4: /SUBTYPE(item_structure)/ (see 5.1.8.5.2)
#5: /SUBTYPE(revision)/ (see 5.1.3.13.3)

5.1.10 offset_table_representations UoF

5.1.10.1 OFFSET_POINT_TABLE_MODEL

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: /ROOT_CLASS(compound_representation_item, 'offset point table model')/

5.1.10.1.1 offset_point_table_type

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.54
Reference path: /COMPOUND('offset point table type')/
descriptive_representation_item
{(descriptive_representation_item.name = 'waterline table')
(descriptive_representation_item.name = 'station table')
(descriptive_representation_item.name = 'buttock table')
(descriptive_representation_item.name = 'user defined table')}

5.1.10.1.2 offset_point_table_model to section_of_offset_point_table (as offset_point_table_sections)

AIM element: PATH
Rules: 5.2.4.105
Reference path: /COMPOUND('offset point table section')/
compound_representation_item
{/CLASS_ID(compound_representation_item, 'section of offset point table')/}

5.1.10.2 OFFSET_TABLE_SHAPE_REPRESENTATION

AIM element: shape_representation
Source: ISO 10303-41
Reference path: {[/CLASS(shape_representation,'offset table shape representation','moulded form shape representation')/]
[/ROOT_CLASS(shape_representation,'moulded form shape representation')/}

5.1.10.2.1 moulded_form_representation_id

NOTE Attribute inherited from supertype Moulded_form_shape_representation (see 5.1.14.8)

AIM element: PATH
Reference path: /SUPERTYPE(Moulded_form_shape_representation)/ (see 5.1.14.8.1)

5.1.10.2.2 offset_table_shape_representation to offset_point_table_model (as items)

AIM element: PATH
Rules: 5.2.4.106
Reference path: shape_representation <=
representation
representation.items[i] ->
representation_item =>
compound_representation_item
{/CLASS_ID(compound_representation_item, 'offset point table model')/}

5.1.10.2.3 offset_table_shape_representation to symmetry (as moulded_-form_symmetry)

NOTE Attribute inherited from supertype Moulded_form_shape_representation (see 5.1.14.8)

AIM element: PATH
Reference path: /SUPERTYPE(Moulded_form_shape_representation)/ (see 5.1.14.8.2)

5.1.10.3 SECTION_OF_OFFSET_POINT_TABLE

AIM element: compound_representation_item
Source: ISO 10303-43
Reference path: /ROOT_CLASS(compound_representation_item, 'section of offset point table')/

5.1.10.3.1 section_identifier

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.33
Reference path: /HAS_ID_1_ROLE(compound_representation_item, 'section identifier')/

5.1.10.3.2 section_of_offset_point_table to offset_point_table_model (as for_-table)

NOTE attribute is inverse relationship for Offset_point_table_model attribute offset_point_table_sections (see 5.1.10.1.2).

AIM element: PATH
Reference path: compound_representation_item <=
representation_item <-
list_representation_item[i]
compound_item_definition =
list_representation_item <-
compound_representation_item.item_element
compound_representation_item
{/CLASS_ID(compound_representation_item, 'offset point table model')/}

5.1.10.3.3 section_of_offset_point_table to ship_point (as section_points)

AIM element: PATH
Rules: 5.2.4.104
Reference path: /COMPOUND('section point')/
compound_representation_item
{/CLASS_ID(compound_representation_item, 'ship point')/}

5.1.11 ship_design_parameter UoF

5.1.11.1 APPENDAGE_MOULDED_FORM_DESIGN_PARAMETER

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.75
Reference path: {[/CLASS(property_definition, 'appendage moulded form design parameter', 'moulded form characteristics definition')/]
[/CLASS(property_definition, 'moulded form characteristics definition', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.11.1.1 appendage_breadth

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.77
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('appendage moulded form design parameter', 'appendage breadth', positive_length_measure)/

5.1.11.1.2 appendage_depth

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.77
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('appendage moulded form design parameter', 'appendage depth', positive_length_measure)/

5.1.11.1.3 appendage_length

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.77
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('appendage moulded form design parameter', 'appendage length', positive_length_measure)/

5.1.11.1.4 description

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.1)

5.1.11.1.5 moulded_form_displacement

AIM element: volume_measure
Source: ISO 10303-43
Rules: 5.2.4.90, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('appendage moulded form design parameter', 'moulded form displacement', volume_measure)/

5.1.11.1.6 moulded_form_outer_surface

AIM element: area_measure
Source: ISO 10303-43
Rules: 5.2.4.115, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('appendage moulded form design parameter', 'moulded form outer surface', area_measure)/

5.1.11.1.7 type_of_appendage

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.77 , 5.2.4.150
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('appendage moulded form design parameter', 'type of appendage')/
{(descriptive_representation_item.description = 'active fin')
(descriptive_representation_item.description = 'air emitter')
(descriptive_representation_item.description = 'arrays')
(descriptive_representation_item.description = 'bilge keel')
(descriptive_representation_item.description = 'bow thruster')
(descriptive_representation_item.description = 'cathodic protection anode')
(descriptive_representation_item.description = 'duck tail')
(descriptive_representation_item.description = 'passive fin')
(descriptive_representation_item.description = 'rudder')
(descriptive_representation_item.description = 'sea chest')
(descriptive_representation_item.description = 'shaft bossings')
(descriptive_representation_item.description = 'shaft strut ')
(descriptive_representation_item.description = 'skeg')
(descriptive_representation_item.description = 'spray rail')
(descriptive_representation_item.description = 'stern flap')
(descriptive_representation_item.description = 'stern thruster')
(descriptive_representation_item.description = 'user defined')}

5.1.11.1.8 usr_def_appendage_type

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.150, 5.2.4.115
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('appendage moulded form design parameter', 'type of appendage')/
descriptive_representation_item.description

5.1.11.1.9 version_id

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.4)

5.1.11.1.10 appendage_moulded_form_design_parameter to moulded_form (as defined_for)

NOTE Attribute inherited from supertype Moulded_form_characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.5)

5.1.11.1.11 appendage_moulded_form_design_parameter to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.6)

5.1.11.1.12 appendage_moulded_form_design_parameter to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.7)

5.1.11.1.13 appendage_moulded_form_design_parameter to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.8)

5.1.11.2 BOTTOM_MOULDED_FORM_DESIGN_PARAMETER

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.76
Reference path: {[/CLASS(property_definition, 'bottom moulded form design parameter', 'moulded form characteristics definition')/]
[/CLASS(property_definition, 'moulded form characteristics definition', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.11.2.1 aft_end_of_flat_of_bottom

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.91
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bottom moulded form design parameter', 'aft end of flat of bottom', positive_length_measure)/

5.1.11.2.2 bilge_radius

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.91
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bottom moulded form design parameter', 'bilge radius', positive_length_measure)/

5.1.11.2.3 description

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.1)

5.1.11.2.4 flat_of_bottom_breadth

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.91
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bottom moulded form design parameter', 'flat of bottom breadth', positive_length_measure)/

5.1.11.2.5 front_end_of_flat_of_bottom

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.91
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bottom moulded form design parameter', 'front end of flat of bottom', positive_length_measure)/

5.1.11.2.6 length_of_flat_of_bottom

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.91
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bottom moulded form design parameter', 'length of flat of bottom', positive_length_measure)/

5.1.11.2.7 moulded_form_displacement

AIM element: volume_measure
Source: ISO 10303-43
Rules: 5.2.4.116, 5.2.4.114
Reference path: property_definition <=
/PROP_DEF_TO_VAL_REP_ITEM('bottom moulded form design parameter',
'moulded form displacement', volume_measure)/

5.1.11.2.8 moulded_form_outer_surface

AIM element: area_measure
Source: ISO 10303-43
Rules: 5.2.4.116, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bottom moulded form design parameter',
'moulded form outer surface', area_measure)/

5.1.11.2.9 rake_of_keel

AIM element: plane_angle_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.91
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bottom moulded form design parameter',
'rake of keel', plane_angle_measure)/

5.1.11.2.10 rise_of_floor

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.91
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bottom moulded form design parameter',
'rise of floor', positive_length_measure)/

5.1.11.2.11 version_id

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.4)

5.1.11.2.12 bottom_moulded_form_design_parameter to moulded_form (as defined_for)

NOTE Attribute inherited from supertype Moulded_form_characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.5)

5.1.11.2.13 bottom_moulded_form_design_parameter to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.6)

5.1.11.2.14 bottom_moulded_form_design_parameter to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.7)

5.1.11.2.15 bottom_moulded_form_design_parameter to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.8)

5.1.11.3 BULB_MOULDED_FORM_DESIGN_PARAMETER

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.77
Reference path: {[/CLASS(property_definition, 'bulb moulded form design parameter', 'moulded form characteristics definition')/]
[/CLASS(property_definition, 'moulded form characteristics definition', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.11.3.1 bulb_breadth

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.92
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bulb moulded form design parameter', 'bulb breadth', positive_length_measure)/

5.1.11.3.2 bulb_breadth_pp

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.92
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bulb moulded form design parameter', 'bulb breadth pp', positive_length_measure)/

5.1.11.3.3 bulb_depth

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.92
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bulb moulded form design parameter', 'bulb depth', positive_length_measure)/

5.1.11.3.4 bulb_depth_pp

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.92
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bulb moulded form design parameter', 'bulb depth pp', positive_length_measure)/

5.1.11.3.5 bulb_frame_section_area_at_pp

AIM element: area_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.92
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bulb moulded form design parameter', 'bulb frame section area at pp', area_measure)/

5.1.11.3.6 bulb_length

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114,5.2.4.92
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bulb moulded form design parameter', 'bulb length', positive_length_measure)/

5.1.11.3.7 bulb_length_from_pp

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.92
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bulb moulded form design parameter', 'bulb length from pp', positive_length_measure)/

5.1.11.3.8 bulb_location

AIM element: descriptive_representation_item.description
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.92
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('bulb moulded form design parameter', 'bulb location')/
{(descriptive_representation_item.description = 'bow')
(descriptive_representation_item.description = 'stern')}

5.1.11.3.9 description

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.1)

5.1.11.3.10 moulded_form_displacement

AIM element: volume_measure
Source: ISO 10303-43
Rules: 5.2.4.117, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bulb moulded form design parameter', 'moulded form displacement', volume_measure)/

5.1.11.3.11 moulded_form_outer_surface

AIM element: area_measure
Source: ISO 10303-43
Rules: 5.2.4.117, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('bulb moulded form design parameter', 'moulded form outer surface', area_measure)/

5.1.11.3.12 version_id

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.4)

5.1.11.3.13 bulb_moulded_form_design_parameter to moulded_form (as defined_for)

NOTE Attribute inherited from supertype Moulded_form_characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.5)

5.1.11.3.14 bulb_moulded_form_design_parameter to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.6)

5.1.11.3.15 bulb_moulded_form_design_parameter to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.7)

5.1.11.3.16 bulb_moulded_form_design_parameter to named_unit (as local_-units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.8)

5.1.11.4 DECK_MOULDED_FORM_DESIGN_PARAMETER

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.80
Reference path: {[/CLASS(property_definition, 'deck moulded form design parameter', 'moulded form characteristics definition')/]
[/CLASS(property_definition, 'moulded form characteristics definition', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.11.4.1 camber

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.94
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('deck moulded form design parameter', 'camber', positive_length_measure)/

5.1.11.4.2 description

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.1)

5.1.11.4.3 moulded_form_outer_surface

AIM element: area_measure
Source: ISO 10303-43
Rules: 5.2.4.118, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('deck moulded form design parameter', 'moulded form outer surface', area_measure)/

5.1.11.4.4 moulded_form_displacement

AIM element: volume_measure
Source: ISO 10303-43
Rules: 5.2.4.118, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('deck moulded form design parameter', 'moulded form displacement', volume_measure)/

5.1.11.4.5 sheer_at_AP

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.94
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('deck moulded form design parameter', 'sheer at AP', positive_length_measure)/

5.1.11.4.6 sheer_at_FP

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.94
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('deck moulded form design parameter', 'sheer at FP', positive_length_measure)/

5.1.11.4.7 version_id

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.4)

5.1.11.4.8 deck_moulded_form_design_parameter to moulded_form (as defined_for)

NOTE Attribute inherited from supertype Moulded_form_characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.5)

5.1.11.4.9 deck_moulded_form_design_parameter to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.6)

5.1.11.4.10 deck_moulded_form_design_parameter to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.7)

5.1.11.4.11 deck_moulded_form_design_parameter to named_unit (as local_-units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.8)

5.1.11.5 HULL_MOULDED_FORM_DESIGN_PARAMETER

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.81
Reference path: {[/CLASS(property_definition, 'hull moulded form design parameter', 'moulded form characteristics definition')/]
[/CLASS(property_definition, 'moulded form characteristics definition', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.11.5.1 aft_end_of_flat_of_side

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.96
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'aft end of flat of side', positive_length_measure)/

5.1.11.5.2 aft_end_of_parallel_midbody_at_design_draught

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.96
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'aft end of parallel midbody at design draught', positive_length_measure)/

5.1.11.5.3 block_coefficient

AIM element: ratio_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.96
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'block coefficient', ratio_measure)/

5.1.11.5.4 description

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.1)

5.1.11.5.5 front_end_of_flat_of_side

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.96
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'front end of flat of side', positive_length_measure)/

5.1.11.5.6 front_end_of_parallel_midbody_at_design_draught

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.96
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'front end of parallel midbody at design draught', positive_length_measure)/

5.1.11.5.7 gunwale_radius

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.119
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'gunwale radius', positive_length_measure)/

5.1.11.5.8 hull_breadth

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.119
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'hull breadth', positive_length_measure)/

5.1.11.5.9 hull_depth

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.119
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'hull depth', positive_length_measure)/

5.1.11.5.10 hull_design_draught

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.119
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'hull design draught', positive_length_measure)/

5.1.11.5.11 hull_length_pp

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.119
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'hull length pp', positive_length_measure)/

5.1.11.5.12 hull_length_waterline

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.119
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'hull length waterline', positive_length_measure)

5.1.11.5.13 max_frame_section_area_location

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.119
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'max frame section area location', positive_length_measure)/

5.1.11.5.14 max_wetted_frame_section_area

AIM element: area_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.96
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'max wetted frame section area', area_measure)/

5.1.11.5.15 moulded_form_displacement

AIM element: volume_measure
Source: ISO 10303-43
Rules: 5.2.4.119, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'moulded form displacement', volume_measure)/

5.1.11.5.16 moulded_form_outer_surface

AIM element: area_measure
Source: ISO 10303-43
Rules: 5.2.4.119, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'moulded form outer surface', area_measure)/

5.1.11.5.17 prismatic_coefficient

AIM element: ratio_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.96
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'prismatic coefficient', ratio_measure)/

5.1.11.5.18 version_id

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.4)

5.1.11.5.19 waterline_angle_of_entrance_at_bow

AIM element: plane_angle_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.119
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'waterline angle of entrance at bow', plane_angle_measure)/

5.1.11.5.20 waterline_angle_of_entrance_at_stern

AIM element: plane_angle_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.119
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'waterline angle of entrance at stern', plane_angle_measure)/

5.1.11.5.21 waterplane_coefficient

AIM element: ratio_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.96
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('hull moulded form design parameter', 'waterplane coefficient', ratio_measure)/

5.1.11.5.22 hull_moulded_form_design_parameter to moulded_form (as defined_for)

NOTE Attribute inherited from supertype Moulded_form_characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.5)

5.1.11.5.23 hull_moulded_form_design_parameter to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.6)

5.1.11.5.24 hull_moulded_form_design_parameter to derived_unit (as local_-units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.7)

5.1.11.5.25 hull_moulded_form_design_parameter to named_unit (as local_-units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.8)

5.1.11.5.26 hull_moulded_form_design_parameter to midship_tumble (as midship_tumble_data)

AIM element: PATH
Rules: 5.2.4.46
Reference path: /PROP_DEF_TO_REP/
{/CLASS_ID(representation, 'midship tumble')/}

5.1.11.6 MIDSHIP_TUMBLE

AIM element: representation
Source: ISO 10303-43
Reference path: /ROOT_CLASS(representation, 'midship tumble')/

5.1.11.6.1 tumble_in_at_side

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.102
Reference path: REP_TO_VAL_REP_ITEM('tumble in at side', positive_length_measure)

5.1.11.6.2 tumble_in_at_top

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.102
Reference path: /REP_TO_VAL_REP_ITEM('tumble in at top', positive_length_measure)/

5.1.11.6.3 tumble_out_at_bottom

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.102
Reference path: /REP_TO_VAL_REP_ITEM('tumble out at bottom', positive_length_measure)/

5.1.11.6.4 tumble_out_at_side

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.102
Reference path: /REP_TO_VAL_REP_ITEM('tumble out at side', positive_length_measure)/

5.1.11.7 MOULDED_FORM_CHARACTERISTICS_DEFINITION

#1: Moulded_form_characteristics_definition is an Appendage_moulded_form_design_-parameter
#2: Moulded_form_characteristics_definition is a Bottom_moulded_form_design_parameter
#3: Moulded_form_characteristics_definition is a Bulb_moulded_form_design_parameter

#4: Moulded_form_characteristics_definition is a Hull_moulded_form_design_parameter
#5: Moulded_form_characteristics_definition is a Propeller_moulded_form_design_parameter
#6: Moulded_form_characteristics_definition is a Rudder_moulded_form_design_parameter
#7: Moulded_form_characteristics_definition is a Thruster_moulded_form_design_parameter
#8: Moulded_form_characteristics_definition is a Deck_moulded_form_design_parameter

AIM element: #1/SUBTYPE(Appendage_moulded_form_design_parameter)/ (see 5.1.11.1)
#2/SUBTYPE(Bottom_moulded_form_design_parameter)/ (see 5.1.11.2)
#3/SUBTYPE(Bulb_moulded_form_design_parameter)/ (see 5.1.11.3)
#4/SUBTYPE(Hull_moulded_form_design_parameter)/ (see 5.1.11.5)
#5/SUBTYPE(Propeller_moulded_form_design_parameter)/ (see 5.1.11.9)
#6/SUBTYPE(Rudder_moulded_form_design_parameter)/ (see 5.1.11.10)
#7/SUBTYPE(Thruster_moulded_form_design_parameter)/ (see 5.1.11.12)
#8/SUBTYPE(Deck_moulded_form_design_parameter)/ (see 5.1.11.4)

5.1.11.7.1 description

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: property_definition.description
Source: ISO 10303-41

5.1.11.7.2 moulded_form_displacement

AIM element: #1 /SUBTYPE(Appendage_moulded_form_design_parameter)/ (see 5.1.11.1.5)
#2 /SUBTYPE(Bottom_moulded_form_design_parameter)/ (see 5.1.11.2.7)
#3 /SUBTYPE(Bulb_moulded_form_design_parameter)/ (see 5.1.11.3.10)
#4 /SUBTYPE(Hull_moulded_form_design_parameter)/ (see 5.1.11.5.15)
#5 /SUBTYPE(Propeller_moulded_form_design_parameter)/ (see 5.1.11.9.7)
#6 /SUBTYPE(Rudder_moulded_form_design_parameter)/ (see 5.1.11.10.2)
#7 /SUBTYPE(Thruster_moulded_form_design_parameter)/ (see 5.1.11.12.2)
#8 /SUBTYPE(Deck_moulded_form_design_parameter)/ (see 5.1.11.4.4)

5.1.11.7.3 moulded_form_outer_surface

AIM element: #1 /SUBTYPE(Appendage_moulded_form_design_parameter)/ (see 5.1.11.1.6)
#2 /SUBTYPE(Bottom_moulded_form_design_parameter)/ (see 5.1.11.2.8)
#3 /SUBTYPE(Bulb_moulded_form_design_parameter)/ (see 5.1.11.3.11)
#4 /SUBTYPE(Hull_moulded_form_design_parameter)/ (see 5.1.11.5.16)
#5 /SUBTYPE(Propeller_moulded_form_design_parameter)/ (see 5.1.11.9.8)
#6 /SUBTYPE(Rudder_moulded_form_design_parameter)/ (see 5.1.11.10.3)
#7 /SUBTYPE(Thruster_moulded_form_design_parameter)/ (see 5.1.11.12.3)
#8 /SUBTYPE(Deck_moulded_form_design_parameter)/ (see 5.1.11.4.3)

5.1.11.7.4 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(property_definition)/

5.1.11.7.5 moulded_form_characteristics_definition to moulded_form (as defined_for)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/
{/CLASS_ID(product_definition, 'moulded_form')/}

5.1.11.7.6 moulded_form_characteristics_definition to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.89
Reference path: property_definition
identification_item = property_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.11.7.7 moulded_form_characteristics_definition to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.11.7.8 moulded_form_characteristics_definition to named_unit (as local_-units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: /PROP_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.11.8 PROPELLER_LOCATION

AIM element: representation
Source: ISO 10303-43
Reference path: /ROOT_CLASS(representation, 'propeller location')/

5.1.11.8.1 shaft_line_inclination_x

AIM element: plane_angle_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.107
Reference path: /REP_TO_VAL_REP_ITEM('shaft line inclination x', plane_angle_measure)/

5.1.11.8.2 shaft_line_inclination_y

AIM element: plane_angle_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.107
Reference path: /REP_TO_VAL_REP_ITEM('shaft line inclination y', plane_angle_measure)/

5.1.11.8.3 propeller_location to centre_location (as propeller_location)

AIM element: PATH
Rules: 5.2.4.114, 5.2.4.107
Reference path: property_definition
represented_definition = property_definition
represented_definition <-
property_definition_representation.definition
property_definition_representation
property_definition_representation.used_representation ->
/REP_ITEM(‘propeller location’)/
compound_representation_item
{/CLASS_ID(compound_representation_item, 'centre_location')/}

5.1.11.8.4 propeller_location to centre_location (as shaft_line_location)

AIM element: PATH
Rules: 5.2.4.114, 5.2.4.107
Reference path: property_definition
represented_definition = property_definition
represented_definition <-
property_definition_representation.definition
property_definition_representation
property_definition_representation.used_representation ->
/REP_ITEM('shaft line location')/
compound_representation_item
{/CLASS_ID(compound_representation_item, 'centre_location')/}

5.1.11.9 PROPELLER_MOULDED_FORM_DESIGN_PARAMETER

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.85
Reference path: {[/CLASS(property_definition, 'propeller moulded form design parameter', 'moulded form characteristics definition')/]
[/CLASS(property_definition, 'moulded form characteristics definition', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.11.9.1 blade_mean_height

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.122
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('propeller moulded form design parameter', 'blade mean height', positive_length_measure)/

5.1.11.9.2 chord_length_at_0_7_radius

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.108
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('propeller moulded form design parameter', 'chord length at 0 7 radius', positive_length_measure)/

5.1.11.9.3 description

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.1)

5.1.11.9.4 design_sense_of_rotation

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.108
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('propeller moulded form design parameter', 'design sense of rotation')/
{(descriptive_representation_item.description = 'right')
(descriptive_representation_item.description = 'left')}

5.1.11.9.5 expanded_area_ratio

AIM element: ratio_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.108
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('propeller moulded form design parameter', 'expanded area ratio', ratio_measure)/

5.1.11.9.6 hub_diameter_ratio

AIM element: ratio_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.108
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('propeller moulded form design parameter', 'hub diameter ratio', ratio_measure)/

5.1.11.9.7 moulded_form_displacement

AIM element: volume_measure
Source: ISO 10303-43
Rules: 5.2.4.122, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('propeller moulded form design parameter', 'moulded form displacement', volume_measure)/

5.1.11.9.8 moulded_form_outer_surface

AIM element: area_measure
Source: ISO 10303-43
Rules: 5.2.4.122, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('propeller moulded form design parameter', 'moulded form outer surface', area_measure)/

5.1.11.9.9 nominal_design_pitch_ratio

AIM element: ratio_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.108
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('propeller moulded form design parameter', 'nominal design pitch ratio', ratio_measure)/

5.1.11.9.10 number_of_propeller_blades

AIM element: count_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.108
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('propeller moulded form design parameter', 'number of propeller blades', count_measure)/

5.1.11.9.11 propeller_diameter

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.108
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('propeller moulded form design parameter', 'propeller diameter', positive_length_measure)/

5.1.11.9.12 rake

AIM element: plane_angle_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.108
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('propeller moulded form design parameter', 'rake', plane_angle_measure)/

5.1.11.9.13 skew

AIM element: plane_angle_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.108
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('propeller moulded form design parameter', 'skew', plane_angle_measure)/

5.1.11.9.14 thickness_at_0_7_radius

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.108
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('propeller moulded form design parameter', 'thickness at 0 7 radius', positive_length_measure)/

5.1.11.9.15 type_of_propeller_blades

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.108
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('propeller moulded form design parameter', 'type of propeller blades')/
{(descriptive_representation_item.description = 'fixed blade')
(descriptive_representation_item.description = 'adjustable blade')
(descriptive_representation_item.description = 'controllable blade')}

5.1.11.9.16 type_of_propulsion

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.108
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('propeller moulded form design parameter', 'type of propulsion')/
{(descriptive_representation_item.description = 'water jet')
(descriptive_representation_item.description = 'paddle wheel')
(descriptive_representation_item.description = 'vertical axis propeller')
(descriptive_representation_item.description = 'screw propeller')
(descriptive_representation_item.description = 'ducted propeller')
(descriptive_representation_item.description = 'thruster')
(descriptive_representation_item.description = 'user defined')}

5.1.11.9.17 version_id

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.4)

5.1.11.9.18 propeller_moulded_form_design_parameter to moulded_form (as defined_for)

NOTE Attribute inherited from supertype Moulded_form_characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.5)

5.1.11.9.19 propeller_moulded_form_design_parameter to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.6)

5.1.11.9.20 propeller_moulded_form_design_parameter to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.7)

5.1.11.9.21 propeller_moulded_form_design_parameter to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.8)

5.1.11.9.22 propeller_moulded_form_design_parameter to propeller_location (as location_of_the_propeller_at_the_ship_hull)

AIM element: PATH
Rules: 5.2.4.74
Reference path: /PROP_DEF_TO_REP/
{/CLASS_ID(representation, 'propeller location')/}

5.1.11.10 RUDDER_MOULDED_FORM_DESIGN_PARAMETER

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.86
Reference path: {[/CLASS(property_definition, 'rudder moulded form design parameter', 'moulded form characteristics definition')/]
[/CLASS(property_definition, 'moulded form characteristics definition', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.11.10.1 description

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.1)

5.1.11.10.2 moulded_form_displacement

AIM element: volume_measure
Source: ISO 10303-43
Rules: 5.2.4.123, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('rudder moulded form design parameter', 'moulded form displacement', volume_measure)/

5.1.11.10.3 moulded_form_outer_surface

AIM element: area_measure
Source: ISO 10303-43
Rules: 5.2.4.123, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('rudder moulded form design parameter', 'moulded form outer surface', area_measure)/

5.1.11.10.4 projected_rudder_area

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.109
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('rudder moulded form design parameter', 'projected rudder area', positive_length_measure)/

5.1.11.10.5 rudder_height

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.109
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('rudder moulded form design parameter', 'rudder height', positive_length_measure)/

5.1.11.10.6 rudder_length

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.109
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('rudder moulded form design parameter', 'rudder length', positive_length_measure)/

5.1.11.10.7 rudder_mean_height

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.109
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('rudder moulded form design parameter', 'rudder mean height', positive_length_measure)/

5.1.11.10.8 rudder_mean_length

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.109
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('rudder moulded form design parameter', 'rudder mean length', positive_length_measure)/

5.1.11.10.9 rudder_thickness

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.109
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('rudder moulded form design parameter', 'rudder thickness', positive_length_measure)/

5.1.11.10.10 type_of_the_rudder

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.109
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('rudder moulded form design parameter', 'type of the rudder')/
{(descriptive_representation_item.description = 'balanced')
(descriptive_representation_item.description = 'unbalanced')}

5.1.11.10.11 version_id

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.4)

5.1.11.10.12 rudder_moulded_form_design_parameter to moulded_form (as defined_for)

NOTE Attribute inherited from supertype Moulded_form_characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.5)

5.1.11.10.13 rudder_moulded_form_design_parameter to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.6)

5.1.11.10.14 rudder_moulded_form_design_parameter to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.7)

5.1.11.10.15 rudder_moulded_form_design_parameter to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.8)

5.1.11.10.16 rudder_moulded_form_design_parameter to centre_location (as rudder_location)

AIM element: PATH
Rules: 5.2.4.114, 5.2.4.109
Reference path: property_definition
represented_definition = property_definition
represented_definition <-
property_definition_representation.definition
property_definition_representation
property_definition_representation.used_representation ->
/REP_ITEM('rudder location')/
compound_representation_item
{/CLASS_ID(compound_representation_item, 'centre_location')/}

5.1.11.11 SHIP_OVERALL_DIMENSIONS

AIM element: product_definition
Source: ISO 10303-41
Rules: 5.2.4.140
Reference path: {[/CLASS(product_definition, 'ship overall dimensions', 'general characteristics definition')/]
[/CLASS(product_definition, 'general characteristics definition', 'definition')/]
[/CLASS(product_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(product_definition, 'versionable object')/]}

5.1.11.11.1 description

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: product_definition.description
Source: ISO 10303-41

5.1.11.11.2 overall_breadth

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.132
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('ship overall dimensions', 'overall breadth', positive_length_measure)/

5.1.11.11.3 overall_depth

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.132
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('ship overall dimensions', 'overall depth', positive_length_measure)/

5.1.11.11.4 overall_length

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.132
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('ship overall dimensions', 'overall length', positive_length_measure)/

5.1.11.11.5 stem_overhang

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.132
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('ship overall dimensions', 'stem overhang', positive_length_measure)/

5.1.11.11.6 stern_overhang

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.132
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('ship overall dimensions', 'stern overhang', positive_length_measure)/

5.1.11.11.7 version_id

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(product_definition)/

5.1.11.11.8 ship_overall_dimensions to ship (as defined_for)

NOTE Attribute inherited from supertype General_characteristics_definition (see 5.1.4.4).

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product, 'ship')/}

5.1.11.11.9 ship_overall_dimensions to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype General_characteristics_-definition (see 5.1.4.4).

AIM element: PATH
Source: ISO 10303-41
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.11.11.10 ship_overall_dimensions to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype General_characteristics_-definition (see 5.1.4.4).

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.11.11.11 ship_overall_dimensions to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype General_characteristics_-definition (see 5.1.4.4).

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.11.12 THRUSTER_MOULDED_FORM_DESIGN_PARAMETER

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.87
Reference path: {[/CLASS(property_definition, 'thruster moulded form design parameter', 'moulded form characteristics definition')/]
[/CLASS(property_definition, 'moulded form characteristics definition', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.11.12.1 description

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.1)

5.1.11.12.2 moulded_form_displacement

AIM element: volume_measure
Source: ISO 10303-43
Rules: 5.2.4.124, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('thruster moulded form design parameter', 'moulded form displacement', volume_measure)/

5.1.11.12.3 moulded_form_outer_surface

AIM element: area_measure
Source: ISO 10303-43
Rules: 5.2.4.124, 5.2.4.114
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('thruster moulded form design parameter', 'moulded form outer surface', area_measure)/

5.1.11.12.4 thruster_location

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.112
Reference path: /PROP_DEF_TO_DESC_REP_ITEM('thruster moulded form design parameter', 'thruster location')/
{(descriptive_representation_item.description = 'bow')
(descriptive_representation_item.description = 'stern')}

5.1.11.12.5 thruster_tunnel_diameter

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.112
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('thruster moulded form design parameter', 'thruster tunnel diameter', positive_length_measure)/

5.1.11.12.6 thruster_tunnel_max_length

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.112
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('thruster moulded form design parameter', 'thruster tunnel max length', positive_length_measure)/

5.1.11.12.7 thruster_tunnel_min_length

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.112
Reference path: /PROP_DEF_TO_VAL_REP_ITEM('thruster moulded form design parameter', 'thruster tunnel min length', positive_length_measure)/

5.1.11.12.8 version_id

NOTE Attribute inherited from supertype Versionable_object (5.1.9.17).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.4)

5.1.11.12.9 thruster_moulded_form_design_parameter to moulded_form (as defined_for)

NOTE Attribute inherited from supertype Moulded_form_characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.5)

5.1.11.12.10 thruster_moulded_form_design_parameter to centre_location (as geometric_thruster_location)

AIM element: PATH
Rules: 5.2.4.114, 5.2.4.112
Reference path: property_definition
represented_definition = property_definition
represented_definition <-
property_definition_representation.definition
property_definition_representation
property_definition_representation.used_representation ->
/REP_ITEM('geometric thruster location')/
compound_representation_item
{/CLASS_ID(compound_representation_item, 'centre_location')/}

5.1.11.12.11 thruster_moulded_form_design_parameter to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.6)

5.1.11.12.12 thruster_moulded_form_design_parameter to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.7)

5.1.11.12.13 thruster_moulded_form_design_parameter to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supertype Moulded_form_-characteristics_definition (see 5.1.11.7).

AIM element: /SUPERTYPE(Moulded_form_characteristics_definition)/ (see 5.1.11.7.8)

5.1.11.12.14 thruster_moulded_form_design_parameter to propeller_-moulded_form_design_parameter (as thruster_propeller_parameter)

AIM element: PATH
Rules: 5.2.4.88
Reference path: property_definition <-
property_definition_relationship.relating_property_definition
property_definition_relationship
{property_definition_relationship.name = 'thruster propeller parameter'}
property_definition_relationship
property_definition_relationship.related_property_definition ->
property_definition=>
property_definition
{/CLASS_ID(property_definition, 'propeller moulded form design parameter')/}

5.1.12 ship_general_characteristics UoF

5.1.12.1 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’)/]}

5.1.12.1.1 description

NOTE Attribute inherited from supertype Shiptype (see 5.1.12.11).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.1)

5.1.12.1.2 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 = ‘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 liquefied gas’)
(product_category.name = ‘Barge for oil’)
(product_category.name = ‘Barge pontoon’)
(product_category.name = ‘LNG carrier’)
(product_category.name = ‘LPG carrier’)
(product_category.name = ‘user defined’)}

5.1.12.1.3 user_def_function

NOTE Attribute inherited from supertype Shiptype (see 5.1.12.11).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.2)

5.1.12.1.4 version_id

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.3)

5.1.12.1.5 carrier to ship (as defined_for)

NOTE Attribute inherited from supertype Shiptype (see 5.1.12.11).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.4)

5.1.12.1.6 carrier to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11)

5.1.12.2 CLASS_AND_STATUTORY_DESIGNATION

AIM element: product_definition
Source: ISO 10303-41
Rules: 5.2.4.28
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')/]}

5.1.12.2.1 class_number

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.93
Reference path: /PROD_DEF_TO_DESC_REP_ITEM('class and statutory designation', 'class number')/

5.1.12.2.2 description

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: product_definition.description
Source: ISO 10303-41

5.1.12.2.3 version_id

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(product_definition)/

5.1.12.2.4 class_and_statutory_designation to ship (as defined_for)

NOTE Attribute inherited from supertype General_characteristics_definition (see 5.1.4.4).

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product,'ship')/}

5.1.12.2.5 class_and_statutory_designation to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Source: ISO 10303-41
Rules: 5.2.4.45, 5.2.4.71
Reference path: product_definition
identification_item = product_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.12.2.6 class_and_statutory_designation to class_notation (as the_class)

AIM element: PATH
Rules: 5.2.4.57
Reference path: /PROD_DEF_PROP_DEF/
{/CLASS_ID(property_definition, 'class notation')/}

5.1.12.2.7 class_and_statutory_designation to regulation (as the_statutory)

AIM element: PATH
Rules: 5.2.4.64
Reference path: /PROD_DEF_PROP_DEF/
{/CLASS_ID(property_definition, 'regulation')/}

5.1.12.3 CLASS_NOTATION

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.78
Reference path: /ROOT_CLASS(property_definition, 'class notation')/

5.1.12.3.1 approval_required_for_heavy_cargo

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.126
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')}

5.1.12.3.2 approval_required_for_oil_cargo

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.129
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‘)}

5.1.12.3.3 approval_required_loading_unloading_aground

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.129
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‘)}

5.1.12.3.4 approval_required_loading_unloading_grabs

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.129
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‘)}

5.1.12.3.5 class_notations_hull

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.29
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘class notation’, ‘class notations hull’)/

5.1.12.3.6 class_notations_machinery

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.29
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘class notation’, ‘class notations machinery’) /

5.1.12.3.7 class_society

AIM element: organization_assignment.assigned_organization
Source: ISO 10303-41
Rules: 5.2.4.79
Reference path: /ORG_ASSGN(property_definition,‘class society’)/

5.1.12.3.8 ice_class_notation

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.126
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘class notation’, ‘ice class notation‘)/

5.1.12.3.9 service_area

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.129
Reference path: /PROP_DEF_TO_DESC_REP_ITEM(‘class notation’, ‘service area‘)/

5.1.12.3.10 service_factor

AIM element: count_measure
Rules: 5.2.4.114, 5.2.4.126
Reference path: /PROP_DEF_TO_VAL_REP_ITEM(‘class notation’, ‘service factor‘, count_measure)/

5.1.12.4 CLASS_PARAMETERS

AIM element: product_definition
Source: ISO 10303-41
Rules: 5.2.4.30
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’)/]}

5.1.12.4.1 block_coefficient_class

AIM element: ratio_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.130
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘class parameters’, ‘block coefficient class‘, ratio_measure)/

5.1.12.4.2 description

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: product_definition.description
Source: ISO 10303-41

5.1.12.4.3 design_speed_ahead

AIM element: context_dependent_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.130
Reference path: /PROD_DEF_TO_SPECIAL_VAL_REP_ITEM('class parameters', 'design speed ahead', 'speed unit')/

5.1.12.4.4 design_speed_astern

AIM element: context_dependent_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.130
Reference path: /PROD_DEF_TO_SPECIAL_VAL_REP_ITEM('class parameters', 'design speed astern', 'speed unit')/

5.1.12.4.5 length_class

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.130
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('class parameters', 'length class', positive_length_measure)/

5.1.12.4.6 length_solas

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.130
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('class parameters', 'length solas', positive_length_measure)/

5.1.12.4.7 scantlings_draught

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.130
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('class parameters', 'scantlings draught', positive_length_measure)/

5.1.12.4.8 version_id

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(product_definition)/

5.1.12.4.9 class_parameters to ship (as defined_for)

NOTE Attribute inherited from supertype General_characteristics_definition (see 5.1.4.4).

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product,'ship')/}

5.1.12.4.10 class_parametersto global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Source: ISO 10303-41
Rules: 5.2.4.45, 5.2.4.70
Reference path: product_definition
identification_item = product_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.12.4.11 class_parameters to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS('local units')/
unit
unit = derived_unit
derived_unit

5.1.12.4.12 class_parameters to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.12.5 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’)/]}

5.1.12.5.1 description

NOTE Attribute inherited from supertype Shiptype (see 5.1.12.11).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.1)

5.1.12.5.2 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’)}

5.1.12.5.3 user_def_function

NOTE Attribute inherited from supertype Shiptype (see 5.1.12.11)).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.2)

5.1.12.5.4 version_id

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.3)

5.1.12.5.5 navy_ship to ship (as defined_for)

NOTE Attribute inherited from supertype Shiptype (see 5.1.12.11).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.4)

5.1.12.5.6 navy_ship to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.5)

5.1.12.6 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')/]}

5.1.12.6.1 description

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: product_definition.description
Source: ISO 10303-41

5.1.12.6.2 managing_company

AIM element: organization_assignment.assigned_organization
Source: ISO 10303-41
Reference path: /ORG_ASSGN(product_definition,'managing company')/

5.1.12.6.3 ordering_company

AIM element: organization_assignment.assigned_organization
Source: ISO 10303-41
Reference path: /ORG_ASSGN(product_definition,‘ordering company‘)/

5.1.12.6.4 owner_approval

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.127
Reference path: /PROD_DEF_TO_DESC_REP_ITEM('owner designation', 'owner approval')/

5.1.12.6.5 owning_company

AIM element: organization_assignment.assigned_organization
Source: ISO 10303-41
Reference path: /ORG_ASSGN(product_definition,‘owning company‘)/

5.1.12.6.6 version_id

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(product_definition)/

5.1.12.6.7 owner_designation to ship (as defined_for)

NOTE Attribute inherited from supertype General_characteristics_definition (see 5.1.4.4).

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product,’ship’)/}

5.1.12.6.8 owner_designation to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Source: ISO 10303-41
Rules: 5.2.4.45, 5.2.4.70
Reference path: product_definition
identification_item= product_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.12.7 PRINCIPAL_CHARACTERISTICS

AIM element: product_definition
Source: ISO 10303-41
Rules: 5.2.4.55
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')/]}

5.1.12.7.1 block_coefficient

AIM element: ratio_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.128
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('principal characteristics', 'block coefficient', ratio_measure)/

5.1.12.7.2 description

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: product_definition.description
Source: ISO 10303-41

5.1.12.7.3 design_deadweight

AIM element: mass_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.128
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('principal characteristics', 'design deadweight', mass_measure)/

5.1.12.7.4 design_draught

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.128
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('principal characteristics', 'design draught', positive_length_measure)/

5.1.12.7.5 length_between_perpendiculars

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.131
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('principal characteristics', 'length

between perpendiculars', positive_length_measure)/

5.1.12.7.6 max_draught_at_ap

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.128
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('principal characteristics', 'max draught at ap', positive_length_measure)/

5.1.12.7.7 max_draught_at_fp

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.128
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('principal characteristics', 'max draught at fp', positive_length_measure)/

5.1.12.7.8 min_draught_at_ap

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.128
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('principal characteristics', 'min draught at ap', positive_length_measure)/

5.1.12.7.9 min_draught_at_fp

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.128
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('principal characteristics', 'min draught at fp', positive_length_measure)/

5.1.12.7.10 moulded_breadth

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.131
Reference path: /PROD_DEF_TO_VAL_REP_ITEM('principal characteristics', 'moulded breadth', positive_length_measure)/

5.1.12.7.11 moulded_depth

AIM element: positive_length_measure
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.131
Reference path: /PROD_DEF_TO_VAL_REP_ITEM(‘principal characteristics’, ‘moulded depth‘, positive_length_measure)/

5.1.12.7.12 version_id

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(product_definition)/

5.1.12.7.13 principal_characteristics to ship (as defined_for)

NOTE Attribute inherited from supertype General_characteristics_definition (see 5.1.4.4).

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product,’ship’)/}

5.1.12.7.14 principal_characteristics to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Source: ISO 10303-41
Rules: 5.2.4.45, 5.2.4.70
Reference path: product_definition
identification_item = product_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.12.7.15 principal_characteristics to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS(‘local units’)/
unit
unit = derived_unit
derived_unit

5.1.12.7.16 principal_characteristics to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Reference path: /PROD_DEF_TO_UNITS('local units')/
unit
unit = named_unit
named_unit

5.1.12.8 REGULATION

AIM element: property_definition
Source: ISO 10303-41
Reference path: {/ROOT_CLASS(property_definition, 'regulation')/}

5.1.12.8.1 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')/}

5.1.12.8.2 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')/}

5.1.12.8.3 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')/}

5.1.12.9 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’)/]}

5.1.12.9.1 description

NOTE Attribute inherited from supertype Shiptype (see 5.1.12.11).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.1)

5.1.12.9.2 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’}

5.1.12.9.3 user_def_function

NOTE Attribute inherited from supertype Shiptype (see 5.1.12.11).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.2)

5.1.12.9.4 version_id

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.3)

5.1.12.9.5 research_ship to ship (as defined_for)

NOTE Attribute inherited from supertype Shiptype (see 5.1.12.11).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.4)

5.1.12.9.6 research_ship to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.5)

5.1.12.10 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')/]}

5.1.12.10.1 call_sign

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.56
Reference path: /HAS_ID_1_ROLE(product_definition, 'call sign')/

5.1.12.10.2 description

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: product_definition.description
Source: ISO 10303-41

5.1.12.10.3 flag_state

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.58
Reference path: /HAS_ID_1_ROLE(product_definition, 'flag state')/

5.1.12.10.4 port_of_registration

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.63
Reference path: /HAS_ID_1_ROLE(product_definition, 'port of registration')/

5.1.12.10.5 ship_identification

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.138
Reference path: /HAS_ID_2_ROLES(product_definition, 'IMO number', 'pennant hull number')/

5.1.12.10.6 ship_name

AIM element: product_definition.name
Source: ISO 10303-41
Reference path: /NAME_ASSGN(product_definition)/

5.1.12.10.7 version_id

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(product_definition)/

5.1.12.10.8 ship_designation to ship (as defined_for)

NOTE Attribute inherited from supertype General_characteristics_definition (see 5.1.4.4).

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product,'ship')/}

5.1.12.10.9 ship_designation to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Source: ISO 10303-41
Rules: 5.2.4.45, 5.2.4.70
Reference path: product_definition
identification_item= product_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.12.11 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 5.1.12.1)
#2: /SUBTYPE(navy_ship)/ (see 5.1.12.5)
#3: /SUBTYPE(research_ship)/ (see 5.1.12.9)
#4: /SUBTYPE(working_ship)/ (see 5.1.12.13)

5.1.12.11.1 description

AIM element: product_category.description
Source: ISO 10303-41
Reference path: product_related_product_category <=
product_category
product_category.description

5.1.12.11.2 user_def_function

NOTE Attribute inherited from supertype Function_definition (see 5.1.4.3).

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

5.1.12.11.3 version_id

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(product_related_product_category)/

5.1.12.11.4 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’)/}

5.1.12.11.5 Shiptype to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.72
Reference path: product_related_product_category
identification_item = product_related_product_category <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.12.12 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’)/]}

5.1.12.12.1 description

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: product_definition.description
Source: ISO 10303-41

5.1.12.12.2 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’)}}

5.1.12.12.3 shipyard

AIM element: organization_assignment.assigned_organization
Source: ISO 10303-41
Rules: 5.2.4.65
Reference path: /ORG_ASSGN(product_definition,‘shipyard‘)/

5.1.12.12.4 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’)/

5.1.12.12.5 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

5.1.12.12.6 version_id

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(product_definition)/

5.1.12.12.7 shipyard_designation to ship (as defined_for)

NOTE Attribute inherited from supertype General_characteristics_definition (see 5.1.4.4).

AIM element: PATH
Reference path: /PROD_DEF_PRODUCT/
{/CLASS_ID(product,’ship’)/}

5.1.12.12.8 shipyard_designation to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: PATH
Source: ISO 10303-41
Rules: 5.2.4.45, 5.2.4.70
Reference path: product_definition
identification_item= product_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.12.13 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’)/]}

5.1.12.13.1 description

NOTE Attribute inherited from supertype Shiptype (see 5.1.12.11).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.1)

5.1.12.13.2 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’)}

5.1.12.13.3 user_def_function

NOTE Attribute inherited from supertype Shiptype (see 5.1.12.11).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.2)

5.1.12.13.4 version_id

NOTE Attribute inherited from supertype Versionable_object (See 5.1.9.17).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.3)

5.1.12.13.5 working_ship to ship (as defined_for)

NOTE Attribute inherited from supertype Shiptype (see 5.1.12.11).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.4)

5.1.12.13.6 working_ship to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1).

AIM element: /SUPERTYPE(Shiptype)/ (see 5.1.12.11.5)

5.1.13 ship_measures UoF

5.1.13.1 CENTRE_LOCATION

AIM element: compound_representation_item
Source: ISO 10303-43
Rules: 5.2.4.114, 5.2.4.25
Reference path: /ROOT_CLASS(compound_representation_item, ‘centre location’)/

5.1.13.1.1 longitudinal_location

AIM element: length_measure
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}

5.1.13.1.2 transversal_location

AIM element: length_measure
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}

5.1.13.1.3 vertical_location

AIM element: length_measure
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}

5.1.13.2 DERIVED_UNIT

NOTE See L.7 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]}])

5.1.13.3 NAMED_UNIT

NOTE See L.7 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)]

5.1.13.4 PRECISION

AIM element: uncertainty_measure_with_unit
Source: ISO 10303-43

5.1.13.4.1 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]}

5.1.14 ship_moulded_form UoF

5.1.14.1 MOULDED_FORM

AIM element: product_definition
Source: ISO 10303-41
Reference path: {[/CLASS(product_definition, 'moulded form', 'item')/]
[/CLASS(product_definition, 'item', 'definable object')/]
[/ROOT_CLASS(product_definition, 'definable object')/]}

5.1.14.1.1 description

NOTE Attribute inherited from supertype Item (see 5.1.8.3)

AIM element: product_definition.description
Source: ISO 10303-41

5.1.14.1.2 name

NOTE Attribute inherited from supertype Item (see 5.1.8.3)

AIM element: product_definition.name
Source: ISO 10303-41
Reference path: /NAME_ASSGN(product_definition)/

5.1.14.1.3 moulded_form to external_reference (as documentation)

NOTE Attribute inherited from supertype Item (see 5.1.8.3)

#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')/}

5.1.14.1.4 moulded_form to global_id (as id)

NOTE Attribute inherited from supertype Item (see 5.1.8.3) or Item_structure(see 5.1.8.5)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.70
Reference path: product_definition
identification_item = product_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.14.1.5 moulded_form to ship (as ship_context)

NOTE Attribute inherited from supertype Item (see 5.1.8.3)

AIM element: PATH
Reference path: product_definition
product_definition.formation->
product_definition_formation
{product_definition_formation.id= 'moulded form'}
product_definition_formation.of_product ->
product
{/CLASS_ID(product, 'ship')/}

5.1.14.2 MOULDED_FORM_BOUNDARY_RELATIONSHIP

AIM element: product_definition_relationship
Source: ISO 10303-41
Reference path: {[/CLASS(product_definition_relationship, 'moulded form boundary relationship', 'moulded form relationship')/]
[/CLASS(product_definition_relationship, 'moulded form relationship', 'item relationship')/]
[/CLASS(product_definition_relationship, 'item relationship', 'definable object')/]
[/ROOT_CLASS(product_definition_relationship, 'definable object')/]
[/CLASS(product_definition_relationship, 'item relationship', 'versionable object')/]
[/ROOT_CLASS(product_definition_relationship, 'versionable object')/]}

5.1.14.2.1 description

NOTE Attribute inherited from supertype Moulded_form_relationship (see 5.1.14.5).

AIM element: product_definition_relationship.description
Source: ISO 10303-41

5.1.14.2.2 version_id

NOTE Attribute inherited from supertype Moulded_form_relationship (see 5.1.14.5).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(product_definition_relationship)/

5.1.14.2.3 moulded_form_boundary_relationship to external_instance_-reference (as external_item_1)

NOTE Attribute inherited from supertype Moulded_form_relationship (see 5.1.14.5).

AIM element: PATH
Source: ISO 10303-41
Reference path: product_definition_relationship
{product_definition_relationship. relating_product_definition ->
product_definition
[/CLASS_ID(product_definition, 'moulded form')/]
[/EXT_INST_REF(product_definition, 'ship moulded form schema', 'moulded form')/]}

5.1.14.2.4 moulded_form_boundary_relationship to external_instance_-reference (as external_item_2)

NOTE Attribute inherited from supertype Moulded_form_relationship (see 5.1.14.5).

AIM element: PATH
Source: ISO 10303-41
Reference path: product_definition_relationship
{product_definition_relationship. related_product_definition ->
product_definition
[/CLASS_ID(product_definition, 'moulded form')/]
[/EXT_INST_REF(product_definition, 'ship moulded form schema', 'moulded form')/]}

5.1.14.2.5 moulded_form_boundary_relationship to global_id (as id)

NOTE Attribute inherited from supertype Moulded_form_relationship (see 5.1.14.5).

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.69
Reference path: product_definition_relationship
identification_item = product_definition_relationship <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.14.2.6 moulded_form_boundary_relationship to moulded_form (as item_-1)

NOTE Attribute inherited from supertype Moulded_form_relationship (see 5.1.14.5).

AIM element: PATH
Rules: 5.2.4.68
Reference path: product_definition_relationship
product_definition_relationship.relating_product_definition ->
product_definition
{/CLASS_ID(product_definition, 'moulded form')/}

5.1.14.2.7 moulded_form_boundary_relationship to moulded_form (as item_-2)

NOTE Attribute inherited from supertype Moulded_form_relationship (see 5.1.14.5).

AIM element: PATH
Rules: 5.2.4.68
Reference path: product_definition_relationship
product_definition_relationship.related_product_definition ->
product_definition
{/CLASS_ID(product_definition, 'moulded form')/}

5.1.14.3 MOULDED_FORM_DESIGN_DEFINITION

AIM element: product_definition_shape
Source: ISO 10303-41
Reference path: {[/CLASS(product_definition_shape, 'moulded form 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')/]}

5.1.14.3.1 description

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: property_definition.description
Source: ISO 10303-41

5.1.14.3.2 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(product_definition_shape)/

5.1.14.3.3 moulded_form_design_definition to moulded_form_boundary_-relationship (as borders)

AIM element: PATH
Reference path: product_definition_shape =>
property_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
{/CLASS_ID(product_definition_relationship, ‘moulded form boundary relationship’)/}

5.1.14.3.4 moulded_form_design_definition to plane (as borders)

AIM element: PATH
Reference path: product_definition_shape <-
shape_aspect.of_shape
shape_aspect
{shape_aspect.name = 'border'}
shape_aspect = represented_definition <-
/PDR_NAME('border representation')/ ->
representation
representation.items[i] ->
representation_item =>
{representation_item.name = ‘.UNUSED.’}
geometric_representation_item =>
plane

5.1.14.3.5 moulded_form_design_definition to ship_curve (as borders)

AIM element: PATH
Reference path: product_definition_shape <-
shape_aspect.of_shape
shape_aspect
{shape_aspect.name = 'border'}
shape_aspect = represented_definition <-
/PDR_NAME('border representation')/ ->
representation
representation.items[i] ->
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘ship curve’)/}

5.1.14.3.6 moulded_form_design_definition to spacing_position (as borders)

AIM element: PATH
Reference path: product_definition_shape <-
shape_aspect.of_shape
shape_aspect
{shape_aspect.name = 'border'}
shape_aspect = represented_definition <-
/PDR_NAME('border representation')/ ->
representation
representation.items[i] ->
compound_representation_item
{/CLASS_ID(compound_representation_item, ‘spacing position’)/}

5.1.14.3.7 moulded_form_design_definition to moulded_form (as defined_for)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Design_definition (see 5.1.4.2).

AIM element: PATH
Reference path: product_definition_shape <=
/PROP_TO_PROD_DEF/
{/CLASS_ID(product_definition, 'moulded form')/}

5.1.14.3.8 moulded_form_design_definition to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Design_definition (see 5.1.4.2).

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.70
Reference path: product_definition_shape
identification_item = product_definition_shape <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.14.3.9 moulded_form_design_definition to derived_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Design_definition (see 5.1.4.2).

AIM element: PATH
Reference path: product_definition_shape <=
property_definition
/PROP_DEF_TO_UNITS(‘local units’)/
unit
unit = derived_unit
derived_unit

5.1.14.3.10 moulded_form_design_definition to named_unit (as local_units)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Design_definition (see 5.1.4.2).

AIM element: PATH
Reference path: product_definition_shape <=
property_definition
/PROP_DEF_TO_UNITS(‘local units’)/
unit
unit = named_unit
named_unit

5.1.14.3.11 moulded_form_design_definition to edge_based_wireframe_shape (as moulded_surface)

AIM element: PATH
Reference path: product_definition_shape <-
shape_aspect.of_shape
shape_aspect
{shape_aspect.name = 'moulded surface'}
shape_aspect = represented_definition <-
/PDR_NAME('surface representation')/ ->
representation=>
shape_representation =>
edge_based_wireframe_shape_representation

5.1.14.3.12 moulded_form_design_definition to moulded_form_shape_-representation (as moulded_surface)

AIM element: PATH
Reference path: product_definition_shape <-
shape_aspect.of_shape
shape_aspect
{shape_aspect.name = 'moulded surface'}
shape_aspect = represented_definition <-
/PDR_NAME('surface representation')/ ->
representation =>
shape_representation
{/CLASS_ID(shape_representation, 'moulded form shape representation')/}

5.1.14.3.13 moulded_form_design_definition to non_manifold_surface_-shape(as moulded_surface)

AIM element: PATH
Reference path: product_definition_shape <-
shape_aspect.of_shape
shape_aspect
{shape_aspect.name = 'moulded surface'}
shape_aspect = represented_definition <-
/PDR_NAME('surface representation')/ ->
representation =>
shape_representation =>
non_manifold_surface_shape_representation

5.1.14.3.14 moulded_form_design_definition to ship_surface (as moulded_-surface)

AIM element: PATH
Reference path: product_definition_shape <-
shape_aspect.of_shape
shape_aspect
{shape_aspect.name = 'moulded surface'}
shape_aspect = represented_definition <-
/PDR_NAME('surface representation')/ ->
representation
representation.items[i] ->
compound_representation_item
{/CLASS_ID(compound_representation_item, ship_surface)/}

5.1.14.3.15 moulded_form_design_definition to edge_based_wireframe_-shape (as representations)

NOTE Attribute inherited from supertype Design_definition (see 5.1.4.2).

AIM element: PATH
Reference path: product_definition_shape <=
property_definition <-
represented_definition <-
/SDR_NAME('moulded form design representation')/
property_definition_representation.used_representation ->
representation =>
shape_representation =>
edge_based_wireframe_shape_representation

5.1.14.3.16 moulded_form_design_definition to non_manifold_surface_shape (as representations)

NOTE Attribute inherited from supertype Design_definition (see 5.1.4.2).

AIM element: PATH
Reference path: product_definition_shape <=
property_definition <-
represented_definition <-
/SDR_NAME('moulded form design representation')/
property_definition_representation.used_representation ->
representation =>
shape_representation =>
non_manifold_surface_shape_representation

5.1.14.3.17 status

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.120
Reference path: product_definition_shape <=
/PROP_DEF_REP_HELP('moulded form design parameters')/
representation
representation.items [i] ->
representation_item =>
{representation_item.name = 'status'}
descriptive_representation_item
{(descriptive_representation_item.description = 'partial')
(descriptive_representation_item.description = 'complete')}

5.1.14.4 MOULDED_FORM_FUNCTIONAL_DEFINITION

AIM element: property_definition
Source: ISO 10303-41
Rules: 5.2.4.84
Reference path: {[/CLASS(property_definition, 'moulded form functional definition', 'functional definition')/]
[/CLASS(property_definition, 'functional definition', 'definition')/]
[/CLASS(property_definition, 'definition', 'versionable object')/]
[/ROOT_CLASS(property_definition, 'versionable object')/]}

5.1.14.4.1 description

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17) through supetype Functional_definition (see 5.1.4.3), which inherits from supertype Definition (see 5.1.4.1).

AIM element: property_definition.description
Source: ISO 10303-41

5.1.14.4.2 the_function

AIM element: descriptive_representation_item.description
Source: ISO 10303-45
Rules: 5.2.4.151, 5.2.4.103
Reference path: /PROP_DEF_REP_HELP('moulded form function parameters')/
representation
representation.items [i] ->
representation_item =>
{representation_item.name = 'function'}
descriptive_representation_item
{(descriptive_representation_item.description = 'ship hull')
(descriptive_representation_item.description = 'propeller')
(descriptive_representation_item.description = 'rudder')
(descriptive_representation_item.description = 'bulbous bow')
(descriptive_representation_item.description = 'bulbous stern')
(descriptive_representation_item.description = 'appendage')
(descriptive_representation_item.description = 'thruster')
(descriptive_representation_item.description = 'keel')
(descriptive_representation_item.description = 'transom')
(descriptive_representation_item.description = 'double bottom')
(descriptive_representation_item.description = 'double ship hull')
(descriptive_representation_item.description = 'deck')
(descriptive_representation_item.description = 'superstructure')
(descriptive_representation_item.description = 'horizontal girder')
(descriptive_representation_item.description = 'longitudinal girder')
(descriptive_representation_item.description = 'longitudinal bulkhead')
(descriptive_representation_item.description = 'transverse bulkhead')
(descriptive_representation_item.description = 'frame')
(descriptive_representation_item.description = 'pressure hull')

(descriptive_representation_item.description = 'non-structural bulkhead')
(descriptive_representation_item.description = 'grating')
(descriptive_representation_item.description = 'user defined')}

5.1.14.4.3 user_def_function

NOTE Attribute inherited from supertype Functional_definition (see 5.1.4.3).

AIM element: descriptive_representation_item.description
Source: ISO 10303-41
Rules: 5.2.4.121
Reference path: /PROP_DEF_REP_HELP('moulded form function parameters')/
representation
representation.items [i] ->
representation_item =>
{representation_item.name = 'user def function'}
descriptive_representation_item
descriptive_representation_item.description

5.1.14.4.4 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17) through supetype Functional_definition (see 5.1.4.3), which inherits from supertype Definition (see 5.1.4.1).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(property_definition)/

5.1.14.4.5 moulded_form_functional_definition to moulded_form (as defined_for)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Functional_definition (see 5.1.4.3).

AIM element: PATH
Reference path: /PROP_TO_PROD_DEF/
{/CLASS_ID(product_definition, 'moulded form')/}

5.1.14.4.6 moulded_form_functional_definition to global_id (as id)

NOTE Attribute inherited from supertype Definition (see 5.1.4.1) through supetype Functional_definition (see 5.1.4.3).

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.89
Reference path: property_definition
identification_item = product_definition <-
property_definition
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.14.5 MOULDED_FORM_RELATIONSHIP

AIM element: product_definition_relationship
Source: ISO 10303-41
Rules: 5.2.4.67
Reference path: {[/CLASS(product_definition_relationship, 'moulded form relationship', 'item relationship')/]
[/CLASS(product_definition_relationship, 'item relationship', 'definable object')/]
[/ROOT_CLASS(product_definition_relationship, 'definable object')/]
[/CLASS(product_definition_relationship, 'item relationship', 'versionable object')/]
[/ROOT_CLASS(product_definition_relationship, 'versionable object')/]}

5.1.14.5.1 description

NOTE Attribute inherited from supertype Versionable_object(see 5.1.9.17) through the supertype Item_relationship (see 5.1.8.4).

AIM element: product_definition_relationship.description
Source: ISO 10303-41

5.1.14.5.2 version_id

NOTE Attribute inherited from supertype Item_relationship (see 5.1.8.4).

AIM element: applied_identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(product_definition_relationship)/

5.1.14.5.3 moulded_form_relationship to external_instance_reference (as external_item_1)

NOTE Attribute inherited from supertype Item_relationship (see 5.1.8.4).

AIM element: PATH
Reference path: product_definition_relationship
{product_definition_relationship. relating_product_definition ->
product_definition
[/CLASS_ID(product_definition, 'moulded form')/]
[/EXT_INST_REF(product_definition, 'ship moulded form schema', 'moulded form')/]}

5.1.14.5.4 moulded_form_relationship to external_instance_reference (as external_item_2)

NOTE Attribute inherited from supertype Item_relationship (see 5.1.8.4).

AIM element: PATH
Reference path: product_definition_relationship
{product_definition_relationship. related_product_definition ->
product_definition
[/CLASS_ID(product_definition, 'moulded form')/]
[/EXT_INST_REF(product_definition, 'ship moulded form schema', 'moulded form')/]}

5.1.14.5.5 moulded_form_relationship to global_id (as id)

NOTE Attribute inherited from supertype Item_relationship (see 5.1.8.4).

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.69
Reference path: product_definition_relationship
identification_item = product_definition_relationship <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.14.5.6 moulded_form_relationship to moulded_form (as item_1)

NOTE Attribute inherited from supertype Item_relationship (5.1.8.4).

AIM element: PATH
Rules: 5.2.4.68
Reference path: product_definition_relationship
product_definition_relationship.relating_product_definition ->
product_definition
{/CLASS_ID(product_definition, 'moulded form')/}

5.1.14.5.7 moulded_form_relationship to moulded_form (as item_2)

NOTE Attribute inherited from supertype Item_relationship (see 5.1.8.4).

AIM element: PATH
Rules: 5.2.4.68
Reference path: product_definition_relationship
product_definition_relationship.related_product_definition ->
product_definition
{/CLASS_ID(product_definition, 'moulded form')/}

5.1.14.6 MOULDED_FORM_REPRESENTATION_ITEM

#1: moulded form representation item is a ship_curve
#2: moulded form representation item is a ship_point
#3: moulded form representation item is a ship_surface

AIM element: #1: /SUBTYPE(ship_curve)/ (see 5.1.2.1)
#2: /SUBTYPE(ship_point)/ (see 5.1.2.3)
#3: /SUBTYPE(ship_surface)/ (see 5.1.2.4)

5.1.14.7 MOULDED_FORM_REPRESENTATION_RELATIONSHIP

AIM element: representation_relationship
Source: ISO 10303-43
Reference path: /ROOT_CLASS(representation_relationship,
'moulded form representation relationship')/

5.1.14.7.1 description

AIM element: representation_relationship.description
Source: ISO 10303-43

5.1.14.7.2 id

AIM element: representation_relationship.name
Source: ISO 10303-43

5.1.14.7.3 moulded_form_representation_relationship to moulded_form_-representation (as rep_1)

AIM element: PATH
Reference path: representation_relationship
representation_relationship.rep_1 ->
representation =>
shape_representation
{CLASS_ID<shape_representation, 'moulded form shape representation'>}

5.1.14.7.4 moulded_form_representation_relationship to moulded_form_-representation (as rep_2)

AIM element: PATH
Reference path: representation_relationship
representation_relationship.rep_2 ->
representation =>
shape_representation
{CLASS_ID<shape_representation, 'moulded form shape representation'>}

5.1.14.8 MOULDED_FORM_SHAPE_REPRESENTATION

#1: Moulded_form_shape_representation is an Offset_table_shape_representation
#2: Moulded_form_shape_representation is a Surface_shape_representation
#3: Moulded_form_shape_representation is a Wireframe_shape_representation

AIM element: #1: /SUBTYPE(Offset_table_shape_representation)/ (see 5.1.10.2)
#2: /SUBTYPE(Surface_shape_representation)/ (see 5.1.15.2)
#3: /SUBTYPE(Wireframe_shape_representation)/ (see 5.1.16.4)

5.1.14.8.1 moulded_form_representation_id

AIM element: representation.id
Source: ISO 10303-43
Reference path: /ID_ASSGN(representation)/

5.1.14.8.2 moulded_form_shape_representation to symmetry (as moulded_-form_symmetry)

AIM element: PATH
Reference path: shape_representation <=
representation
representation.items[i] ->
representation_item =>
({representation_item.name = 'rotational axis'}
geometric_representation_item =>
placement =>
axis1_placement)
({representation_item.name = 'symmetry plane'}
geometric_representation_item =>
surface =>
elementary_surface =>
plane)

5.1.14.9 PLANAR_SYMMETRY

AIM element: plane
Source: ISO 10303-42

5.1.14.9.1 planar_symmetry to plane (as the_symmetry_plane)

AIM element: IDENTICAL MAPPING

5.1.14.10 ROTATIONAL_SYMMETRY

AIM element: axis1_placement
Source: ISO 10303-42

5.1.14.10.1 rotational_symmetry to axis1_placement (as the_rotational_axis)

AIM element: IDENTICAL MAPPING

5.1.14.11 SHIP_MOULDED_FORM

AIM element: [product_definition]
[group]
Source: ISO 10303-41,ISO 10303-41
Reference path: {[/CLASS(product_definition, 'ship moulded form', ' item structure')/]
[/CLASS(product_definition,' item structure ', 'definable object')/]
[/CLASS(product_definition, 'item structure', 'versionable object')/]
[/CLASS(product_definition, 'ship moulded form', 'item')/]
[/CLASS(product_definition, 'item','definable object')/]
[/ROOT_CLASS(product_definition, 'definable object')/]
[/ROOT_CLASS(product_definition, 'versionable object')/]
[product_definition
/LINK_TO_GROUP(product_definition)/]}

5.1.14.11.1 description

NOTE Attribute inherited from supertype Item (see 5.1.8.3)

AIM element: product_definition.description
Source: ISO 10303-41

5.1.14.11.2 name

NOTE Attribute inherited from supertype Item (see 5.1.8.3)

AIM element: product_definition.name
Source: ISO 10303-41
Reference path: /NAME_ASSGN(product_definition)/

5.1.14.11.3 version_id

NOTE Attribute inherited from supertype Item_structure(see 5.1.8.5)

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(group)/

5.1.14.11.4 ship_moulded_form to external_reference (as documentation)

NOTE Attribute inherited from supertype Item (see 5.1.8.3)

#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')/}

5.1.14.11.5 ship_moulded_form to external_instance_reference (as external_-items)

NOTE Attribute inherited from supertype Item_structure(see 5.1.8.5)

AIM element: PATH
Reference path: group <-
/GROUPS(product_definition, 'item structure')/
([/CLASS_ID(product_definition, 'moulded form')/]
[/EXT_INST_REF(product_definition, 'ship moulded
form schema','moulded form')/])

5.1.14.11.6 ship_moulded_form to external_instance_reference (as external_-relationships)

NOTE Attribute inherited from supertype Item_structure(see 5.1.8.5)

AIM element: PATH
Reference path: group <-
/GROUPS(product_definition_relationship, 'item structure')/
([/CLASS_ID(product_definition_relationship, ‘moulded form relationship')/]
[/EXT_INST_REF(product_definition_relationship, 'ship moulded form schema', 'moulded form relationship')/])

5.1.14.11.7 ship_moulded_form to global_id (as id)

NOTE Attribute inherited from supertype Item (see 5.1.8.3) or Item_structure(see 5.1.8.5)

AIM element: PATH
Rules: 5.2.4.45, 5.2.4.70
Reference path: product_definition
identification_item = product_definition <-
applied_identification_assignment.items[i]
applied_identification_assignment

5.1.14.11.8 ship_moulded_form to moulded_form (as items)

NOTE Attribute inherited from supertype Item_structure(see 5.1.8.5)

AIM element: PATH
Rules: 5.2.4.152
Reference path: group <-
/GROUPS(product_definition, 'item structure')/
{/CLASS_ID(product_definition, 'moulded form')/}

5.1.14.11.9 ship_moulded_form to moulded_form_relationship (as relationships)

NOTE Attribute inherited from supertype Item_structure(see 5.1.8.5)

AIM element: PATH
Rules: 5.2.4.152
Reference path: group <-
/GROUPS(product_definition_relationship, 'item structure')/
{/CLASS_ID(product_definition_relationship, 'moulded form relationship')/}

5.1.14.11.10 ship_moulded_form to ship (as ship_context)

NOTE Attribute inherited from supertype Item (see 5.1.8.3)

AIM element: PATH
Reference path: product_definition
product_definition.formation->
product_definition_formation
{product_definition_formation.id = 'ship moulded form'}
product_definition_formation.of_product ->
product
{/CLASS_ID(product, 'ship')/}

5.1.14.12 SHIP_MOULDED_FORM_REVISION

AIM element: group
Source: ISO 10303-41
Reference path: {[/CLASS(group, 'ship moulded form revision', 'revision with context')/]
[/CLASS(group, 'revision with context', 'revision')/]
[/CLASS(group, 'revision', 'versionable_object')/]
[/ROOT_CLASS(group, 'versionable_object')/]}

5.1.14.12.1 description

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17) through supetype Revision with context (see 5.1.3.14), which inherits from supertype Revision (see 5.1.3.13).

AIM element: group.description
Source: ISO 10303-41
Rules: 5.2.4.139

5.1.14.12.2 name

NOTE Attribute inherited from supertype Revision with context (see 5.1.3.14), which inherits from supertype Revision (see 5.1.3.13).

AIM element: group.name
Source: ISO 10303-41

5.1.14.12.3 version_id

NOTE Attribute inherited from supertype Versionable_object (see 5.1.9.17) through supetype Revision with context (see 5.1.3.14), which inherits from supertype Revision (see 5.1.3.13).

AIM element: identification_assignment.assigned_id
Source: ISO 10303-41
Rules: 5.2.4.163
Reference path: /VERSION_ID(group)/

5.1.14.12.4 ship_moulded_form_revision to ship_moulded_form (as context_of_revision)

NOTE Attribute inherited from supertype Revision with context (see 5.1.3.14).

AIM element: PATH
Rules: 5.2.4.133
Reference path: group <-
/(RELATE_GROUP_2_DO(product_definition, 'ship moulded form')/

5.1.14.12.5 ship_moulded_form_revision to moulded_form_design_definition (as members)

NOTE Attribute inherited from supertype Revision with context (see 5.1.3.14), which inherits from supertype Revision (see 5.1.3.13).

AIM element: PATH
Rules: 5.2.4.52
Reference path: group <-
(/RELATE_GROUP_2_VO(product_definition_shape,'moulded form design definition')/)
(/RELATE_GROUP_2_VO(applied_external_identification_assignment,'external instance reference')/)

5.1.14.13 SYMMETRY

#1: symmetry is planar_symmetry
#2: symmetry is rotational_symmetry

AIM element: #1: /SUBTYPE(planar_symmetry)/ (see 5.1.14.9)
#2: /SUBTYPE(rotational_symmetry)/ (see 5.1.14.10)

5.1.15 surface_representations UoF

5.1.15.1 NON_MANIFOLD_SURFACE_SHAPE

AIM element: non_manifold_surface_shape_representation
Source: ISO 10303-508
Rules: 5.2.4.113

5.1.15.2 SURFACE_SHAPE_REPRESENTATION

AIM element: non_manifold_surface_shape_representation
Source: ISO 10303-508
Rules: 5.2.4.113
Reference Path: {[/CLASS(non_manifold_surface_shape_representation,'surface shape representation','moulded form shape representation')/]
[/ROOT_CLASS(non_manifold_surface_shape_representation,'moulded form

shape representation')/]}

5.1.15.2.1 moulded_form_representation_id

NOTE Attribute inherited from supertype Moulded_form_shape_representation (see 5.1.14.8).

AIM element: /SUPERTYPE(Moulded_form_shape_representation)/ (see 5.1.14.8.1)

5.1.15.2.2 surface_shape_representation to face_based_surface_model (as items)

AIM element: PATH
Reference path: non_manifold_surface_shape_representation <=
shape_representation <=
representation
representation items[i] ->
face_based_surface_model

5.1.15.2.3 surface_shape_representation to symmetry (as moulded_form_-symmetry)

NOTE Attribute inherited from supertype Moulded_form_shape_representation (see 5.1.14.8).

AIM element: /SUPERTYPE(Moulded_form_shape_representation)/ (see 5.1.14.8.2)

5.1.16 wireframe_representations UoF

5.1.16.1 EDGE_BASED_WIREFRAME_SHAPE

AIM element: edge_based_wireframe_shape_representation
Source: ISO 10303-501
Rules: 5.2.4.113

5.1.16.2 KNOT

AIM element: [vertex_point]
[compound_representation_item]
Source: ISO 10303-42, ISO 10303-42
Reference path: {[/ROOT_CLASS(vertex_point, 'knot')/]
[/ROOT_CLASS(compound_representation_item, 'knot')/]}

5.1.16.2.1 knot to ship_curve (as intersecting_ship_curves)

AIM element: PATH
Rules: 5.2.4.32
Reference path: /COMPOUND('intersecting ship curve')/
compound_representation_item
{/CLASS_ID(compound_representation_item, 'ship curve')/}

5.1.16.2.2 knot to vector (as tangent_information)

AIM element: PATH
Reference path: /COMPOUND('tangent vector')/
geometric_representation_item =>
vector

5.1.16.3 SHIP_CURVE_SEGMENT

AIM element: [edge_curve]
[compound_representation_item]
Source: ISO 10303-42, ISO 10303-42
Reference path: {[/ROOT_CLASS(edge_curve, 'ship curve segment')/]
[/ROOT_CLASS(compound_representation_item, 'ship curve segment')/]}

5.1.16.3.1 ship_curve_segment to ship_curve (as part_of_ship_curve)

AIM element: PATH
Rules: 5.2.4.136
Reference path: /COMPOUND('ship curve segment')/
compound_representation_item
{/CLASS_ID(compound_representation_item, 'ship curve')/}

5.1.16.4 WIREFRAME_SHAPE_REPRESENTATION

AIM element: shape_representation
Source: ISO 10303-41
Rules: 5.2.4.113
Reference Path: {[/CLASS(shape_representation,'wireframe shape representation','moulded form shape representation')/]
[/ROOT_CLASS(shape_representation,'moulded form shape representation')/]}

5.1.16.4.1 moulded_form_representation_id

NOTE Attribute inherited from supertype Moulded_form_shape_representation (see 5.1.14.8).

AIM element: /SUPERTYPE(Moulded_form_shape_representation)/ (see 5.1.14.8.1)

5.1.16.4.2 wireframe_shape_representation to ship_curve (as items)

AIM element: PATH
Reference path: shape_representation <=
representation
representation.items[i] ->
representation_item =>
compound_representation_item
{/CLASS_ID(compound_representation_item, 'ship curve')/}

5.1.16.4.3 wireframe_shape_representation to symmetry (as moulded_form_-symmetry)

NOTE Attribute inherited from supertype Moulded_form_shape_representation (see 5.1.14.8).

AIM element: /SUPERTYPE(Moulded_form_shape_representation)/ (see 5.1.14.8.2)

5.2 AIM EXPRESS short listing

This clause specifies the EXPRESS schema that uses elements from the integrated resources and the AICs 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 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 and subtypes apply exclusively to those items that are imported into the AIM.

```
*)
SCHEMA ship_moulded_form_SCHEMA;

   USE FROM aic_non_manifold_surface                    -- ISO 10303-508
   (non_manifold_surface_shape_representation );

    USE FROM aic_edge_based_wireframe                   -- ISO 10303-501
   (edge_based_wireframe_shape_representation );

   USE FROM aic_topologically_bounded_surface           -- ISO 10303-511
   (advanced_face );

  USE FROM action_schema                                -- ISO 10303-41
  (action,
   action_method,
   action_relationship,
   action_request_solution,
   executed_action,
   versioned_action_request);

  USE FROM approval_schema                              -- ISO 10303-41
  (approval,
   approval_date_time,
   approval_person_organization);

  USE FROM application_context_schema                   -- ISO 10303-41
  (application_context,
  application_protocol_definition);

  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,
 externally_defined_item);

USE FROM effectivity_schema                       -- ISO 10303-41
(effectivity,
serial_numbered_effectivity);

REFERENCE FROM geometry_schema                    -- ISO 10303-42
(dummy_gri);

USE FROM geometric_model_schema
(faceted_brep,
geometric_curve_set);

USE FROM geometry_schema                          -- ISO 10303-42
(axis1_placement,
 axis2_placement_2d,
 axis2_placement_3d,
 b_spline_curve,
 b_spline_surface,
 b_spline_curve_with_knots,
 b_spline_surface_with_knots,
 bezier_curve,
 bezier_surface,
 bounded_curve,
 bounded_pcurve,
 bounded_surface_curve,
 cartesian_point,
 circle,
 composite_curve_on_surface,
 conical_surface,
 curve,
 cylindrical_surface,
 degenerate_pcurve,
 degenerate_toroidal_surface,
 direction,
 elementary_surface,
 ellipse,
 evaluated_degenerate_pcurve,
 geometric_representation_item,
 geometric_representation_context,
 hyperbola,
 intersection_curve,
 line,
 oriented_surface,
```

```
  parabola,
  plane,
  point_on_curve,
  point_on_surface,
  quasi_uniform_curve,
  quasi_uniform_surface,
  rational_b_spline_curve,
  rational_b_spline_surface,
  seam_curve,
  spherical_surface,
  surface_of_linear_extrusion,
  surface_of_revolution,
  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_request_assignment,
  action_assignment,
  approval_assignment,
  classification_assignment,
  classification_role,
  date_and_time_assignment,
  document_reference,
  effectivity_assignment,
  external_identification_assignment,
  group_assignment,
  identification_assignment,
  identification_role,
  identification_assignment_relationship,
  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,
 amount_of_substance_unit,
 area_measure,
 celsius_temperature_measure,
 conversion_based_unit,
 context_dependent_unit,
```



```
  count_measure,
  derived_unit,
  electric_current_measure,
  electric_current_unit,
  global_unit_assigned_context,
  length_measure,
  length_measure_with_unit,
  length_unit,
  luminous_intensity_measure,
  luminous_intensity_unit,
  mass_measure,
  mass_measure_with_unit,
  mass_unit,
  measure_with_unit,
  named_unit,
  plane_angle_measure,
  plane_angle_measure_with_unit,
  plane_angle_unit,
  positive_length_measure,
  positive_plane_angle_measure,
  ratio_measure,
  ratio_unit,
  si_unit,
  solid_angle_measure,
  solid_angle_measure_with_unit,
  solid_angle_unit,
  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_relationship,
   product_definition,
   product_definition_relationship,
   product_related_product_category);

  USE FROM product_property_definition_schema        -- ISO 10303-41
```

```
  (characterized_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 10303-41
  (descriptive_representation_item);

  USE FROM representation_schema                     -- ISO 10303-43
  (compound_representation_item,
   global_uncertainty_assigned_context,
   item_defined_transformation,
   list_representation_item,
   mapped_item,
   parametric_representation_context,
   representation,
   representation_item,
   representation_map,
   representation_relationship,
   set_representation_item,
   value_representation_item);

  REFERENCE FROM support_resource_schema             -- ISO 10303-41
  (bag_to_set);

  REFERENCE FROM topology_schema                     -- ISO 10303-42
  (dummy_tri);

  USE FROM topology_schema                           -- ISO 10303-42
   (edge,
   edge_curve,
   edge_loop,
   face,
   face_surface,
   oriented_edge,
   oriented_face,
   poly_loop,
   subface,
    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
aic_edge_based_wireframe	ISO 10303-501
action_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
external_reference_schema	ISO 10303-41
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 moulded form types

5.2.2.1 action_item

An action_item identifies an **product**, **product_definition**, **property_definition**, **product_definition_-relationship**, **product_definition_shape**, **product_related_product_category**, **action_request_-solution,** and **executed_action** to which an **action** may be assigned.

EXPRESS specification:

```
*)
TYPE action_item = SELECT
    (product,
     product_definition,
     property_definition,
     product_definition_relationship,
     product_definition_shape,
     product_related_product_category,
     action_request_solution,
     executed_action);
END_TYPE;
(*
```

5.2.2.2 action_request_item

An **action_request_item** identifies an **action** and **executed_action** to which an **action_request** may be assigned.

EXPRESS specification:

```
*)
TYPE action_request_item = SELECT
    (action,
     executed_action);
END_TYPE;
(*
```

5.2.2.3 approval_item

An **approval_item** identifies a **product_definition**, **product_definition_shape**, **property_definition,** and **product_related_product_category** to which an **approval** may be assigned.

EXPRESS specification:

```
*)
TYPE approval_item = SELECT
    (product_definition,
     product_definition_shape,
     property_definition,
     product_related_product_category);
END_TYPE;
(*
```

5.2.2.4 classification_item

A **classification_item** identifies an **action**, **action_request_solution**, **applied_action_request_assignment**, **approval**, **axis2_placement_3d**, **compound_representation_item**, **external_source**, **document**, **document_reference**, **edge_curve**, **executed_action**, **group**, **identification_assignment_relationship**, **measure_with_unit**, **product,** **product_definition**, **product_definition_relationship**, **product_definition_shape**, **product_related_product_category**, **property_definition**, **property_definition_representation**, **representation**, **representation_relationship**, **shape_representation**, **vertex_point**, and **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,
     axis2_placement_3d,
     compound_representation_item,
     external_source,
     document,
     document_reference,
     edge_curve,
     executed_action,
     group,
     identification_assignment_relationship,
     measure_with_unit,
     product,
     product_definition,
     product_definition_relationship,
     product_definition_shape,
     product_related_product_category,
     property_definition,
     property_definition_representation,
     representation,
     representation_relationship,
     shape_representation,
     vertex_point,
     versioned_action_request);
END_TYPE;
(*
```

5.2.2.5 date_and_time_item

A **date_and_time_item** identifies a **action**, **action_request_solution**, **product_definition**, **executed_action**, and **versioned_action_request** to which a date may be assigned.

EXPRESS specification:

```
*)
TYPE date_and_time_item = SELECT
    (action_request_solution,
     executed_action,
     versioned_action_request,
     product_definition,
     action);
END_TYPE;
(*
```

5.2.2.6 document_reference_item

A **document_reference_item** identifies an **action, product, property_definition,** or **product_definition** to which a document may be assigned.

EXPRESS specification:

```
*)
TYPE document_reference_item = SELECT
    (action,
     product,
     property_definition,
     product_definition);
END_TYPE;
(*
```

5.2.2.7 effectivity_item

A **effectivity_item** identifies a **product_definition**, **property_definition**, **product_definition_shape**, and **product_related_product_category** to which a **effectivity** may be assigned.

EXPRESS specification:

```
*)
TYPE effectivity_item = SELECT
    (product_definition,
     property_definition,
     product_definition_shape,
     product_related_product_category);
END_TYPE;
(*
```

5.2.2.8 external_identification_item

An **external_identification_item** identifies an **action**, **document**, **product**, **product_definition, product_definition_relationship,** and **property_definition** to which an external_identification may be assigned.

EXPRESS specification:

```
*)
TYPE external_identification_item = SELECT
    (action,
     document,
     product,
     product_definition,
     product_definition_relationship,
     property_definition);
END_TYPE;
(*
```


5.2.2.9 group_item

A **group_item** identifies an **approval**, **identification_assignment_relationship, product_definition**, **product_definition_relationship**, and **shape_aspect** to which a group may be assigned.

EXPRESS specification:

```
*)
TYPE group_item = SELECT
    (approval,
     identification_assignment_relationship,
     product_definition,
     product_definition_relationship);
END_TYPE;
(*
```

5.2.2.10 identification_item

An **identification_item** identifies an **action, action_request_solution, executed_action, property_definition**, **product_definition_shape**, **group**, **product**, **product_definition**, **product_definition_relationship**, **product_related_product_category**, **compound_representation_item**, **versioned_action_request**, and **document** to which an identification may be assigned.

EXPRESS specification:

```
*)
TYPE identification_item = SELECT
    (action,
     action_request_solution,
     document,
     executed_action,
     product,
     product_definition,
     property_definition,
     product_definition_shape,
     product_related_product_category,
     compound_representation_item,
     product_definition_relationship,
     group,
     versioned_action_request);
END_TYPE;
(*
```

5.2.2.11 organization_item

An **organization_item** identifies a **document**, **product_definition**, **property_definition** to which an organization may be identified.

EXPRESS specification:

```
*)
TYPE organization_item = SELECT
    (document,
     product_definition,
     property_definition);
END_TYPE;
(*
```

5.2.2.12 person_item

A **person_item** identifies a **document** to which a person may be identified.

EXPRESS specification:

```
*)
TYPE person_item = SELECT
    (document);
END_TYPE;
(*
```

5.2.2.13 person_and_organization_item

A **person_and_organization_item** identifies an **action, action_request_solution**, **executed_action**, **document**, and **versioned_action_request** to which a **person_and_organization** may be identified.

EXPRESS specification:

```
*)
TYPE person_and_organization_item = SELECT
    (action_request_solution,
     document,
     executed_action,
     versioned_action_request,
     action);
END_TYPE;
(*
```


5.2.3 Ship moulded form entities

5.2.3.1 Ship moulded form entity definitions

5.2.3.1.1 applied_action_assignment

An **applied_action_assignment** specifies those **action_items** 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 5.2.4.26).

5.2.3.1.3 applied_approval_assignment

An **applied_approval_assignment** specifies those **approval_item**s to which an **approval_assignment** is assigned.

EXPRESS specification:

```
*)
ENTITY applied_approval_assignment
  SUBTYPE OF (approval_assignment);
  items : SET [1:?] OF approval_item;
WHERE
  wr1: NOT((SELF\approval_assignment.role.name = 'proposed alternative')) OR
         (SIZEOF(QUERY( app <*
         USEDIN (SELF\approval_assignment.assigned_approval,
        'SHIP_MOULDED_FORM_SCHEMA.APPROVAL_ASSIGNMENT.ASSIGNED_APPROVAL') |
       ('SHIP_MOULDED_FORM_SCHEMA.APPLIED_APPROVAL_ASSIGNMENT'
     IN TYPEOF (app)) AND
         (app\approval_assignment.role.name='subject')))=1);
END_ENTITY;
(*
```

Attribute definition:

items: the set of **approval_item** for which a particular **approval_assignment** is applicable.

Formal proposition:

WR1: Every **applied_approval_assignment** with a **role** attribute defining an **object_role** with a **name** attribute of value 'proposed alternative' shall reference an **approval** through the **assigned_approval** attribute. That **approval** entity shall also be referenced by another **applied_approval_assignment** through the **assigned_approval** attribute with a **role** attribute defining an **object_role** with a **name** of 'subject'.

Associated global rules:

The following global rule defined in this part of ISO 10303 applies to the **applied_approval_assignment** entity:

— approval_history_approves_same_definition (seee 5.2.4.9).

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 rule defined in this part of ISO 10303 applies 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);

— centre_location_compound_representation_has_specified_name (see 5.2.4.25);

— class_and_statutory_designation_has_properties (see 5.2.4.28);

— class_parameters_has_properties (see 5.2.4.30);

— compound_representation_item_with_class_id_knot (see 5.2.4.32);

— compound_representation_item_with_section_identifier (see 5.2.4.33);

— compound_representation_item_with_hydrostatic_properties (see 5.2.4.31);

— floating_position_compound_representation_with_name (see 5.2.4.43);

— global_axis_placement_has_properties (see 5.2.4.44);

— global_id_is_unique (see 5.2.4.45);

— hull_moulded_form_design_parameter_with_class_references (see 5.2.4.46);

— hydrostatic_properties_with_specified_class (see 5.2.4.47);

— hydrostatic_property_with_specified_name (see 5.2.4.48);

— offset_point_table_model_compound_representation_has_name (see 5.2.4.54);

— principal_characteristics_has_properties (see 5.2.4.55);

— product_definition_for_regulation (see 5.2.4.64);

— product_definition_for_class_notation (see 5.2.4.57);

— product_definition_for_owning_company (see 5.2.4.62);

— product_definition_relationship_with_identification_assignment (see 5.2.4.69);

— product_definition_shape_with_identification_assignment (see 5.2.4.70);

— product_definition_for_call_sign (see 5.2.4.56);

— product_definition_for_flag_state (see 5.2.4.58);

— product_definition_for_port_of_registration (see 5.2.4.63);

— product_definition_for_shipyard (see 5.2.4.65);

— product_definition_for_managing_company (see 5.2.4.60);

— product_definition_for_ordering_company (see 5.2.4.61);

— product_definition_for_stability_definition (see 5.2.4.66);

— product_definition_for_hydrostatic_definition_requires_reference (see 5.2.4.59);

— product_definition_with_identification_assignment (see 5.2.4.71);

— product_definition_relationship_related_to_class_moulded_form (see 5.2.4.68);

— product_related_product_category_with_identification_assignment (see 5.2.4.72);

— product_with_identification_assignment (see 5.2.4.73);

— propeller_moulded_form_design_parameter_with_class_references (see 5.2.4.74);

— property_definition_for_class_notation (see 5.2.4.78);

— property_definition_for_rudder_moulded_form_design_parameter (see 5.2.4.86);

— property_definition_for_moulded_form_function_parameters (see 5.2.4.84);

— property_definition_for_local_coordinate_system_with_position (see 5.2.4.83);

— property_definition_for_class_society (see 5.2.4.79);

— property_definition_for_thruster_moulded_form_design_parameter (see 5.2.4.87);

— property_definition_with_identification_assignment (see 5.2.4.89);

— property_definition_for_deck_moulded_form_design_parameter (see 5.2.4.80);

— property_definition_for_thruster_propeller_parameter (see 5.2.4.88);

— property_definition_for_appendage_moulded_form_design_parameter (see 5.2.4.75);

— property_definition_for_bottom_moulded_form_design_parameter (see 5.2.4.76);

— property_definition_for_bulb_moulded_form_design_parameter (see 5.2.4.77);

— property_definition_for_local_coordinate_system (see 5.2.4.82);

— property_definition_of_propeller_moulded_form_design_parameter (see 5.2.4.85);

— property_definition_for_hull_moulded_form_design_parameter (see 5.2.4.81);

— representation_for_stability_table_restricted_by_class_id (see 5.2.4.111);

— representation_for_offset_point_table_model_for_section (see 5.2.4.105);

— representation_for_hydrostatic_table_restricted (see 5.2.4.98);

— representation_for_midship_tumble_restricted_by_class_id (see 5.2.4.102);

— representation_for_hydrostatic_table_restricted_by_class_id (see 5.2.4.99);

— representation_for_propeller_location_restricted_by_class_id (see 5.2.4.107);

— representation_for_offset_table_shape_representation_restricted (see 5.2.4.106);

— representation_for_hydrostatic_table_constrained (see 5.2.4.97);

— representation_for_stability_table_restricted (see 5.2.4.110);

— representation_item_for_transformation_to_parent (see 5.2.4.125);

— ship_curve_has_name (see 5.2.4.135);

— ship_curve_segment_has_class (see 5.2.4.136);

— ship_curve_with_spacing_position_has_class (see 5.2.4.137);

— ship_designation_has_one_specified_names (see 5.2.4.138);

— ship_overall_dimensions_has_properties (see 5.2.4.140)

— ship_point_compound_representation_has_name (see 5.2.4.141);

— ship_surface_compound_representation_has_name (see 5.2.4.142);

— spacing_position_compound_representation_has_name (see 5.2.4.143);

— spacing_position_with_offset_compound_representation_has_class (see 5.2.4.144);

— spacing_position_with_offset_compound_representation_has_name (see 5.2.4.145);

— stability_properties_for_floating_position_has_name (see 5.2.4.147);

— stability_properties_for_floating_position_has_class (see 5.2.4.146);

— stability_property_has_name (see 5.2.4.148);

— valid_product_definition_for_class_moulded_form (see 5.2.4.152);

— versioned_action_request_with_identification_assignment (see 5.2.4.164).

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_time_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 rule defined in this part of ISO 10303 applies to the **applied_date_and_time_-assignment** entity:

— caused_when_for_check (see 5.2.4.20);

— caused_when_for_version_creation (see 5.2.4.22);

— caused_when_for_version_deletion (see 5.2.4.24);

— caused_when_for_version_modification (see 5.2.4.23);

— caused_when_for_envisaged_version_creation (see 5.2.4.21);

— date_time_for_change_plan (see 5.2.4.34);

— date_time_for_change_realisation (see 5.2.4.36);

— date_time_for_change_request (see 5.2.4.35).

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_items** 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;
(*
```

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 5.2.4.39).

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 rule defined in this part of ISO 10303 applies to the **applied_group_assignment** entity:

— approval_history_approves_same_definition (see 5.2.4.9);

— approval_history_has_at_least_one_member (see 5.2.4.10);

— approvals_references_approval_history (see 5.2.4.11);

— change_impact_with_versionable_object_change_event (see 5.2.4.26);

— members_is_referenced_by_at_least_one_revision (see 5.2.4.52);

— revision_with_context_referenced_for_context_of_revision (see 5.2.4.134);

— version_history_referenced_by_multiple_roles (see 5.2.4.158);

— version_history_is_referenced_by_at_least_one_versions (see 5.2.4.156);

— unique_approvals_in_approval_history (see 5.2.4.149);

— version_history_referenced_by_exactly_one_current_version (see 5.2.4.157);

— version_history_has_exactly_one_assigned_group (see 5.2.4.155);

— versions_is_referenced_by_at_least_one_version_history (see 5.2.4.165).

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 rule defined in this part of ISO 10303 applies to the **applied_identification_-assignment** entity:

— compound_representation_item_with_section_identifier (see 5.2.4.33);

— global_id_is_unique (see 5.2.4.45);

— product_definition_for_port_of_registration (see 5.2.4.63);

— product_definition_for_call_sign (see 5.2.4.56);

— product_definition_for_flag_state (see 5.2.4.58);

— representation_for_offset_point_table_model_for_point (see 5.2.4.104);

— ship_designation_has_one_specified_names (see 5.2.4.138);

— version_relationship_associates_with_versionable_object (see 5.2.4.160);

— versionable_object_has_one_version_id (see 5.2.4.163).

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 rule defined in this part of ISO 10303 applies to the **applied_organization_-assignment** entity:

— property_definition_for_class_society (see 5.2.4.79);

— product_definition_for_managing_company (see 5.2.4.60);

— product_definition_for_ordering_company (see 5.2.4.61);

— product_definition_for_owning_company (see 5.2.4.62);

— product_definition_for_shipyard (see 5.2.4.65).

5.2.3.1.12 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_organization_-assignment entity:

— document_has_exactly_one_author (see 5.2.4.38).

5.2.3.1.13 applied_person_and_organization_assignment

An **applied_person_and_organization_assignment** specifies those **person_and_organization_item**s to which a **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.

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_organization_-assignment entity:

— author_for_change_plan (see 5.2.4.12);

— author_for_change_realisation (see 5.2.4.13);

— author_for_change_request (see 5.2.4.14);

— caused_by_for_check (see 5.2.4.15);

— caused_by_for_envisaged_version_creation (see 5.2.4.16);

— caused_by_for_version_creation (see 5.2.4.17);

— caused_by_for_version_deletion (see 5.2.4.18);

— caused_by_for_version_modification (see 5.2.4.19);

— initiator_for_change_request (see 5.2.4.50).

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_MOULDED_FORM_SCHEMA.GROUP_ASSIGNMENT.ASSIGNED_GROUP')|
                   NOT ('SHIP_MOULDED_FORM_SCHEMA.APPLIED_GROUP_ASSIGNMENT'
                 IN TYPEOF(oa))
             )) =0);
END_ENTITY;
(*
```

Formal proposition:

WR1: Each **class** shall not be referenced by any **applied_group_assignment** through the **assigned_group** attribute.

5.2.3.2 Ship moulded form 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 applies to the **action** entity:

— action_request_solution_connected_to_action (see 5.2.4.1);

— caused_by_for_check (see 5.2.4.15);

— caused_by_for_envisaged_version_creation (see 5.2.4.16);

— caused_by_for_version_creation (see 5.2.4.17);

— caused_by_for_version_modification (see 5.2.4.19);

— caused_by_for_version_deletion (see 5.2.4.18);

— caused_when_for_check (see 5.2.4.20);

— caused_when_for_envisaged_version_creation (see 5.2.4.21);

— caused_when_for_version_creation (see 5.2.4.22);

— caused_when_for_version_deletion (see 5.2.4.24);

— caused_when_for_version_modification (see 5.2.4.23);

— envisaged_version_creation_has_mandatory_attribute_description (see 5.2.4.40);

— version_creation_has_mandatory_attribute_description (see 5.2.4.153);

— version_deletion_has_mandatory_attribute_description (see 5.2.4.154);

— version_modification_has_mandatory_attribute_description (see 5.2.4.159);.

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 applies to the **action_request_solution** entity:

— action_request_solution_connected_to_action (see 5.2.4.1);

— change_plan_has_mandatory_attribute_description (see 5.2.4.27);

— date_time_for_change_plan (see 5.2.4.34).

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 applies to the **action_request_solution** entity:

— approval_event_with_approval_date_time (see 5.2.4.8);

— approval_event_with_approval_person_organization (see 5.2.4.7);

— no_approvals_except_in_approval_history (see 5.2.4.53).

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 5.2.4.8).

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 5.2.4.7).

5.2.3.2.6 compound_representation_item

The base definition of the **compound_representation_item** 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 applies to the **compound_representation_-item** entity:

— centre_location_compound_representation_has_specified_name (see 5.2.4.25);

— compound_representation_item_with_section_identifier (see 5.2.4.33);

— compound_representation_item_with_class_id_knot (see 5.2.4.32);

— compound_representation_item_with_hydrostatic_properties (see 5.2.4.31);

— floating_position_compound_representation_with_name (see 5.2.4.43);

— hydrostatic_properties_with_specified_class (see 5.2.4.47);

— hydrostatic_property_with_specified_name (see 5.2.4.48);

— offset_point_table_model_compound_representation_has_name (see 5.2.4.54);

— ship_curve_has_name (see 5.2.4.135);

— ship_curve_segment_has_class (see 5.2.4.136);

— ship_curve_with_spacing_position_has_class (see 5.2.4.137);

— ship_point_compound_representation_has_name (see 5.2.4.141);

— ship_surface_compound_representation_has_name (see 5.2.4.142);

— spacing_position_compound_representation_has_name (see 5.2.4.143);

— spacing_position_with_offset_compound_representation_has_name (see 5.2.4.145);

— spacing_position_with_offset_compound_representation_has_class (see 5.2.4.144);

— stability_properties_for_floating_position_has_name (see 5.2.4.147);

— stability_properties_for_floating_position_has_class (see 5.2.4.146);

— stability_property_has_name (see 5.2.4.148).

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 applies to the **date_time_role** entity:

— caused_when_for_check (see 5.2.4.20);

— caused_when_for_envisaged_version_creation (see 5.2.4.21);

— caused_when_for_version_creation (see 5.2.4.22);

— caused_when_for_version_deletion (see 5.2.4.24);

— caused_when_for_version_modification (see 5.2.4.23);

— date_time_for_change_plan (see 5.2.4.34);

— date_time_for_change_realisation (see 5.2.4.36);

— 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 applies to the **document** entity:

— document_has_exactly_one_author (see 5.2.4.38);

— document_reference_with_address_has_at_least_one_references (see 5.2.4.39).

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 rules defined in this part of ISO 10303 applies to the **document_representation_type** entity:

— document_has_at_least_one_references (see 5.2.4.37);

— document_has_exactly_one_author (see 5.2.4.38).

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 applies to the **executed_action** entity:

— author_for_change_realisation (see 5.2.4.13);

— date_time_for_change_realisation (see 5.2.4.36).

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 5.2.4.51).

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 5.2.4.51).

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 applies to the **group** entity:

— approval_history_has_at_least_one_member (see 5.2.4.10);

— approvals_references_approval_history (see 5.2.4.11);

— change_impact_with_versionable_object_change_event (see 5.2.4.26);

— class_and_statutory_designation_has_properties (see 5.2.4.28);

— class_parameters_has_properties (see 5.2.4.30);

— global_axis_placement_has_properties (see 5.2.4.44);

— members_is_referenced_by_at_least_one_revision (see 5.2.4.52);

— principal_characteristics_has_properties (see 5.2.4.55):

— product_definition_for_regulation (see 5.2.4.64);

— product_definition_for_shipyard (see 5.2.4.65);

— product_definition_for_owning_company (see 5.2.4.62);

— product_definition_for_ordering_company (see 5.2.4.61);

— product_definition_for_managing_company (see 5.2.4.60);

— product_definition_for_class_notation (see 5.2.4.57);

— product_definition_for_port_of_registration (see 5.2.4.63);

— product_definition_for_flag_state (see 5.2.4.58);

— product_definition_for_call_sign (see 5.2.4.56);

— product_related_product_category_with_identification_assignment (see 5.2.4.72);

— property_definition_for_local_coordinate_system_with_position (see 5.2.4.83);

— property_definition_for_class_notation (see 5.2.4.78);

— property_definition_for_class_society (see 5.2.4.79);

— representation_item_for_transformation_to_parent (see 5.2.4.125);

— revision_has_mandatory_attribute_description (see 5.2.4.133);

— revision_with_context_referenced_for_context_of_revision (see 5.2.4.134);

— ship_designation_has_one_specified_names (see 5.2.4.138);

— spacing_position_compound_representation_has_name (see 5.2.4.143);

— spacing_position_with_offset_compound_representation_has_class (see 5.2.4.144);

— spacing_position_with_offset_compound_representation_has_name (see 5.2.4.145);

— unique_approvals_in_approval_history (see 5.2.4.149);

— version_history_has_exactly_one_assigned_group (see 5.2.4.155);

— version_history_is_referenced_by_at_least_one_versions (see 5.2.4.156);

— version_history_referenced_by_exactly_one_current_version (see 5.2.4.157);

— version_history_referenced_by_multiple_roles (see 5.2.4.158);

— versions_is_referenced_by_at_least_one_version_history (see 5.2.4.165).

5.2.3.2.14 group_assignment

The base definition of the **group_assignment** 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 applies to the **group_assignment** entity:

— valid_product_definition_for_class_moulded_form (see 5.2.4.152).

5.2.3.2.15 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 applies 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 5.2.4.161);

— version_relationship_has_unique_versions (see 5.2.4.162).

5.2.3.2.16 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 applies to the **identification_role** entity:

— global_id_is_unique (see 5.2.4.45);

— identification_role_optional_attribute_description_required (see 5.2.4.49);

— representation_has_global_unit_assigned_context (see 5.2.4.114).

5.2.3.2.17 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 5.2.4.125).

5.2.3.2.18 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 rule defined in this part of ISO 10303 applies to the **object_role** entity:

— approvals_references_approval_history (see 5.2.4.11).

5.2.3.2.19 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 applies to the **person_and_organization_role** entity:

— author_for_change_plan (see 5.2.4.12);

— author_for_change_realisation (see 5.2.4.13);

— author_for_change_request (see 5.2.4.14);

— caused_by_for_check (see 5.2.4.15);

— caused_by_for_envisaged_version_creation (see 5.2.4.16);

— caused_by_for_version_creation (see 5.2.4.17);

— caused_by_for_version_deletion (see 5.2.4.18);

— caused_by_for_version_modification (see 5.2.4.19);

— initiator_for_change_request (see 5.2.4.50).

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 applies to the **product_definition** entity:

— class_and_statutory_designation_has_properties (see 5.2.4.28);

— class_parameters_has_properties (see 5.2.4.30);

— product_definition_for_call_sign (see 5.2.4.56);

— product_definition_for_flag_state (see 5.2.4.58);

— product_definition_for_managing_company (see 5.2.4.60);

— product_definition_for_ordering_company (see 5.2.4.61);

— product_definition_for_owning_company (see 5.2.4.62);

— product_definition_for_port_of_registration (see 5.2.4.63);

— product_definition_for_regulation (see 5.2.4.64);

— product_definition_for_shipyard (see 5.2.4.65);

— property_definition_for_class_notation (see 5.2.4.78);

— ship_designation_has_one_specified_names (see 5.2.4.138);

— valid_product_definition_for_class_moulded_form (see 5.2.4.152).

5.2.3.2.21 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 rule defined in this part of ISO 10303 applies to the **product_definition_shape** entity:

— product_definition_shape_with_identification_assignment (see 5.2.4.70).

5.2.3.2.22 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 applies to the **product_definition_relationship** entity:

— product_definition_relationship_references_are_distinct (see 5.2.4.67);

— product_definition_relationship_related_to_class_moulded_form (see 5.2.4.68);

— ship_moulded_form_revision_has_description (see 5.2.4.139).

5.2.3.2.23 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 applies to the **property_definition** entity:

— class_and_statutory_designation_has_properties (see 5.2.4.28);

— hull_moulded_form_design_parameter_with_class_references (see 5.2.4.46);

— product_definition_for_hydrostatic_definition_requires_reference (see 5.2.4.59);

— product_definition_for_regulation (see 5.2.4.64);

— product_definition_for_stability_definition (see 5.2.4.66);

— propeller_moulded_form_design_parameter_with_class_references (see 5.2.4.74);

— property_definition_for_appendage_moulded_form_design_parameter (see 5.2.4.75);

— property_definition_for_bottom_moulded_form_design_parameter (see 5.2.4.76);

— property_definition_for_bulb_moulded_form_design_parameter (see 5.2.4.77);

— property_definition_for_class_notation (see 5.2.4.78);

— property_definition_for_class_society (see 5.2.4.79);

— property_definition_for_hull_moulded_form_design_parameter (see 5.2.4.81);

— property_definition_for_local_coordinate_system (see 5.2.4.82);

— property_definition_for_local_coordinate_system_with_position (see 5.2.4.83);

— property_definition_for_moulded_form_function_parameters (see 5.2.4.84);

— property_definition_of_propeller_moulded_form_design_parameter (see 5.2.4.85);

— property_definition_for_rudder_moulded_form_design_parameter (see 5.2.4.86);

— property_definition_for_thruster_moulded_form_design_parameter (see 5.2.4.87);

— property_definition_for_thruster_propeller_parameter (see 5.2.4.88);

— representation_item_for_transformation_to_parent (see 5.2.4.125);

— ship_overall_dimensions_has_properties (see 5.2.4.140).

5.2.3.2.24 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 applies to the **property_definition_-representation** entity:

— class_and_statutory_designation_has_properties (see 5.2.4.28);

— class_notation_with_named_representation_items (see 5.2.4.29);

— class_parameters_has_properties (see 5.2.4.30);

— global_axis_placement_has_properties (see 5.2.4.44);

— hull_moulded_form_design_parameter_with_class_references (see 5.2.4.46);

— principal_characteristics_has_properties (see 5.2.4.55):

— propeller_moulded_form_design_parameter_with_class_references (see 5.2.4.74);

— property_definition_for_appendage_moulded_form_design_parameter (see 5.2.4.75);

— property_definition_for_bottom_moulded_form_design_parameter (see 5.2.4.76);

— property_definition_for_bulb_moulded_form_design_parameter (see 5.2.4.77);

— property_definition_for_hull_moulded_form_design_parameter (see 5.2.4.81);

— property_definition_for_local_coordinate_system (see 5.2.4.82);

— property_definition_for_local_coordinate_system_with_position (see 5.2.4.83);

— property_definition_for_moulded_form_function_parameters (see 5.2.4.84);

— property_definition_of_propeller_moulded_form_design_parameter (see 5.2.4.85);

— property_definition_for_rudder_moulded_form_design_parameter (see 5.2.4.86);

— property_definition_for_thruster_moulded_form_design_parameter (see 5.2.4.87);

— property_definition_for_thruster_propeller_parameter (see 5.2.4.88);

— representation_for_class_and_statutory_designation (see 5.2.4.93);

— representation_for_global_axis_placement (see 5.2.4.95);

— representation_for_local_coordinate_system (see 5.2.4.100);

— representation_of_local_coordinate_system_with_position_reference (see 5.2.4.101);

— representation_items_optional_for_class_notation (see 5.2.4.126);

— representation_items_optional_for_principal_characteristics (see 5.2.4.128);

— representation_restricted_by_name_principal_characteristics (see 5.2.4.131);

— representation_restricted_by_name_class_notation (see 5.2.4.129);

— representation_restricted_by_name_class_parameters (see 5.2.4.130);

— ship_overall_dimensions_has_properties (see 5.2.4.140).

5.2.3.2.25 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 applies to the **representation** entity:

— class_notation_with_named_representation_items (see 5.2.4.29);

— hull_moulded_form_design_parameter_with_class_references (see 5.2.4.46);

— propeller_moulded_form_design_parameter_with_class_references (see 5.2.4.74);

— representation_for_moulded_form_function_parameters (see 5.2.4.103);

— representation_for_local_coordinate_system (see 5.2.4.100);

— representation_for_offset_point_table_model_for_section (see 5.2.4.105);

— representation_for_deck_moulded_form_design_parameter (see 5.2.4.94);

— representation_for_class_and_statutory_designation (see 5.2.4.93);

— representation_for_class_and_statutory_designation (see 5.2.4.93);

— representation_for_global_axis_placement (see 5.2.4.95);

— representation_for_rudder_moulded_form_design_parameter (see 5.2.4.109);

— representation_for_propeller_moulded_form_design_parameter (see 5.2.4.108);

— representation_for_propeller_location_restricted_by_class_id (see 5.2.4.107);

— representation_for_hydrostatic_table_restricted_by_class_id (see 5.2.4.99);

— representation_for_offset_point_table_model_for_point (see 5.2.4.104);

— representation_for_hydrostatic_table_restricted (see 5.2.4.98);

— representation_for_hydrostatic_table_constrained (see 5.2.4.97);

— representation_for_stability_table_restricted_by_class_id (see 5.2.4.111);

— representation_for_stability_table_restricted (see 5.2.4.110);

— representation_for_offset_table_shape_representation_restricted (see 5.2.4.106);

— representation_for_thruster_moulded_form_design_parameter (see 5.2.4.112);

— representation_for_bottom_moulded_form_design_parameter (see 5.2.4.91);

— representation_for_appendage_moulded_form_design_parameter (see 5.2.4.90);

— representation_for_hull_moulded_form_design_parameter (see 5.2.4.96);

— representation_for_midship_tumble_restricted_by_class_id (see 5.2.4.102);

— representation_for_bulb_moulded_form_design_parameter (see 5.2.4.92);

— representation_has_global_unit_assigned_context (see 5.2.4.114);

— representation_has_global_uncertainty_assigned_context (see 5.2.4.113);

— representation_items_for_deck_moulded_form_design_parameter (see 5.2.4.118);

— representation_items_for_rudder_moulded_form_design_parameter (see 5.2.4.123);

— representation_items_optional_for_owner_designation (see 5.2.4.127);

— representation_items_propeller_moulded_form_design_parameter (see 5.2.4.122);

— representation_items_for_hull_moulded_form_design_parameter (see 5.2.4.119);

— representation_items_for_bulb_moulded_form_design_parameter (see 5.2.4.117);

— representation_items_of_thruster_moulded_form_design_parameter (see 5.2.4.124);

— representation_item_for_transformation_to_parent (see 5.2.4.125);

— representation_items_optional_for_class_notation (see 5.2.4.126);

— representation_items_optional_for_principal_characteristics (see 5.2.4.128);

— representation_items_propeller_moulded_form_design_parameter (see 5.2.4.122);

— representation_items_for_bottom_moulded_form_design_parameter (see 5.2.4.116);

— representation_items_appendage_moulded_form_design_parameter (see 5.2.4.115);

— representation_items_for_moulded_form_function_parameters (see 5.2.4.121);

— representation_items_for_moulded_form_design_paramters(see 5.2.4.120);

— representation_of_local_coordinate_system_with_position_reference (see 5.2.4.101);

— representation_restricted_by_name_principal_characteristics (see 5.2.4.131);

— representation_restricted_by_name_class_notation (see 5.2.4.129);

— representation_restricted_by_name_class_parameters (see 5.2.4.130);

— representation_restricted_by_name_ship_overall_dimensions (see 5.2.4.132);

— user_def_appendage_type_description_required (see 5.2.4.150);

— user_def_function_description_required (see 5.2.4.151).

5.2.3.2.26 representation_item

The base definition of the **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 applies to the **representation_item** entity:

— class_notation_with_named_representation_items (see 5.2.4.29);

— representation_for_class_and_statutory_designation (see 5.2.4.93);

— representation_for_global_axis_placement (see 5.2.4.95);

— representation_for_local_coordinate_system (see 5.2.4.100);

— representation_of_local_coordinate_system_with_position_reference (see 5.2.4.101);

— representation_items_optional_for_class_notation (see 5.2.4.126);

— representation_items_optional_for_owner_designation (see 5.2.4.127);

— representation_items_optional_for_principal_characteristics (see 5.2.4.128);

— representation_restricted_by_name_class_notation (see 5.2.4.129);

— representation_restricted_by_name_class_parameters (see 5.2.4.130);

— representation_restricted_by_name_principal_characteristics (see 5.2.4.131).

5.2.3.2.27 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 applies to the **versioned_action_request** entity:

— author_for_change_request (see 5.2.4.14);

— date_time_for_change_request (see 5.2.4.35);

— initiator_for_change_request (see 5.2.4.50).

5.2.4 Ship moulded form 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;
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;

(*
```

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_MOULDED_FORM_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** has a **name** attribute of value '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_MOULDED_FORM_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** has a **name** attribute of value ‘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.name** equals '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 **applied_classification_assignment .assigned_class.name** equals '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** points to 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_MOULDED_FORM_SCHEMA.IDENTIFICATION_ASSIGNMENT_RELATIONSHIP.'
            +'RELATING_IDENTIFICATION_ASSIGNMENT'))) > 0) OR
     (SIZEOF(USEDIN(applied_identification_assignment[i],

('SHIP_MOULDED_FORM_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:=NOTVALUE_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.

identification_assignment_relationship: the set of all instances of **identification_assignment_relationship** entities.

Formal propositions:

WR1: Every instance of **identification_assignment_relationship** that has an **applied_classification_assignment** whose **assigned_class.name** equals 'alternative 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'.

5.2.4.7 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_MOULDED_FORM_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 **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_person_organization** through **authorized_approval**.

5.2.4.8 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_MOULDED_FORM_SCHEMA.APPROVAL_DATE_TIME.' +
      'DATED_APPROVAL'));
    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 'approval event' is referenced by exactly one instance of **approval_date_time** through **dated_approval**.

5.2.4.9 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_MOULDED_FORM_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 instances 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_classification** is a **group** with attribute **name** equals '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_classification** is a **group** with attribute **name** equals 'definition' in its **items** attribute.

5.2.4.10 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_-classification** is a **group** with attribute **name** equals 'approval history' shall be the **assigned_group** in exactly one instance of **applied_group_assignment**.

5.2.4.11 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_classification** attribute identifies an entity with attribute **name** equals 'approval history' shall only be referenced by instances of type **applied_group_assignment** whose role equals 'approvals' through attribute **assigned_group**.

5.2.4.12 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'.

5.2.4.13 author_for_change_realisation

The **author_for_change_realisation** 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_realisation
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'.

5.2.4.14 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 instance 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'.

5.2.4.15 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'.

5.2.4.16 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'.

5.2.4.17 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'.

5.2.4.18 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'.

5.2.4.19 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'.

5.2.4.20 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'.

5.2.4.21 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'.

5.2.4.22 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'.

5.2.4.23 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'.

5.2.4.24 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'.

5.2.4.25 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 attribute **assigned_classification.name** equals '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.

5.2.4.26 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_MOULDED_FORM_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 'versionable object change event' through its attribute **items**.

5.2.4.27 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 id '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 'change plan'.

5.2.4.28 class_and_statutory_designation_has_properties

The **class_and_statutory_designation_has_properties** rule specifies that a **product_definition** with a class id '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 attribute **assigned_classification** is a **group** with attribute **name** equal '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'.

5.2.4.29 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_items** 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_MOULDED_FORM_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 either '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.

5.2.4.30 class_parameters_has_properties

The **class_parameters_has_properties** rule specifies that a **product_definition** with a class id '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 attribute **assigned_class** is a **group** with attribute **name** equal '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'.

5.2.4.31 compound_representation_item_with_hydrostatic_properties

The **compound_representation_item_with_hydrostatic_properties** rule specifies the **item_element** attribute of a **compound_representation_item** with the class id 'hydrostatic properties for constant floating position' to have in the **list_representation_item** for each entry in the list one or more **representation_items** whose **name** attribute has the value given in the list.

EXPRESS specification:

```
*)
RULE
compound_representation_item_with_hydrostatic_properties
```



```
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF COMPOUND_REPRESENTATION_ITEM := [];
    t2_set:  LIST OF representation_item := [];
    arg_list:  LIST OF STRING := ['hydrostatic property value'];
    violation: LOGICAL := FALSE;
  END_LOCAL;


   (* get all classification_assignment instances with id 'hydrostatic
properties for constant floating position' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                       i.Assigned_class.name = 'hydrostatic properties for
constant floating position');

   (* get all instances of compound_representation_item that have class id
'hydrostatic properties for constant 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 one or more 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 'hydrostatic properties for constant floating position' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall for each value out of 'hydrostatic property value' collect at least one instance of **representation_item** whose name attribute equals that value.

5.2.4.32 compound_representation_item_with_class_id_knot

The **compound_representation_item_with_class_id_knot** rule specifies the **item_element** attribute of a **compound_representation_item** with the class id 'knot' to have in the **list_representation_item** two or more **compound_representation_item** with the class id 'ship curve'

EXPRESS specification:

```
*)
RULE compound_representation_item_with_class_id_knot

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 'knot' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'knot');

   (* get all instances of compound_representation_item that have class id
'knot' *)
  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 'ship curve' *)
  c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'ship curve');

   (* get all instances of compound_representation_item that have class id
'ship curve' *)
  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 the second list of
   compound_representation_item is greater then or equal to 2
   *)
 REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
   l_rep_item := t1_set[i].item_element;
      REPEAT j := 1 TO HIINDEX(t1_set[i].item_element);
          t3_set := t3_set + l_rep_item[j];
      END_REPEAT;
   violation := (SIZEOF(t3_set * t2_set) < 2);
   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 'knot' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall collect at least two instances of **compound_representation_item** that has an **applied_classification_assignment** whose attribute **assigned_class** equals 'ship curve'.

5.2.4.33 compound_representation_item_with_section_identifier

The **compound_representation_item_with_section_identifier** rule specifies that an instance of **compound_representation_item** with class id 'section of offset point table' is referenced by exactly one instance of **applied_identification_assignment** via **items** whose attribute **role** has the value 'section identifier'.

EXPRESS specification:

```
*)
RULE  compound_representation_item_with_section_identifier
FOR(APPLIED_CLASSIFICATION_ASSIGNMENT);
  LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT := [];
    t1_set:  SET OF compound_representation_item := [];
    t2_set:  SET OF applied_identification_assignment := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'section of offset
point table' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'section of offset point table');

   (* get all instances of compound_representation_item that have class id
'section of offset point 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;

  (* for all instances of compound_representation_item in t1_set:
```

```
    get the applied_identification_assignment instances that are referencing
a compound_representation_item instance via items,
       filter out those applied_identification_assignment instances whose
attribute role has the value 'section identifier'
     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_MOULDED_FORM_SCHEMA.' +
'APPLIED_IDENTIFICATION_ASSIGNMENT' + '.ITEMS'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.role.name =
'section identifier')) = 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 **compound_representation_item** that is referenced by a **applied_classification_-assignment** with **name** = 'section of offset point table', is referenced by exactly one **applied_-identification_assignment.items**, where **applied_identification_assignment.role** = 'section identifier'.

5.2.4.34 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 **role** 'change plan' 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** 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 date_time_for_change_realisation

The **date_time_for_change_realisation** 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_realisation
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 realisation'));

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'.

5.2.4.37 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_MOULDED_FORM_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 'document' shall be referenced by one or more instances of **document_representation_type** through attribute **represented_document**.

5.2.4.38 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** or **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_MOULDED_FORM_SCHEMA.' +
         'APPLIED_PERSON_ASSIGNMENT.ITEMS');
     bag_1 := QUERY( assign <* bag_1 | assign.role.name = ‘author’);
   bag_2 := USEDIN(document[i], 'SHIP_MOULDED_FORM_SCHEMA.' +
```

```
        'APPLIED_PERSON_AND_ORGANIZATION_ASSIGNMENT.ITEMS');
  bag_2 := QUERY( assign <* bag_2 | assign.role.name = 'author');
  bag_3 := USEDIN(document[i], 'SHIP_MOULDED_FORM_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_person_-assignment**, **applied_organization_assignment** ,or **applied_person_and_organization_assignment** whose role equals 'author'.

5.2.4.39 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_assignments** 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_MOULDED_FORM_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 'document reference with address' shall be referenced by one or more instances of **applied_external_identification_assignments** through attribute **items**.

5.2.4.40 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 'envisaged version creation'.

5.2.4.41 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_MOULDED_FORM_SCHEMA.APPLIED_IDENTIFICATION_ASSIGNMENT.ITEMS'));
      t2_set := QUERY ( j <* t2_set | j.role.name = 'globally unambiguous
identifier');
            iolation := 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** has a **name** attribute of value 'change realization' shall require an **applied_identification_assignment** to define the instance identifier.

5.2.4.42 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_MOULDED_FORM_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; -- external_instance_reference_has_same_identifier
(*
```

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.

5.2.4.43 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;

  (* get all classification_assignment instances with id 'floating position'
*)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.Assigned_class.name = 'floating position');

   (* get all instances of compound_representation_item that have class id
'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 'floating position' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall for each value out of 'moulded form displacement', 'draught at amidships', 'length of waterline', 'breadth of waterline', 'angle of trim', 'angle of heel' collect exactly one instance of **representation_item** whose **name** attribute equals that value.

5.2.4.44 global_axis_placement_has_properties

The **global_axis_placement_has_properties** rule specifies that a **product_definition** with a class id '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 attribute **assigned_class** is a **group** with attribute **name** equal '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** has an atttribute **role** that references an **identification_role** with **name** = 'globally unambiguous identifier', and 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 hull_moulded_form_design_parameter_with_class_references

The **hull_moulded_form_design_parameter_with_class_references** rule specifies that an instance of **property_definition** with class id 'hull moulded form design parameter' is referenced by at most one instance of **property_definition_representation** whose used_representation attribute points to a representation with class id ‘midship_tumble’.

EXPRESS specification:

```
*)

RULE hull_moulded_form_design_parameter_with_class_references
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 := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id
  'hull moulded form design parameter' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'hull moulded form design
parameter');

  (* get all instances of property_definition that have class id
  'hull moulded form design parameter' *)
  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, get those
property_definition_representation.used_representation instances whose
class id is 'midship_tumble' and make sure their number is less than or
equal to 1
   *)
 REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
   'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
   'DEFINITION'));

   REPEAT j := 1 to HIINDEX(t2_set);
```

```
    t3_set := t3_set + t2_set[j].used_representation;
   END_REPEAT;

   violation := SIZEOF(QUERY(t2_inst <* t3_set | 'midship tumble' 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** has a **name** attribute of value 'hull moulded form design parameter' shall be referenced by one or less instances of **property_definition_representation** that references a representation via its **used_representation** attribute that has an **applied_classification_assignment** whose **assigned_class** has a **name** attribute of value 'midship tumble'.

5.2.4.47 hydrostatic_properties_with_specified_class

The **hydrostatic_properties_with_specified_class** rule specifies the **item_element** attribute of a **compound_representation_item** with the class id 'hydrostatic properties for constant floating position' to have in the **list_representation_item** exactly one **compound_representation_item** with the class id 'floating position'.

EXPRESS specification:

```
*)
RULE
hydrostatic_properties_with_specified_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 'hydrostatic
properties for constant floating position' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.Assigned_class.name = 'hydrostatic properties for
constant floating position');
```



```
  (* get all instances of compound_representation_item that have class id
'hydrostatic properties for constant 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 'floating position'
*)
  c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.Assigned_class.name = 'floating position');

  (* get all instances of compound_representation_item that have class id
'floating 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;

  (* 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 equal 1
   *)
  REPEAT i:=1 TO HIINDEX(t1_set) WHILE (NOT violation);
     l_rep_item := t1_set[i].item_element;
     REPEAT j := 1 TO HIINDEX(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 'hydrostatic properties for constant floating position' 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 'floating position'.

5.2.4.48 hydrostatic_property_with_specified_name

The **hydrostatic_property_with_specified_name** rule specifies the **item_element** attribute of a **compound_representation_item** with the class id 'hydrostatic 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 hydrostatic_property_with_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 := ['property type'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'hydrostatic
property' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.Assigned_class.name = 'hydrostatic property');

  (* get all instances of compound_representation_item that have class id
'hydrostatic 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 'hydrostatic property' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall for each value out of 'property type' collect exactly one instance of **representation_item** whose **name** attribute equals that value.

5.2.4.49 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 is name equals 'external reference'.

5.2.4.50 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'.

5.2.4.51 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'.

5.2.4.52 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**.

5.2.4.53 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_MOULDED_FORM_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'.

5.2.4.54 offset_point_table_model_compound_representation_has_name

The **offset_point_table_model_compound_representation_has_name** rule specifies the **item_-element** attribute of a **compound_representation_item** with the class id 'offset point table model' 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 offset_point_table_model_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 point table type'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'offset point
table model' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'offset point table model');

  (* get all instances of compound_representation_item that have class id
'offset point table model' *)
  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 'offset point table model' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall for each value out of 'offset point table type' collect exactly one instance of **representation_item** whose **name** attribute equals that value.

5.2.4.55 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 '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'.

5.2.4.56 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_MOULDED_FORM_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'.

5.2.4.57 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_MOULDED_FORM_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**.

5.2.4.58 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_MOULDED_FORM_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'.

5.2.4.59 product_definition_for_hydrostatic_definition_requires_reference

The **product_definition_for_hydrostatic_definition_requires_reference** rule specifies that a **property_definition** with a class id 'hydrostatic definition' is referenced by one or more **representation**s with a class id 'hydrostatic table' via a **property_definition_representation**

EXPRESS specification:

```
*)
RULE product_definition_for_hydrostatic_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 'hydrostatic
definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'hydrostatic definition');

  (* get all instances of property_definition that have class id
'hydrostatic 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 'hydrostatic table'
*)
  c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'hydrostatic table');

  (* get all instances of representation that have class id 'hydrostatic
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
'hydrostatic table' *)
  REPEAT i := 1 TO HIINDEX(t2_set);
    t3_set := t3_set + bag_to_set(USEDIN(t2_set[i],
```



```
            'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_-
REPRESENTATION.USED_REPRESENTATION'));
  END_REPEAT;

(* get all property_definition instances which are the .definition of the
property_definition_representation *)
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;

(* compare both lists with property_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 attribute **assigned_class** equals 'hydrostatic 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 attribute **assigned_-class.name** equals 'hydrostatic table'.

5.2.4.60 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 instances 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_MOULDED_FORM_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'.

5.2.4.61 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_MOULDED_FORM_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'.

5.2.4.62 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_MOULDED_FORM_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'.

5.2.4.63 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_MOULDED_FORM_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'.

5.2.4.64 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_MOULDED_FORM_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**.

5.2.4.65 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_MOULDED_FORM_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'.

5.2.4.66 product_definition_for_stability_definition

The **product_definition_for_stability_definition** rule specifies that a **property_definition** with a class id 'stability definition' is referenced by one or more **representations** with a class id 'stability table' via a **property_definition_representation**.

EXPRESS specification:

```
*)
RULE product_definition_for_stability_definition
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_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_-
```

```
REPRESENTATION.USED_REPRESENTATION'));
  END_REPEAT;

(* get all property_definition instances which are the .definition of the
property_definition_representaiton *)
  REPEAT i := 1 TO HIINDEX(t3_set);
    t4_set := t4_set + t3_set[i].definition;
  END_REPEAT;

(* compare both lists with property_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 attribute **assigned_class** equals '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 attribute **assigned_class.name** equals 'stability table'.

5.2.4.67 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.

5.2.4.68 product_definition_relationship_related_to_class_moulded_form

The **product_definition_relationship_related_to_class_moulded_form** rule specifies the **related** or **relating** attribute of a **product_definition_relationship** of class 'moulded form relationship' shall be an entity type **product_definition** with a class 'moulded form'.

EXPRESS specification:

```
*)
RULE
 product_definition_relationship_related_to_class_moulded_form

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 product_definition_RELATIONSHIP := [];
    t2_set:  SET OF product_definition := [];
    t3_set:  SET OF product_definition := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'moulded form
relationship' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'moulded form relationship');

  (* get all instances of product_definition_relationship that have class
id 'moulded form relationship' *)
  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 related_product_definition attribute instances and
relating_product_definition attribute instances of the
product_definition_relationship *)
  REPEAT i := 1 TO HIINDEX(t1_set);
```

```
    t2_set := t2_set + t1_set[i].related_product_definition;
    t2_set := t2_set + t1_set[i].relating_product_definition;
   END_REPEAT;

(* get all classification_assignment instances with id 'moulded form' *)
  c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'moulded form');

  (* get all instances of product_definition that have class id 'moulded
form' *)
  REPEAT i := 1 TO HIINDEX(c2_a_set);
    REPEAT j := 1 TO HIINDEX(c2_a_set[i].items);
      t3_set := t3_set + c2_a_set[i].items[j];
    END_REPEAT;
  END_REPEAT;
violation := NOT(t2_set<= t3_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: The **related_product_definition** and the **relating_product_definition** of every instance of **product_definition_relationship** that has an **applied_classification_assignment** whose **assigned_class.name** equals 'moulded form relationship' shall be instances of **product_definition** having an **applied_classification_assignment** whose **assigned_class.name** equals 'moulded form'.

5.2.4.69 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 := ['item 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_MOULDED_FORM_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 'item relationship' shall require an **applied_identification_assignment** to define the instance identifier.

5.2.4.70 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 := ['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_MOULDED_FORM_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 'definition' shall require an **applied_identification_assignment** to define the instance identifier.

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_MOULDED_FORM_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** attribute.

```
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_MOULDED_FORM_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** attribute.

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_MOULDED_FORM_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 propeller_moulded_form_design_parameter_with_class_references

The **propeller_moulded_form_design_parameter_with_class_references** rule specifies that an instance of **property_definition** with class id 'propeller moulded form design parameter' is referenced by at most one instances of **property_definition_representation** whose **used_representation** attribute points to a **representation** with class id ‘propeller location’.

EXPRESS specification:

```
*)
RULE propeller_moulded_form_design_parameter_with_class_references
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 := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'propeller moulded form design
parameter');

  (* get all instances of property_definition that have class id
  'propeller moulded form design parameter' *)
  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, get those
property_definition_representation.used_representation instances whose
class id is 'midship_tumble' and make sure their number is less than or
equal to 1
   *)
  REPEAT i := 1 TO HIINDEX(t1_set) WHILE NOT violation;
    t2_set := bag_to_set(USEDIN(t1_set[i],
   'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
   'DEFINITION'));

   REPEAT j := 1 to HIINDEX(t2_set);
        t3_set := t3_set + t2_set[j].used_representation;
   END_REPEAT;

    violation := SIZEOF(QUERY(t2_inst <* t3_set | 'propeller location' 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** has a **name** attribute of value 'propeller moulded form design parameter' shall be referenced by one or less instances of **property_definition_representation** that references a representation via its **used_representation** attribute that has an **applied_classification_assignment** whose **assigned_class** has a **name** attribute of value 'propeller location'.

5.2.4.75 property_definition_appendage_moulded_-form_design_parameter

The **property_definition_for_appendage_moulded_form_design_parameter** rule specifies that an instance of **property_definition** with class id 'appendage moulded form design parameter' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'appendage moulded form design parameter'.

EXPRESS specification:

```
*)
RULE  property_definition_appendage_moulded_form_design_parameter
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 'appendage moulded
form design parameter' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'appendage moulded form design
parameter');

  (* get all instances that have class id 'appendage moulded form design
parameter' *)
  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 'appendage moulded form design parameter'
     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_MOULDED_FORM_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name =
'appendage moulded form design parameter')) = 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 a **applied_classification_assignment** with **assigned_class.name** = 'appendage moulded form design parameter', is referenced by exactly one **property_definition_representation.definition**, where **property_definition_representation.name** = 'appendage moulded form design parameter'.

5.2.4.76 property_definition_for_bottom_moulded_form_design_parameter

The **property_definition_for_bottom_moulded_form_design_parameter** rule specifies that an instance of **property_definition** with class id 'bottom moulded form design parameter' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'bottom moulded form design parameter'.

EXPRESS specification:

```
*)
RULE  property_definition_for_bottom_moulded_form_design_parameter
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 'bottom moulded
form design parameter' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'bottom moulded form design
parameter');

  (* get all instances that have class id 'bottom moulded form design
parameter' *)
  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 'bottom moulded form design parameter'
     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_MOULDED_FORM_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name =
'bottom moulded form design parameter')) = 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** = 'bottom moulded form design parameter', is referenced by exactly one **property_definition_representation.definition**, where **property_definition_representation.name** = 'bottom moulded form design parameter'.

5.2.4.77 property_definition_for_bulb_moulded_form_design_parameter

The **property_definition_for_bulb_moulded_form_design_parameter** rule specifies that an instance of **property_definition** with class id 'bulb moulded form design parameter' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'bulb moulded form design parameter'

EXPRESS specification:

```
*)
RULE  property_definition_for_bulb_moulded_form_design_parameter
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 'bulb moulded form
design parameter' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'bulb moulded form design
parameter');

  (* get all instances that have class id 'bulb moulded form design
parameter' *)
  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 'bulb moulded form design parameter'
     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_MOULDED_FORM_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name = 'bulb
moulded form design parameter')) = 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** = 'bulb moulded form design parameter', is referenced by exactly one **property_definition_representation.definition**, where **property_definition_representation.name** = 'bulb moulded form design parameter'.

5.2.4.78 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;

  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_MOULDED_FORM_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 **assigned_class.name** = 'class notation', is referenced by exactly one **property_definition_-representation** attribute **definition**, where **property_definition_representation** attribute **name** = 'class notation'.

5.2.4.79 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.name** 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;

  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_MOULDED_FORM_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 **assigned_class.name** = 'class notation', is referenced by exactly one **applied_organization_-assignment** attribute **items**, where **applied_organization_assignment** attribute **role.name** = 'class society'.

5.2.4.80 property_definition_for_deck_moulded_form_design_parameter

The **property_definition_for_deck_moulded_form_design_parameter** rule specifies that an instance of **property_definition** with class id 'deck moulded form design parameter' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'deck moulded form design parameter'.

EXPRESS specification:

```
*)
RULE  property_definition_for_deck_moulded_form_design_parameter
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 moulded form
design parameter' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'deck moulded form design
parameter');

  (* get all instances that have class id 'deck moulded form design
parameter' *)
  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 moulded form design parameter'
     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_MOULDED_FORM_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name = 'deck
moulded form design parameter')) = 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** = 'deck moulded form design parameter', is referenced by exactly one **property_definition_representation.definition**, where **property_definition_representation.name** = 'deck moulded form design parameter'.

5.2.4.81 property_definition_for_hull_moulded_form_design_parameter

The **property_definition_for_hull_moulded_form_design_parameter** rule specifies that an instance of **property_definition** with class id 'hull moulded form design parameter' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'hull moulded form design parameter'.

EXPRESS specification:

```
*)
RULE  property_definition_for_hull_moulded_form_design_parameter
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 'hull moulded form
design parameter' *)
```

```
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                    i.ASSIGNED_CLASS.NAME = 'hull moulded form design
parameter');

  (* get all instances that have class id 'hull moulded form design
parameter' *)
  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 'hull moulded form design parameter'
     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_MOULDED_FORM_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name = 'hull
moulded form design parameter')) = 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** = 'hull moulded form design parameter', is referenced by exactly one **property_definition_representation.definition**, where **property_definition_representation.name** = 'hull moulded form design parameter'.

5.2.4.82 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_MOULDED_FORM_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** = 'local coordinate system'.

5.2.4.83 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_MOULDED_FORM_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** = 'local coordinate system with position reference'.

5.2.4.84 property_definition_for_moulded_form_function_parameters

The **property_definition_for_moulded_form_function_parameters** rule specifies that an instance of **property_definition** with class id 'moulded form functional definition' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'moulded form function parameters'.

EXPRESS specification:

```
*)
RULE
 property_definition_for_moulded_form_function_parameters
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 'moulded form
functional definition' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'moulded form functional
definition');

  (* get all instances that have class id 'moulded form 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 'moulded form 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_MOULDED_FORM_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name =
'moulded form 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** = 'moulded form functional definition', is referenced by exactly one **property_definition_representation.definition**, where **property_definition_representation.name** = 'moulded form function parameters'.

5.2.4.85 property_definition_of_propeller_moulded_form_design_parameter

The **property_definition_of_propeller_moulded_form_design_parameter** rule specifies that an instance of **property_definition** with class id 'propeller moulded form design parameter' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'propeller moulded form design parameter'.

EXPRESS specification:

```
*)
RULE  property_definition_of_propeller_moulded_form_design_parameter
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 'propeller moulded
form design parameter' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'propeller moulded form design
parameter');

  (* get all instances that have class id 'propeller moulded form design
parameter' *)
  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 'propeller moulded form design parameter'
     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_MOULDED_FORM_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name =
'propeller moulded form design parameter')) = 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** = 'propeller moulded form design parameter', is referenced by exactly one **property_definition_representation.definition**, where **property_definition_representation.name** = 'propeller moulded form design parameter'.

5.2.4.86 property_definition_for_rudder_moulded_form_design_parameter

The **property_definition_for_rudder_moulded_form_design_parameter** rule specifies that an instance of **property_definition** with class id 'rudder moulded form design parameter' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'rudder moulded form design parameter'.

EXPRESS specification:

```
*)
RULE  property_definition_for_rudder_moulded_form_design_parameter
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 'rudder moulded
form design parameter' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'rudder moulded form design
parameter');

  (* get all instances that have class id 'rudder moulded form design
parameter' *)
  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 'rudder moulded form design parameter'
     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_MOULDED_FORM_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name =
'rudder moulded form design parameter')) = 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** = 'rudder moulded form design parameter', is referenced by exactly one **property_definition_representation.definition**, where **property_definition_representation.name** = 'rudder moulded form design parameter'.

5.2.4.87 property_definition_for_thruster_moulded_form_design_parameter

The **property_definition_for_thruster_moulded_form_design_parameter** rule specifies that an instance of **property_definition** with class id 'thruster moulded form design parameter' is referenced by exactly one instance of **property_definition_representation** via **definition** whose attribute **name** has the value 'thruster moulded form design parameter'.

EXPRESS specification:

```
*)
RULE  property_definition_for_thruster_moulded_form_design_parameter
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 'thruster moulded
form design parameter' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'thruster moulded form design
parameter');

  (* get all instances that have class id 'thruster moulded form design
parameter' *)
```



```
  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 'thruster moulded form design parameter'
     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_MOULDED_FORM_SCHEMA.' +
'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name =
'thruster moulded form design parameter')) = 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** = 'thruster moulded form design parameter', is referenced by exactly one **property_definition_representation.definition**, where **property_definition_representation.name** = 'thruster moulded form design parameter'.

5.2.4.88 property_definition_for_thruster_propeller_parameter

The **property_definition_for_thruster_propeller_parameter** rule specifies that an instance of **property_definition** with class id 'thruster moulded form design parameter' is referenced by exactly one instance of **property_definition_relationship** via **relating_property_definition** whose attribute **name** has the value 'thruster propeller parameter'.

EXPRESS specification:

```
*)
RULE  property_definition_for_thruster_propeller_parameter
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_relationship := [];
    violation: LOGICAL := FALSE;
  END_LOCAL;
```

```
  (* get all classification_assignment instances with id 'thruster moulded
form design parameter' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'thruster moulded form design
parameter');

  (* get all instances that have class id 'thruster moulded form design
parameter' *)
  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_relationship instances that are
referencing a property_definition instance via relating_property_-
definition,
     filter out those property_definition_relationship instances whose
attribute name has the value 'thruster propeller parameter'
     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_MOULDED_FORM_SCHEMA.' +
'PROPERTY_DEFINITION_RELATIONSHIP' + '.RELATING_PROPERTY_DEFINITION'));
    violation := NOT (SIZEOF(QUERY(t2_inst <* t2_set | t2_inst.name =
'thruster propeller parameter')) = 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 a **applied_classification_assignment** with **assigned_class.name** = 'thruster moulded form design parameter', is referenced by exactly one **property_definition_relationship.relating_property_definition**, where **property_definition_-relationship.name** = 'thruster propeller parameter'.

5.2.4.89 property_definition_with_identification_assignment

The **property_definition_with_identification_assignment** rule specifies a list of entities that require a identification. The identification is defined by the **applied_identification_assignment** attribute.

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 := [ 'moulded form characteristics
definition',
                                   'moulded form functional definition',
                                   'local co ordinate system',
                                   'spacing table',
                                   'hydrostatic definition',
                                   'stability 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_MOULDED_FORM_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 'moulded form characteristics definition', 'moulded form functional definition', ‘local co ordinate system', 'spacing table', ‘hydrostatic definition’, ‘stability definition’ shall require an **applied_identification_assignment** to define the instance identifier.

5.2.4.90 representation_for_appendage_moulded_form_design_parameter

The **representation_for_appendage_moulded_form_design_parameter** 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 'appendage moulded form design parameter'.

EXPRESS specification:

```
*)

RULE representation_for_appendage_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['appendage length','appendage breadth',
                                  'appendage depth','type of appendage'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal TO
      'appendage moulded form design parameter' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
               'USED_REPRESENTATION'))
                     | (temp_prop_def_rep.name =
                      'appendage moulded form design parameter')
                  )
             ) > 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_definiton_representation** with **name** = 'appendage moulded form design parameter', 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 'appendage length', 'appendage breadth', 'appendage depth', or 'type of appendage'.

5.2.4.91 representation_for_bottom_moulded_form_design_parameter

The **representation_for_bottom_moulded_form_design_parameter** rule specifies that the **item** 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 'bottom moulded form design parameter'.

EXPRESS specification:

```
*)
RULE representation_for_bottom_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['bilge radius','rise of floor','aft end
of flat of bottom','front end of flat of bottom','flat of bottom
breadth','rake of keel'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'bottom
moulded form design parameter' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.USED_-
REPRESENTATION'))
                    | (temp_prop_def_rep.name = 'bottom moulded form design
parameter')
                  )
             ) > 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_definiton_representation** with **name** = 'bottom moulded form design parameter', 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 'bilge radius','rise of floor','aft end of flat of bottom','front end of flat of bottom','flat of bottom breadth','rake of keel'.

5.2.4.92 representation_for_bulb_moulded_form_design_parameter

The **representation_for_bulb_moulded_form_design_parameter** rule specifies that the **item** 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 'bulb moulded form design parameter'.

EXPRESS specification:

```
*)
RULE representation_for_bulb_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['bulb length','bulb length from pp','bulb
breadth','bulb breadth pp','bulb depth','bulb depth pp','bulb frame section
area at pp','bulb location'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'bulb
moulded form design parameter' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.USED_-
REPRESENTATION'))
                    | (temp_prop_def_rep.name = 'bulb moulded form design
parameter')
                 )
             ) > 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_definiton_representation** with **name** = 'bulb moulded form design parameter', 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 'bulb length','bulb length from pp','bulb breadth','bulb breadth pp','bulb depth','bulb depth pp','bulb frame section area at pp',or 'bulb location'.

5.2.4.93 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_MOULDED_FORM_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_definiton_representation** with **name** = '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'.

5.2.4.94 representation_for_deck_moulded_form_design_parameter

The **representation_for_deck_moulded_form_design_parameter** rule specifies that the **item** 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 moulded form design parameter'.

EXPRESS specification:

```
*)
RULE representation_for_deck_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['camber','sheer at ap','sheer at fp'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'deck
moulded form design parameter' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.USED_-
REPRESENTATION'))
                   | (temp_prop_def_rep.name = 'deck moulded form design
parameter')
```



```
                    )
               ) > 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_definiton_representation** with **name** = 'deck moulded form design parameter', 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 'camber','sheer at ap','sheer at fp'.

5.2.4.95 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_MOULDED_FORM_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_definiton_representation** with **name** = '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'.

5.2.4.96 representation_for_hull_moulded_form_design_parameter

The **representation_for_hull_moulded_form_design_parameter** 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 'hull moulded form design parameter'

EXPRESS specification:

```
*)
RULE representation_for_hull_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['aft end of parallel midbody at design
draught', 'front end of parallel midbody at design draught', 'aft end of
flat of side', 'front end of flat of side', 'block coefficient', 'prismatic
coefficient', 'max wetted frame section area', ‘waterplane coefficient’];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'hull
moulded form design parameter' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
```



```
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.USED_-
REPRESENTATION'))
                    | (temp_prop_def_rep.name = 'hull moulded form design
parameter')
                  )
               ) > 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_definiton_representation** with **name** = 'hull moulded form design parameter', 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 'aft end of parallel midbody at design draught', 'front end of parallel midbody at design draught', 'aft end of flat of side', 'front end of flat of side','block coefficient', 'prismatic coefficient', 'max wetted frame section area', and ‘waterplane coefficient’.

5.2.4.97 representation_for_hydrostatic_table_constrained

The **representation_for_hydrostatic_table_constrained** rule specifies the **items** attribute of a **representation** with the class id 'hydrostatic table' to have one or more **representation_items** whose class id is 'hydrostatic properties for constant floating position'.

EXPRESS specification:

```
*)
RULE representation_for_hydrostatic_table_constrained
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 'hydrostatic
table' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'hydrostatic table');

  (* get all instances of representation that have class id 'hydrostatic
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 'hydrostatic
properties for constant floating position' *)
  c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'hydrostatic properties for
constant floating position');

  (* get all instances of representation_item that have class id
'hydrostatic properties for constant 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 'hydrostatic 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 'hydrostatic table' shall collect at least one instance of **representation_item** that has an **applied_classification_assignment** whose **assigned_class.name** equals 'hydrostatic properties for constant floating position'.

5.2.4.98 representation_for_hydrostatic_table_restricted

The **representation_for_hydrostatic_table_restricted** rule specifies the **items** attribute of a **representation** with the class id 'hydrostatic table' to have one or more **representation_items** whose class id is ‘hydrostatic property’.

EXPRESS specification:

```
*)
RULE representation_for_hydrostatic_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 'hydrostatic
table' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'hydrostatic table');

  (* get all instances of representation that have class id 'hydrostatic
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 'hydrostatic
property' *)
  c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'hydrostatic property');

  (* get all instances of representation_item that have class id
'hydrostatic property' *)
  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 'hydrostatic 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** those attribute **assigned_class.name** equals 'hydrostatic table' shall collect at least one instance of **representation_item** that has an **applied_classification_assignment** whose **assigned_class.name** equals 'hydrostatic property'.

5.2.4.99 representation_for_hydrostatic_table_restricted_by_class_id

The **representation_for_hydrostatic_table_restricted_by_class_id** rule specifies the **items** attribute of a **representation** with the class id 'hydrostatic 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_hydrostatic_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 'hydrostatic
table' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'hydrostatic table');

  (* get all instances of representation that have class id 'hydrostatic
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** entities.

Formal propositions:

WR1: Every instance of **representation** that has an **applied_classification_assignment** whose attribute **assigned_class.name** equals 'hydrostatic table' shall for each value out of 'mean shell thickness' collect exactly one instance of **representation_item** in its attribute **items** whose **name** attribute equals that value.

5.2.4.100 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_MOULDED_FORM_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_definiton_representation** with **name** = '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'.

5.2.4.101 representation_of_local_coordinate_system_with_position_-reference

The **representation_of_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_of_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_MOULDED_FORM_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_definiton_representation** with **name** = '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'.

5.2.4.102 representation_for_midship_tumble_restricted_by_class_id

The **representation_for_midship_tumble_restricted_by_class_id** rule specifies the **item** attribute of a **representation** with the class id 'midship tumble' 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_midship_tumble_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 := ['tumble out at bottom', 'tumble in at
top', 'tumble out at side', 'tumble in at side'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'midship tumble'
*)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'midship tumble');
```



```
  (* get all instances of representation that have class id 'midship
tumble' *)
  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** entities.

Formal propositions:

WR1: Every instance of **representation** that has an **applied_classification_assignment** whose attribute **assigned_class.name** equals 'midship tumble' shall for each value out of 'tumble out at bottom', 'tumble in at top', 'tumble out at side', or 'tumble in at side' collect exactly one instance of **representation_item** in its attribute **items** whose **name** attribute equals that value.

5.2.4.103 representation_for_moulded_form_function_parameters

The **representation_for_moulded_form_function_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 'moulded form function parameters'.

EXPRESS specification:

```
*)
RULE representation_for_moulded_form_function_parameters
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['function'];
    violation: LOGICAL := FALSE;
  END_LOCAL;
```



```
  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'moulded
form function parameters' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENT
ATION'))
                      | (temp_prop_def_rep.name = 'moulded form 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_definiton_representation** with **name** = 'moulded form 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 'function'.

5.2.4.104 representation_for_offset_point_table_model_for_point

The **representation_for_offset_point_table_model_for_point** rule specifies the **item_element** attribute of a **compound_representation_item** with the class id 'section of offset point table' to have in the **list_representation_item** for each entry in the list one or more **representation_item** whose **name** attribute has the value given in the list.

EXPRESS specification:

```
*)
RULE representation_for_offset_point_table_model_for_point
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 := ['section point'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'section of offset point
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 compound_representation_item found above; stop, if
one of
     them has not one or more 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 'section of offset point table' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall for each value out of 'section point' collect one or more instances of **representation_item** whose **name** attribute equals that value.

5.2.4.105 representation_for_offset_point_table_model_for_section

The **representation_for_offset_point_table_model_for_section** rule specifies the **item_element** attribute of a **compound_representation_item** with the class id 'offset point table model' to have in the **list_representation_item** for each entry in the list one or more **representation_item** whose **name** attribute has the value given in the list.

EXPRESS specification:

(Blank page)

```
*)
RULE representation_for_offset_point_table_model_for_section
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 point table section'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'offset point table model');

  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 one or more 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 'offset point table model' shall have its attribute **item_element** instantiated as a **list_representation_item** which shall for each value out of 'offset point table section' collect one or more instances of **representation_item** whose **name** attribute equals that value.

5.2.4.106 representation_for_offset_table_shape_representation_restricted

The **representation_for_offset_table_shape_representation_restricted** rule specifies the **items** attribute of a **representation** with the class id 'offset table shape representation' to have one or more **representation_items** whose class id is 'offset point table model'.

EXPRESS specification:

```
*)
RULE representation_for_offset_table_shape_representation_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 'offset table
shape representation' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                i.ASSIGNED_CLASS.NAME = 'offset table shape
representation');

  (* get all instances of representation that have class id 'offset table
shape 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;

(* get all classification_assignment instances with id 'offset point table
model' *)
  c2_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'offset point table model');

  (* get all instances of representation_item that have class id 'offset
point table model' *)
  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 'offset table shape representation' *)
  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 than 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 'offset table shape representation' shall collect at least one instance of **representation_item** that has an **applied_classification_assignment** whose **assigned_-class.name** equals 'offset point table model'.

5.2.4.107 representation_for_propeller_location_restricted_by_class_id

The **representation_for_propeller_location_restricted_by_class_id** rule specifies the **item** attribute of a **representation** with the class id 'propeller location' 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_propeller_location_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 := ['shaft line inclination x', 'shaft line
inclination y', 'shaft line location', 'propeller location'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'propeller
location' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'propeller location');

  (* get all instances of representation that have class id 'propeller
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 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** entities.

Formal propositions:

WR1: Every instance of **representation** that has an **applied_classification_assignment** whose attribute **assigned_class.name** equals 'propeller location' shall for each value out of 'shaft line inclination x', 'shaft line inclination y', 'shaft line location', or 'propeller location' collect exactly one instance of **representation_item** in its attribute **items** whose **name** attribute equals that value.

5.2.4.108 representation_for_propeller_moulded_form_design_parameter

The **representation_for_propeller_moulded_form_design_parameter** rule specifies that the **item** 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 'propeller moulded form design parameter'.

EXPRESS specification:

```
*)
RULE representation_for_propeller_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['type of propulsion', 'propeller
diameter', 'chord length at 0 7 radius', 'thickness at 0 7 radius', 'number
of propeller blades', 'expanded area ratio', 'hub diameter ratio', 'nominal
design pitch ratio', 'type of propeller blades', 'rake', 'skew', 'design
sense of rotation'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'propeller
moulded form design parameter' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.USED_-
REPRESENTATION'))
          | (temp_prop_def_rep.name = 'propeller moulded form design
parameter')
                   )
             ) > 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_definiton_representation** with **name** = 'propeller moulded form design parameter', 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 propulsion', 'propeller diameter', 'chord length at 0 7 radius', 'thickness at 0 7 radius', 'number of propeller blades', 'expanded area ratio', 'hub diameter ratio', 'nominal design pitch ratio', 'type of propeller blades', 'rake', 'skew', 'design sense of rotation'.

5.2.4.109 representation_for_rudder_moulded_form_design_parameter

The **representation_for_rudder_moulded_form_design_parameter** rule specifies that the **item** 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 'rudder moulded form design parameter'.

EXPRESS specification:

```
*)
RULE representation_for_rudder_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['rudder height', 'rudder mean height',
'rudder length', 'rudder mean length', 'rudder thickness', 'projected
rudder area', 'type of the rudder', 'rudder location'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'rudder
moulded form design parameter' *)
  reps := QUERY(
```



```
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.USED_-
REPRESENTATION'))
                    | (temp_prop_def_rep.name = 'rudder moulded form design
parameter')
                  )
             ) > 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_definiton_representation** with **name** = 'rudder moulded form design parameter', 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 'rudder height', 'rudder mean height', 'rudder length', 'rudder mean length', 'rudder thickness', 'projected rudder area', 'type of the rudder',or 'rudder location'.

5.2.4.110 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'.

5.2.4.111 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** entities.

Formal propositions:

WR1: Every instance of **representation** that has an **applied_classification_assignment** whose attribute **assigned_class.name** equals 'stability table' shall for each value out of 'mean shell thickness' collect exactly one instance of **representation_item** in its attribute **items** whose **name** attribute equals that value.

5.2.4.112 representation_for_thruster_moulded_form_design_parameter

The **representation_for_thruster_moulded_form_design_parameter** rule specifies that the **item** 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 'thruster moulded form design parameter'.

EXPRESS specification:

```
*)
RULE representation_for_thruster_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['thruster tunnel diameter', 'thruster
tunnel min length', 'thruster tunnel max length', 'geometric thruster
location', 'thruster location'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* find all instances of representation  which are used
     by a property_definition_representation with name equal to 'thruster
moulded form design parameter' *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.USED_-
REPRESENTATION'))
                    | (temp_prop_def_rep.name = 'thruster moulded form
design parameter')
                 )
             ) > 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_definiton_representation** with **name** = 'thruster moulded form design parameter', 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 'thruster tunnel diameter', 'thruster tunnel min length', 'thruster tunnel max length', 'geometric thruster location', or 'thruster location'.

5.2.4.113 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_MOULDED_FORM_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.**

5.2.4.114 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_items** of type **value_representation_item** and/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_MOULDED_FORM_SCHEMA.VALUE_REPRESENTATION_ITEM'
          IN TYPEOF(REPRESENTATION[i].ITEMS[j])) OR
          ('SHIP_MOULDED_FORM_SCHEMA.GEOMETRIC_REPRESENTATION_ITEM'
           IN TYPEOF(REPRESENTATION[i].ITEMS[j]))) THEN
        has_guac :=
('SHIP_MOULDED_FORM_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_items** or **geometric_-representation_items** has a **representation_context** that is a **global_unit_assigned_context**.

5.2.4.115 representation_items_appendage_moulded_form_design_-parameter

The **representation_items_appendage_moulded_form_design_parameter** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** with the **name** attribute value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** with the **name** attribute having a value of 'appendage moulded form design parameter'.

EXPRESS specification:

```
*)
```



```
RULE representation_items_appendage_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['moulded form outer surface',
      'moulded form displacement', 'user def appendage type'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
     'appendage moulded form design parameter'  *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
      'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
      'USED_REPRESENTATION'))
                    | (temp_prop_def_rep.name =
                    'appendage moulded form design parameter')
                  )
              ) > 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** = 'appendage moulded form design parameter', the values 'moulded form outer surface', 'moulded form displacement', or 'user def appendage type' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.116 representation_items_for_bottom_moulded_form_design_-parameter

The **representation_items_for_bottom_moulded_form_design_parameter** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** with the **name** attribute value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** with the a **name** attribute of value 'bottom moulded form design parameter'.

EXPRESS specification:

```
*)

RULE representation_items_for_bottom_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['moulded form outer surface',
                                 'moulded form displacement'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
     'bottom moulded form design parameter' *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
              'USED_REPRESENTATION'))
                           | (temp_prop_def_rep.name =
                           'bottom moulded form design parameter')
                     )
               ) > 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** = 'bottom moulded form design parameter', the values 'moulded form outer surface', or 'moulded form displacement' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.117 representation_items_for_bulb_moulded_form_design_-parameter

The **representation_items_for_bulb_moulded_form_design_parameter** rule specifies the **item** attribute of a **representation** to have for each entry in the list zero or one **representation_item** with the **name** attribute value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** with the **name** attribute value 'bulb moulded form design parameter'.

EXPRESS specification:

```
*)

RULE representation_items_for_bulb_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['moulded form outer surface',
                                 'moulded form displacement'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
     'bulb moulded form design parameter'  *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
            'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
            'USED_REPRESENTATION'))
                           | (temp_prop_def_rep.name =
                           'bulb moulded form design parameter')
                       )
              ) > 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** = 'bulb moulded form design parameter', the values 'moulded form outer surface', or 'moulded form displacement' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.118 representation_items_for_deck_moulded_form_design_parameter

The **representation_items_for_deck_moulded_form_design_parameter** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** with the **name** attribute value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** with the **name** attribute of value 'deck moulded form design parameter'.

EXPRESS specification:

```
*)

RULE representation_items_for_deck_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['moulded form outer surface',
                                 'moulded form displacement'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
     'deck moulded form design parameter'   *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
             'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
             'USED_REPRESENTATION'))
                        | (temp_prop_def_rep.name =
                        'deck moulded form design parameter')
                     )
               ) > 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** = 'deck moulded form design parameter', the values 'moulded form outer surface',or 'moulded form displacement' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.119 representation_items_for_hull_moulded_form_design_-parameter

The **representation_items_for_hull_moulded_form_design_parameter** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** with the **name** attribute value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** with the **name** attribute of value 'hull moulded form design parameter'.

EXPRESS specification:

```
*)

RULE representation_items_for_hull_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['moulded form outer surface',
                                 'moulded form displacement',
                                 'waterline angle of entrance at stern',
                                 'waterline angle of entrance at bow',
                                 'max frame section area location',
                                 'hull length pp',
                                 'hull length waterline',
                                 'hull breadth',
                                 'hull depth',
                                 'hull design draught',
                                 'gunwale radius'];
    found: LOGICAL := FALSE;
  END_LOCAL;
```

```
  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
     'hull moulded form design parameter'  *)
  reps := QUERY(
            temp_rep <* representation |
               SIZEOF (
                  QUERY(
                     temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
              'USED_REPRESENTATION'))
                           | (temp_prop_def_rep.name =
                           'hull moulded form design parameter')
                     )
               ) > 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** = 'hull moulded form design parameter', the values 'moulded form outer surface', 'moulded form displacement', 'waterline angle of entrance at stern', 'waterline angle of entrance at bow', 'max frame section area location', 'hull length pp', 'hull length waterline', 'hull breadth', 'hull depth', 'hull design draught',or 'gunwale radius' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.120 representation_items_for_moulded_form_design_parameters

The **representation_items_for_moulded_form_design_parameters** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** with the **name** attribute value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** with the **name** attribute of value 'moulded form design parameters'.

EXPRESS specification:

```
*)

RULE representation_items_for_moulded_form_design_parameters
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['status'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
     'moulded form design parameters'  *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
            'USED_REPRESENTATION'))
                      | (temp_prop_def_rep.name =
                      'moulded form design 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** = 'moulded form design parameters', the value 'status' shall not occur more than once as a value of the **representation_item** attribute **name**.

5.2.4.121 representation_items_for_moulded_form_function_parameters

The **representation_items_for_moulded_form_function_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 'moulded form function parameters'.

EXPRESS specification:

```
*)

RULE representation_items_for_moulded_form_function_parameters
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 'moulded
form function parameters'
   *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENT
ATION'))
                     | (temp_prop_def_rep.name = 'moulded form 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** = 'moulded form function parameters', the value 'user def function' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.122 representation_items_propeller_moulded_form_design_parameter

The **representation_items_propeller_moulded_form_design_parameter** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** with the **name** attribute value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** with the **name** attribute of value 'propeller moulded form design parameter'.

EXPRESS specification:

```
*)

RULE representation_items_propeller_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['blade_mean_height',
                                 'moulded form outer surface',
                                 'moulded form displacement'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
     'propeller moulded form design parameter'  *)
  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
         'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
         'USED_REPRESENTATION'))
                    | (temp_prop_def_rep.name =
                    'propeller moulded form design parameter')
                 )
              ) > 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** = 'propeller moulded form design parameter', the values 'blade_mean_height', 'moulded form outer surface', or 'moulded form displacement' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.123 representation_items_for_rudder_moulded_form_design_-parameter

The **representation_items_for_rudder_moulded_form_design_parameter** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** with the **name** attribute value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** with the **name** attribute of value 'rudder moulded form design parameter'.

EXPRESS specification:

```
*)

RULE representation_items_for_rudder_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['moulded form outer surface',
                                 'moulded form displacement'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
     'rudder moulded form design parameter'  *)
  reps := QUERY(
           temp_rep <* representation |
```



```
              SIZEOF (
                 QUERY(
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
             'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
             'USED_REPRESENTATION'))
                           | (temp_prop_def_rep.name =
                            'rudder moulded form design parameter')
                       )
              ) > 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** = 'rudder moulded form design parameter', the values 'moulded form outer surface', or 'moulded form displacement' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.124 representation_items_of_thruster_moulded_form_design_-parameter

The **representation_items_of_thruster_moulded_form_design_parameter** rule specifies the **items** attribute of a **representation** to have for each entry in the list zero or one **representation_item** with the **name** attribute value given in the list if the **representation** is the **used_representation** in a **property_definition_representation** with the **name** attribute of value 'thruster moulded form design parameter'.

EXPRESS specification:

```
*)

RULE representation_items_of_thruster_moulded_form_design_parameter
FOR (representation);
  LOCAL
    reps: BAG OF REPRESENTATION := [];
    arg_list: LIST OF STRING := ['moulded form outer surface',
```

```
                                  'moulded form displacement'];
    found: LOGICAL := FALSE;
  END_LOCAL;

  (* Find all instances of representation  which are used
     by a property_definition_representation with name equal to
       'thruster moulded form design parameter'  *)
  reps := QUERY(
          temp_rep <* representation |
             SIZEOF (
                QUERY(
                   temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
           'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
           'USED_REPRESENTATION'))
                          | (temp_prop_def_rep.name =
                          'thruster moulded form design parameter')
                      )
             ) > 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** = 'thruster moulded form design parameter', the values 'moulded form outer surface',or 'moulded form displacement' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.125 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 instances 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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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 a **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** = '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 an 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'.

5.2.4.126 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_MOULDED_FORM_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** referenced by the **items** attribute of every **representation** referenced by the **definition** attribute of a **property_definition_representation** with **name** = 'class notation', the values of 'ice class notation', 'service factor' or 'approval required for heavy cargo' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.127 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_MOULDED_FORM_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**.

5.2.4.128 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;

  reps := QUERY(
          temp_rep <* representation |
```



```
            SIZEOF (
               QUERY(
                  temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_MOULDED_FORM_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;

(*
```

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** = 'principal characteristics', the values of 'block coefficient', 'design draught', 'design deadweight', 'min draught at fp', 'max draught at fp', 'min draught at ap', 'max draught at ap' shall not occur more than once as values of the **representation_item** attribute **name**.

5.2.4.129 representation_restricted_by_name_class_notation

The **representation_restricted_by_name_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_restricted_by_name_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_MOULDED_FORM_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 **property_definiton_representation** with **name** = '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'.

5.2.4.130 representation_restricted_by_name_class_parameters

The **representation_restricted_by_name_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_restricted_by_name_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_MOULDED_FORM_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;
(*
```

Argument definitions:

representation: the set of all instances of **representation** entities

Formal propositions:

WR1: Every **property_definiton_representation** with **name** = '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'.

5.2.4.131 representation_restricted_by_name_principal_characteristics

The **representation_restricted_by_name_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_restricted_by_name_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_MOULDED_FORM_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;

(*
```

Argument definitions:

representation: the set of all instances of **representation** entities

Formal propositions:

WR1: Every **property_definiton_representation** with **name** = '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', or 'moulded depth'.

5.2.4.132 representation_restricted_by_name_ship_overall_dimensions

The **representation_restricted_by_name_ship_overall_dimensions** 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 'ship overall dimensions'.

EXPRESS specification:

```
*)
RULE representation_restricted_by_name_ship_overall_dimensions
FOR (representation);
  LOCAL
    reps:      BAG OF REPRESENTATION := [];
    arg_list:  LIST OF STRING := ['overall breadth’, ‘overall depth’,
     ‘overall length’, ‘stem overhang’', 'stern overhang'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  reps := QUERY(
           temp_rep <* representation |
              SIZEOF (
                 QUERY(
```



```
                    temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,
  'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.'+
  'USED_REPRESENTATION'))
                    | (temp_prop_def_rep.name =
                    'ship overall dimensions')
                   )
              ) > 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_definiton_representation** with **name** = 'ship overall dimensions ', 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 'overall breadth', ‘overall depth’, ‘overall length’, 'stem overhang', or 'stern overhang'.

5.2.4.133 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;
  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;

(*
```

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 'revision'.

5.2.4.134 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;
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;

(*
```

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_-classification** is a **group** with attribute **name** equals 'revision with context' shall be referenced by exactly one instance of type **applied_group_assignment** whose **role** equals 'context of revision' through attribute **assigned_group**.

5.2.4.135 ship_curve_has_name

The **ship_curve_has_name** rule specifies the **item_element** attribute of a **compound_-representation_item** with the class id 'ship curve' 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 ship_curve_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 := ['side condition', 'curve shape'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'ship curve' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'ship curve');

  (* get all instances of compound_representation_item that have class id
'ship curve' *)
  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 'ship curve' shall have its attribute **item_-element** instantiated as a **list_representation_item** which shall for each value out of 'side condition', or 'curve shape' collect exactly one instance of **representation_item** whose **name** attribute equals that value.

5.2.4.136 ship_curve_segment_has_class

The **ship_curve_segment_has_class** rule specifies the **item_element** attribute of a **compound_-representation_item** with the class id 'ship curve segment' to have in the **list_representation_item** exactly one **compound_representation_item** with the class id 'ship curve'.

EXPRESS specification:

```
*)
RULE ship_curve_segment_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 'ship curve
segment' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'ship curve segment');

  (* get all instances of compound_representation_item that have class id
'ship curve segment' *)
  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 'ship curve' *)
  c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'ship curve');

  (* get all instances of compound_representation_item that have class id
'ship curve' *)
  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 equal 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 'ship curve segment' 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 'ship curve'.

5.2.4.137 ship_curve_with_spacing_position_has_class

The **ship_curve_with_spacing_position_has_class** rule specifies the **item_element** attribute of a **compound_representation_item** with the class id 'ship curve with spacing position' to have in the **list_representation_item** exactly one **compound_representation_item** with the class id 'spacing position'

EXPRESS specification:

```
*)
RULE ship_curve_with_spacing_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 'ship curve with
spacing position' *)
```

```
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'ship curve with spacing
position');

  (* get all instances of compound_representation_item that have class id
'ship curve with 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;

(* get all classification_assignment instances with id 'spacing position'
*)
  c_a_set2 := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                   i.ASSIGNED_CLASS.NAME = 'spacing position');

  (* get all instances of compound_representation_item that have class id
'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;

  (* 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 equal 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 'ship curve with spacing position' 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'.

5.2.4.138 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** that defines an entity with an attribute **name** has either the value '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;

  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_MOULDED_FORM_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 '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'.

5.2.4.139 ship_moulded_form_revision_has_description

The **ship_moulded_form_revision_has_description** rule specifies that for an instance of **product_-definition_relatonship** with class id 'ship moulded form revision' the optional attribute **description** is instantiated.

EXPRESS specification:

```
*)
RULE ship_moulded_form_revision_has_description
FOR (product_definition_relationship);
LOCAL
    t1_set:  SET OF product_definition_relationship := [];
    violate: LOGICAL := FALSE;
  END_LOCAL;

(* get all instances of product_definition_relatonship being classified as
'ship moulded form revision' *)
t1_set := QUERY(i <* product_definition_relationship |
VALUE_IN(WHICH_CLASS(i), 'ship moulded form revision'));

(* from all instances found above:
   find those for which attribute description is not instanciated
*)

violate := (SIZEOF(QUERY(k <* t1_set | NOT EXISTS (k.description))) > 0);

WHERE
    wr1: NOT violate;
END_RULE;

(*
```

Argument definitions:

product_definition_relatonship: the set of all instances of **product_definition_relatonship** entities.

Formal propositions:

WR1: The optional attribute **description** shall exist for every instance of **product_definition_-relatonship** that has an **applied_classification_assignment** whose **assigned_class.name** equals 'ship moulded form revision'.

5.2.4.140 ship_overall_dimensions_has_properties

The **ship_overall_dimensions_has_properties** rule specifies that a **product_definition** with a class

id 'ship overall dimensions' is referenced by one **property_definition_representation** with the **name** 'ship overall dimensions' via a **property_definition**.

EXPRESS specification:

```
*)
RULE ship_overall_dimensions_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 =
                  'ship overall dimensions');
            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 = 'ship overall dimensions');
            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 'ship overall dimensions' 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 'ship overall dimensions'.

5.2.4.141 ship_point_compound_representation_has_name

The **ship_point_compound_representation_has_name** rule specifies the **item_element** attribute of a **compound_representation_item** with the class id 'ship point' 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 ship_point_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 := ['point shape'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'ship point' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'ship point');

  (* get all instances of compound_representation_item that have class id
'ship point' *)
  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 'ship point' shall have its attribute **item_-element** instantiated as a **list_representation_item** which shall for each value out of 'point shape' collect exactly one instance of **representation_item** whose **name** attribute equals that value.

5.2.4.142 ship_surface_compound_representation_has_name

The **ship_surface_compound_representation_has_name** rule specifies the **item_element** attribute of a **compound_representation_item** with the class id 'ship surface' 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 ship_surface_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 := ['surface shape'];
    violation: LOGICAL := FALSE;
  END_LOCAL;

  (* get all classification_assignment instances with id 'ship surface' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'ship surface');

  (* get all instances of compound_representation_item that have class id
'ship surface' *)
  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 'ship surface' shall have its attribute **item_-element** instantiated as a **list_representation_item** which shall for each value out of 'surface shape' collect exactly one instance of **representation_item** whose **name** attribute equals that value.

5.2.4.143 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 out of 'position number',or 'position' collect exactly one instance of **representation_item** whose **name** attribute equals that value.

5.2.4.144 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'.

5.2.4.145 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 for each value out of 'offset' collect exactly one instance of **representation_item** whose **name** attribute equals that value.

5.2.4.146 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'.

5.2.4.147 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.

5.2.4.148 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.

5.2.4.149 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_MOULDED_FORM_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_classification** is a **group** with attribute **name** equals 'approval history' must be unique.

5.2.4.150 user_def_appendage_type_description_required

The **user_def_appendage_type_description_required** rule specifies a representation that has a **descriptive_representation_item** with **name** 'type of appendage' and **description** 'user defined' to have another **descriptive_representation_item** with **name** 'user def appendage type'.

EXPRESS specification:

```
*)

RULE user_def_appendage_type_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_MOULDED_FORM_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
       IN TYPEOF(r)) AND
       (r.NAME = 'type of appendage') AND
       (r\DESCRIPTIVE_REPRESENTATION_ITEM.DESCRIPTION = 'user defined'))) >
0)
                 AND
                 (SIZEOF(QUERY(r <* REPRESENTATION[i].ITEMS |
                         (r.NAME = 'user def appendage 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 appendage' and **description** 'user defined' in its set of items shall have another **descriptive_representation_item** with **name** 'user def appendage type' in its set of **items**.

5.2.4.151 user_def_function_description_required

The **user_def_function_description_required** rule specifies a **representation** that has a **descriptive_representation_item** with **name** 'function' 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_MOULDED_FORM_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
                          IN TYPEOF(r)) AND
                         (r.NAME = 'function') 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** 'function' 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**.

5.2.4.152 valid_product_definition_for_class_moulded_form

The **valid_product_definition_for_class_moulded_form** specifies a **product_definition** of class 'ship moulded form' that also is a **group.** The **group** shall be referenced by an **applied_group_assignment** whose attribute **role.name** = ‘item structure’ and which collects **product_definition** instances with the class 'moulded form', and **product_definition_relationship** instances with class ‘moulded form relationship’ via its **applied_group_assignment.items** attribute.

EXPRESS specification:

```
*)
RULE
 valid_product_definition_for_class_moulded_form
FOR (APPLIED_CLASSIFICATION_ASSIGNMENT,APPLIED_GROUP_ASSIGNMENT);
LOCAL
    c_a_set: SET OF APPLIED_CLASSIFICATION_ASSIGNMENT:= [];
    t1_set:  SET OF PRODUCT_DEFINITION := [];
    gr_ass:    SET OF APPLIED_GROUP_ASSIGNMENT := [];
    groups:    SET OF GROUP := [];
    violation: LOGICAL := FALSE;
    violate1, violate2 : LOGICAL;
  END_LOCAL;

  (* get all classification_assignment instances with id 'ship
moulded form' *)
  c_a_set := QUERY(i <* APPLIED_CLASSIFICATION_ASSIGNMENT |
                  i.ASSIGNED_CLASS.NAME = 'ship moulded form');

  (* get all instances of product_definition that have class id
'ship moulded form' *)
  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 the groups associated with these product_definitions *)
  gr_ass := QUERY(i <* APPLIED_GROUP_ASSIGNMENT |
                 i.role.name = 'equivalence');
  REPEAT i := 1 TO HIINDEX(gr_ass);
    REPEAT j := 1 TO HIINDEX(gr_ass[i].items);  -- should always be
just one
      IF (gr_ass[i].items[j] IN t1_set) THEN
        groups := groups + gr_ass[i].assigned_group;
      END_IF;
    END_REPEAT;
  END_REPEAT;

  (* get the non-empty group_assignments having role 'item
structure' and
     referencing these groups
   *)
  gr_ass := QUERY(i <* APPLIED_GROUP_ASSIGNMENT |
                 (SIZEOF(i.items) <> 0) AND
                 (i.role.name = 'item structure') AND
                 (i.assigned_group IN groups));

  (* check if there are only instances of class 'moulded form' and
of  class ‘moulded form relationship’  in each
group_assignment.items
   *)
  REPEAT i := 1 TO HIINDEX(gr_ass) WHILE NOT(violation);
    REPEAT j := 1 TO HIINDEX(gr_ass[i].items) WHILE NOT(violation);
      violate1 := VALUE_IN(WHICH_CLASS(gr_ass[i].items[j]),’moulded
```



```
form');
      violate2 := VALUE_IN(WHICH_CLASS(gr_ass[i].items[j]),'moulded
form relationship');
      violation := NOT(violate1 OR violate2);
    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.
applied_group_assignment: the set of all instances of **applied_group_assignment** entities.

Formal propositions:

WR1: For every instance of **product_definition** of class 'ship moulded form' that also is a **group** referenced by an **applied_group_assignment** whose attribute **role.name** = 'item structure', the **applied_group_assignment** shall only collect **product_definition** instances with the class 'moulded form', and **product_definition_relationship** instances with class 'moulded form relationship' via its **applied_group_assignment.items** attribute.

5.2.4.153 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;
  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;

(*
```

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 'version creation'.

5.2.4.154 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;
  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;

(*
```

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 'version deletion'.

5.2.4.155 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;


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;

(*
```

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 referenced by exactly one instance of type **applied_group_assignment** through attribute **assigned_group**.

5.2.4.156 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**.

5.2.4.157 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.name** '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 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** identifies an entity with attribute **name** equal 'version history' shall be referenced by exactly one instance of type **applied_group_assignment** whose role equals 'current version' through attribute **assigned_group**.

5.2.4.158 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**.

5.2.4.159 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;
  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;

(*
```

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 'version modification'.

5.2.4.160 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_MOULDED_FORM_SCHEMA.IDENTIFICATION_ASSIGNMENT_RELATIONSHIP.'
             +'RELATING_IDENTIFICATION_ASSIGNMENT'))) > 0) OR
      (SIZEOF(USEDIN(applied_identification_assignment[i],
      ('SHIP_MOULDED_FORM_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'.

5.2.4.161 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 instanciated.

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;
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;

(*
```

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 'version relationship'.

5.2.4.162 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.

5.2.4.163 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'.

5.2.4.164 versioned_action_request_with_identification_assignment

The **versioned_action_request_with_identification_assignment** rule specifies a list of entities that require a 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_MOULDED_FORM_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.

5.2.4.165 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 structures 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_MOULDED_FORM_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 methods supported, and the relevant 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 Information Conformance Statement (PICS) proforma lists the options or combinations of options that may be included in the implementation. The PICS proforma is provided in annex D.

This part of ISO 10303 provides 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 hydrostatic data (see 4.1.6);

— Class 2 is a conformance class to exchange moulded form geometry as an offset table (see 4.1.9);

— Class 3 is a conformance class to exchange moulded form geometry as a wireframe representation (see 4.1.13 and 4.1.15);

— Class 4 is a conformance class to exchange moulded form geometry as a surface representation (see 4.1.13 and 4.1.14);

— Class 5 is a conformance class to exchange moulded form geometry as a surface representation with hull applicability (see 4.1.5, 4.1.13 and 4.1.14);.

Conformance to a particular class requires that all AIM elements defined as part of that class be supported. Table 2 defines the conformance classes to which each Unit of Functionality belongs. Table 3 defines the classes to which each AIM element belongs.

NOTE AP 216 Test Case Definitions [13] defines the abstract test suite to be used in the assessment of conformance 2)

Table 2 — Conformance classes

Unit of Functionality	Conformance class				
	Class 1	Class 2	Class 3	Class 4	Class 5
basic_geometry		X	X	X	X
configuration_management	X	X	X	X	X
definitions	X	X	X	X	X
external_references	X	X	X	X	X
hull_class_applicability					X
hydrostatics	X				
items	X	X	X	X	X
location_concepts	X	X	X	X	X
offset_table_representations		X			
ship_design_parameter	X	X	X	X	X
ship_general_characteristics	X	X	X	X	X
ship_measures	X	X	X	X	X
ship_moulded_forms	X	X	X	X	X
surface_representations (AIC 508)				X	X
wireframe_representations (AIC 501)			X		

Table 3 — Conformance class elements

AIM element	Class				
	1	2	3	4	5
Action	X	X	X	X	X
Action_assignment	X	X	X	X	X
Action_relationship	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
Amount_of_substance_unit	X	X	X	X	X
Application_context	X	X	X	X	X

AIM element	Class				
	1	2	3	4	5
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_assignement					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
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	X	X
Axis2_placement_2d	X	X	X	X	X
Axis2_placement_3d	X	X	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

AIM element	Class				
	1	2	3	4	5
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
Classification_assignment	X	X	X	X	X
Classification_role	X	X	X	X	X
Closed_shell				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_edge_set			X		
Connected_face_set				X	X
Context_dependent_unit	X	X	X	X	X
Conversion_based_unit	X	X	X	X	X
Coordinated_universal_time_offset	X	X	X	X	X

AIM element	Class				
	1	2	3	4	5
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
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
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	X
Edge_based_wireframe_model			X		
Edge_based_wireframe_shape_representation	X	X	X	X	X
Edge_curve		X	X	X	X
Edge_loop				X	X
Effectivity					X
Effectivity_assignment					X

AIM element	Class				
	1	2	3	4	5
Electric_current_unit	X	X	X	X	X
Elementary_surface	X	X	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
Externally_defined_item	X	X	X	X	X
Face		X	X	X	X
Face_based_surface_model				X	X
Face_bound				X	X
Face_outer_bound				X	X
Face_surface		X	X	X	X
Faceted_brep				X	X
Founded_item			X	X	X
Functionally_defined_transformation			X	X	X
Geometric_curve_set				X	X
Geometric_representation_context	X	X	X	X	X
Geometric_representation_item	X	X	X	X	X
Geometric_set				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

AIM element	Class				
	1	2	3	4	5
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
Length_measure_with_unit	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
Luminous_intensity_unit	X	X	X	X	X
Manifold_solid_brep				X	X
Mapped_item	X	X	X	X	X
Mass_measure_with_unit	X	X	X	X	X
Mass_unit	X	X	X	X	X
Measure_with_unit	X	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	X	X
Object_role	X	X	X	X	X
Offset_curve_3d			X	X	X
Offset_surface			X	X	X
Open_shell				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

AIM element	Class				
	1	2	3	4	5
Organizational_address	X	X	X	X	X
Organizational_project	X	X	X	X	X
Oriented_closed_shell				X	X
Oriented_edge		X	X	X	X
Oriented_face		X	X	X	X
Oriented_open_shell				X	X
Oriented_path				X	X
Oriented_surface			X	X	X
Parabola			X	X	X
Parametric_representation_context			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	X	X
Plane	X	X	X	X	X
Plane_angle_measure_with_unit	X	X	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
Point_replica			X	X	X

AIM element	Class				
	1	2	3	4	5
Poly_loop				X	X
Polyline			X	X	X
Product	X	X	X	X	X
Product_category	X	X	X	X	X
Product_category_relationship	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_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
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
Serial_numbered_effectivity					X

Table 3 - Conformance class elements (concluded)

AIM element	Class				
	1	2	3	4	5
Shape_aspect	X	X	X	X	X
Shape_definition_representation	X	X	X	X	X
Shape_representation	X	X	X	X	X
Si_unit	X	X	X	X	X
Solid_angle_measure_with_unit	X	X	X	X	X
Solid_angle_unit	X	X	X	X	X
Solid_model				X	X
Spherical_surface			X	X	X
Subface		X	X	X	X
Surface	X	X	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	X
Time_unit	X	X	X	X	X
Topological_representation_item				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
Vector			X	X	X
Versioned_action_request	X	X	X	X	X
Vertex			X	X	X

AIM element	Class				
	1	2	3	4	5
Vertex_loop				X	X
Vertex_point		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 discrepancy between the short form and this expanded listing, the expanded listing shall be used.

```
SCHEMA ship_moulded_form_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
    (product,
     product_definition,
     property_definition,
     product_definition_relationship,
     product_definition_shape,
     product_related_product_category,
     action_request_solution,
     executed_action);
  END_TYPE; -- action_item

  TYPE action_request_item = SELECT
    (action,
     executed_action);
  END_TYPE; -- action_request_item

  TYPE ahead_or_behind = ENUMERATION OF
    (ahead,
     exact,
     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,
     property_definition,
     product_related_product_category);
  END_TYPE; -- approval_item
```



```
TYPE area_measure = REAL;
END_TYPE; -- area_measure

TYPE attribute_type = SELECT
  (label,
   text);
END_TYPE; -- attribute_type

TYPE axis2_placement = SELECT
  (axis2_placement_2d,
   axis2_placement_3d);
END_TYPE; -- axis2_placement

TYPE b_spline_curve_form = ENUMERATION OF
  (polyline_form,
   circular_arc,
   elliptic_arc,
   parabolic_arc,
   hyperbolic_arc,
   unspecified);
END_TYPE; -- b_spline_curve_form

TYPE b_spline_surface_form = ENUMERATION OF
  (plane_surf,
   cylindrical_surf,
   conical_surf,
   spherical_surf,
   toroidal_surf,
   surf_of_revolution,
   ruled_surf,
   generalised_cone,
   quadric_surf,
   surf_of_linear_extrusion,
   unspecified);
END_TYPE; -- b_spline_surface_form

TYPE boolean_operand = SELECT
  (solid_model);
END_TYPE; -- boolean_operand

TYPE celsius_temperature_measure = REAL;
END_TYPE; -- celsius_temperature_measure

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,
     axis2_placement_3d,
     compound_representation_item,
     external_source,
     document,
     document_reference,
     edge_curve,
     executed_action,
     group,
     identification_assignment_relationship,
     measure_with_unit,
     product,
     product_definition,
     product_definition_relationship,
     product_definition_shape,
     product_related_product_category,
     property_definition,
     property_definition_representation,
     representation,
     representation_relationship,
     shape_representation,
     vertex_point,
     versioned_action_request);
  END_TYPE; -- classification_item

  TYPE compound_item_definition = SELECT
    (list_representation_item,
     set_representation_item);
  END_TYPE; -- compound_item_definition

  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_request_solution,
     executed_action,
     versioned_action_request,
     product_definition,
     action);
  END_TYPE; -- date_and_time_item

  TYPE date_time_or_event_occurrence = SELECT
    (date_time_select);
  END_TYPE; -- date_time_or_event_occurrence

  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 derived_property_select = SELECT
  (property_definition);
END_TYPE; -- derived_property_select

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,
   property_definition,
   product_definition);
END_TYPE; -- document_reference_item

TYPE effectivity_item = SELECT
  (product_definition,
   property_definition,
   product_definition_shape,
```

```
    product_related_product_category);
  END_TYPE; -- effectivity_item

  TYPE electric_current_measure = REAL;
  END_TYPE; -- electric_current_measure

  TYPE external_identification_item = SELECT
    (action,
     document,
     product,
     product_definition,
     product_definition_relationship,
     property_definition);
  END_TYPE; -- external_identification_item

  TYPE founded_item_select = SELECT
    (founded_item,
     representation_item);
  END_TYPE; -- founded_item_select

  TYPE geometric_set_select = SELECT
    (point,
     curve,
     surface);
  END_TYPE; -- geometric_set_select

  TYPE group_item = SELECT
    (approval,
     identification_assignment_relationship,
     product_definition,
     product_definition_relationship);
  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,
```



```
   product,
   product_definition,
   property_definition,
   product_definition_shape,
   product_related_product_category,
   compound_representation_item,
   product_definition_relationship,
   group,
   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,
   celsius_temperature_measure,
   amount_of_substance_measure,
   luminous_intensity_measure,
   plane_angle_measure,
   solid_angle_measure,
   area_measure,
   volume_measure,
   ratio_measure,
   parameter_value,
```

```
    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_request_solution,
   document,
   executed_action,
   versioned_action_request,
   action);
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
  (curve_3d,
   pcurve_s1,
   pcurve_s2);
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_assignment,
   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; -- shell

  TYPE si_prefix = ENUMERATION OF
    (exa,
     peta,
     tera,
     giga,
     mega,
     kilo,
     hecto,
     deca,
     deci,
     centi,
     milli,
     micro,
     nano,
     pico,
     femto,
     atto);
  END_TYPE; -- si_prefix

  TYPE si_unit_name = ENUMERATION OF
    (metre,
     gram,
     second,
     ampere,
     kelvin,
     mole,
     candela,
```



```
    radian,
    steradian,
    hertz,
    newton,
    pascal,
    joule,
    watt,
    coulomb,
    volt,
    farad,
    ohm,
    siemens,
    weber,
    tesla,
    henry,
    degree_celsius,
    lumen,
    lux,
    becquerel,
    gray,
    sievert);
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 supported_item = SELECT
  (action,
   action_method);
END_TYPE; -- supported_item

TYPE surface_boundary = SELECT
  (degenerate_pcurve);
END_TYPE; -- surface_boundary

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,
     continuous,
     cont_same_gradient,
     cont_same_gradient_same_curvature);
  END_TYPE; -- transition_code

  TYPE trimming_select = SELECT
    (cartesian_point,
     parameter_value);
  END_TYPE; -- trimming_select

  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 wireframe_model = SELECT
    (edge_based_wireframe_model);
  END_TYPE; -- wireframe_model

  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_MOULDED_FORM_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_MOULDED_FORM_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_relationship;
    name            : label;
    description     : OPTIONAL text;
    relating_action : action;
    related_action  : action;
END_ENTITY; -- action_relationship

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_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.' +
              'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1);
    wr2: (SIZEOF(USEDIN(SELF,'SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.ELEMENTARY_SURFACE',
                'SHIP_MOULDED_FORM_SCHEMA.B_SPLINE_SURFACE',
                'SHIP_MOULDED_FORM_SCHEMA.SWEPT_SURFACE'] * TYPEOF(
                face_geometry)) = 1);
      wr2 : (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_LOOP' IN
TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list | (NOT
('SHIP_MOULDED_FORM_SCHEMA.EDGE_CURVE' IN
                TYPEOF(oe\oriented_edge.edge_element))) )) = 0)) ))
= 0);
      wr3 : (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_LOOP' IN
TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list | (NOT
(SIZEOF(['SHIP_MOULDED_FORM_SCHEMA.LINE',
                'SHIP_MOULDED_FORM_SCHEMA.CONIC',
                'SHIP_MOULDED_FORM_SCHEMA.POLYLINE',
                'SHIP_MOULDED_FORM_SCHEMA.SURFACE_CURVE',
                'SHIP_MOULDED_FORM_SCHEMA.B_SPLINE_CURVE'] *
TYPEOF(oe.
                edge_element\edge_curve.edge_geometry)) = 1)) )) =
0)) )) =
                 0);
      wr4 : (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_LOOP' IN
TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list | (NOT
(('SHIP_MOULDED_FORM_SCHEMA.VERTEX_POINT'
                IN TYPEOF(oe\edge.edge_start)) AND (
                'SHIP_MOULDED_FORM_SCHEMA.CARTESIAN_POINT' IN
TYPEOF(oe\
                edge.edge_start\vertex_point.vertex_geometry)) AND (
                'SHIP_MOULDED_FORM_SCHEMA.VERTEX_POINT' IN
```



```
TYPEOF(oe\edge.
                edge_end)) AND
('SHIP_MOULDED_FORM_SCHEMA.CARTESIAN_POINT'
                IN
TYPEOF(oe\edge.edge_end\vertex_point.vertex_geometry)))) ))
                = 0)) )) = 0);
      wr5 : (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_LOOP' IN
TYPEOF(bnds.bound)) )
                 | ('SHIP_MOULDED_FORM_SCHEMA.ORIENTED_PATH' IN
TYPEOF(
                elp_fbnds.bound)) )) = 0);
      wr6 : ((NOT ('SHIP_MOULDED_FORM_SCHEMA.SWEPT_SURFACE' IN
TYPEOF(
                face_geometry))) OR (SIZEOF([
                'SHIP_MOULDED_FORM_SCHEMA.LINE',
                'SHIP_MOULDED_FORM_SCHEMA.CONIC',
                'SHIP_MOULDED_FORM_SCHEMA.POLYLINE',
                'SHIP_MOULDED_FORM_SCHEMA.B_SPLINE_CURVE'] * TYPEOF(
                face_geometry\swept_surface.swept_curve)) = 1));
      wr7 : (SIZEOF(QUERY ( vlp_fbnds <* QUERY ( bnds <* bounds | (
                'SHIP_MOULDED_FORM_SCHEMA.VERTEX_LOOP' IN
TYPEOF(bnds.bound)) )
                 | (NOT (('SHIP_MOULDED_FORM_SCHEMA.VERTEX_POINT' IN

TYPEOF(vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex))
                AND ('SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.EDGE_LOOP',
                'SHIP_MOULDED_FORM_SCHEMA.VERTEX_LOOP'] *
TYPEOF(bnd.bound))
                = 1)) )) = 0);
      wr9 : (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_LOOP' IN
TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list |
(('SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.PCURVE' IN TYPEOF(sc_ag)))
)) = 0))) ))
                = 0)) )) = 0);
      wr10: (((NOT ('SHIP_MOULDED_FORM_SCHEMA.SWEPT_SURFACE' IN
TYPEOF(
                face_geometry))) OR (NOT (
                'SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.EDGE_LOOP' IN
TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list | (('SHIP_MOULDED_FORM_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 amount_of_substance_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 = 1) AND
(SELF\
               named_unit.dimensions.luminous_intensity_exponent =
0));
  END_ENTITY; -- amount_of_substance_unit

  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_MOULDED_FORM_SCHEMA.' +
                'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1);
      wr2: (SIZEOF(USEDIN(SELF,'SHIP_MOULDED_FORM_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;
    WHERE
      wr1: ((NOT (SELF\approval_assignment.role.name =
                'proposed alternative')) OR (SIZEOF(QUERY ( app <*
USEDIN(

SELF\approval_assignment.assigned_approval,'SHIP_MOULDED_FORM_SCHEMA
.APPROVAL_ASSIGNMENT.ASSIGNED_APPROVAL')
                 |
(('SHIP_MOULDED_FORM_SCHEMA.APPLIED_APPROVAL_ASSIGNMENT'
               IN TYPEOF(app)) AND
(app\approval_assignment.role.name =
                'subject')) )) = 1));
  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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.UNIFORM_CURVE' IN
TYPEOF(SELF)) OR
               ('SHIP_MOULDED_FORM_SCHEMA.QUASI_UNIFORM_CURVE' IN
TYPEOF(
               SELF)) OR ('SHIP_MOULDED_FORM_SCHEMA.BEZIER_CURVE' IN
               TYPEOF(SELF)) OR (
               'SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.UNIFORM_SURFACE' IN
TYPEOF(SELF))
               OR ('SHIP_MOULDED_FORM_SCHEMA.QUASI_UNIFORM_SURFACE'
IN
               TYPEOF(SELF)) OR
('SHIP_MOULDED_FORM_SCHEMA.BEZIER_SURFACE'
               IN TYPEOF(SELF)) OR (

'SHIP_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.GROUP_ASSIGNMENT.ASSIGNED_GROUP')
                | (NOT
('SHIP_MOULDED_FORM_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_MOULDED_FORM_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_edge_set
    SUBTYPE OF (topological_representation_item);
      ces_edges : SET [1:?] OF edge;
  END_ENTITY; -- connected_edge_set
```



```
  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 conversion_based_unit
    SUBTYPE OF (named_unit);
      name              : label;
      conversion_factor : measure_with_unit;
  END_ENTITY; -- conversion_based_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_MOULDED_FORM_SCHEMA.' +
                'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1);
  END_ENTITY; -- date_time_role

  ENTITY definitional_representation
    SUBTYPE OF (representation);
    WHERE
      wr1:
('SHIP_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_based_wireframe_model
    SUBTYPE OF (geometric_representation_item);
      ebwm_boundary : SET [1:?] OF connected_edge_set;
  END_ENTITY; -- edge_based_wireframe_model

  ENTITY edge_based_wireframe_shape_representation
    SUBTYPE OF (shape_representation);
    WHERE
      wr1: (SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([

'SHIP_MOULDED_FORM_SCHEMA.EDGE_BASED_WIREFRAME_MODEL',
                'SHIP_MOULDED_FORM_SCHEMA.MAPPED_ITEM',
                'SHIP_MOULDED_FORM_SCHEMA.AXIS2_PLACEMENT_3D'] *
TYPEOF(it))
                = 1)) )) = 0);
      wr2: (SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([

'SHIP_MOULDED_FORM_SCHEMA.EDGE_BASED_WIREFRAME_MODEL',
                'SHIP_MOULDED_FORM_SCHEMA.MAPPED_ITEM'] * TYPEOF(it))
= 1) ))
                >= 1);
      wr3: (SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_BASED_WIREFRAME_MODEL'
IN
                TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( eb <* ebwm\
```



```
                edge_based_wireframe_model.ebwm_boundary | (NOT
(SIZEOF(
                QUERY ( edges <* eb.ces_edges | (NOT (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_CURVE' IN
TYPEOF(edges))) ))
                = 0)) )) = 0)) )) = 0);
      wr4: (SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_BASED_WIREFRAME_MODEL'
IN
                TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( eb <* ebwm\
                edge_based_wireframe_model.ebwm_boundary | (NOT
(SIZEOF(
                QUERY ( pline_edges <* QUERY ( edges <* eb.ces_edges
| (
                'SHIP_MOULDED_FORM_SCHEMA.POLYLINE' IN TYPEOF(edges\
                edge_curve.edge_geometry)) ) | (NOT
(SIZEOF(pline_edges\
                edge_curve.edge_geometry\polyline.points) > 2)) )) =
0)) ))
                = 0)) )) = 0);
      wr5: (SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_BASED_WIREFRAME_MODEL'
IN
                TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( eb <* ebwm\
                edge_based_wireframe_model.ebwm_boundary | (NOT
(SIZEOF(
                QUERY ( edges <* eb.ces_edges | (NOT ((
                'SHIP_MOULDED_FORM_SCHEMA.VERTEX_POINT' IN
TYPEOF(edges.
                edge_start)) AND
('SHIP_MOULDED_FORM_SCHEMA.VERTEX_POINT' IN
                 TYPEOF(edges.edge_end)))) )) = 0)) )) = 0)) )) = 0);
      wr6: (SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_BASED_WIREFRAME_MODEL'
IN
                TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( eb <* ebwm\
                edge_based_wireframe_model.ebwm_boundary | (NOT
(SIZEOF(
                QUERY ( edges <* eb.ces_edges | (NOT

valid_wireframe_edge_curve(edges\edge_curve.edge_geometry)) ))
                = 0)) )) = 0)) )) = 0);
      wr7: (SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_BASED_WIREFRAME_MODEL'
IN
                TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( eb <* ebwm\
                edge_based_wireframe_model.ebwm_boundary | (NOT
(SIZEOF(
                QUERY ( edges <* eb.ces_edges | (NOT (

valid_wireframe_vertex_point(edges.edge_start\vertex_point.
                vertex_geometry) AND
valid_wireframe_vertex_point(edges.
                edge_end\vertex_point.vertex_geometry))) )) = 0)) ))
= 0)) ))
```



```
                = 0);
      wr8: (SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_SCHEMA.MAPPED_ITEM' IN TYPEOF(it))
) | (
                NOT (('SHIP_MOULDED_FORM_SCHEMA.' +
                'EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION') IN
TYPEOF(mi\
                mapped_item.mapping_source.mapped_representation)))
)) = 0);
      wr9: (SELF.context_of_items\geometric_representation_context.
                coordinate_space_dimension = 3);
  END_ENTITY; -- edge_based_wireframe_shape_representation

  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_MOULDED_FORM_SCHEMA.' +
                'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1);
      wr2: (SIZEOF(USEDIN(SELF,'SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.' +
                'ROLE_ASSOCIATION.ITEM_WITH_ROLE')) <= 1);
  END_ENTITY; -- effectivity_assignment

  ENTITY electric_current_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 = 1)
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; -- electric_current_unit

  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_MOULDED_FORM_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 externally_defined_item;
      item_id : source_item;
      source  : external_source;
  END_ENTITY; -- externally_defined_item

  ENTITY face
    SUPERTYPE OF (ONEOF (face_surface,subface,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_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.ORIENTED_SURFACE' IN
TYPEOF(
                face_geometry)));
  END_ENTITY; -- face_surface

  ENTITY faceted_brep
    SUBTYPE OF (manifold_solid_brep);
  END_ENTITY; -- faceted_brep

  ENTITY founded_item;
  END_ENTITY; -- founded_item

  ENTITY functionally_defined_transformation;
```



```
      name        : label;
      description : OPTIONAL text;
  END_ENTITY; -- functionally_defined_transformation

  ENTITY geometric_curve_set
    SUBTYPE OF (geometric_set);
    WHERE
      wr1: (SIZEOF(QUERY ( temp <* SELF\geometric_set.elements | (
                'SHIP_MOULDED_FORM_SCHEMA.SURFACE' IN TYPEOF(temp))
)) = 0);
  END_ENTITY; -- geometric_curve_set

  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,solid_model,

face_based_surface_model,edge_based_wireframe_model,geometric_set))
    SUBTYPE OF (representation_item);
    DERIVE
      dim : dimension_count := dimension_of(SELF);
    WHERE
      wr1: (SIZEOF(QUERY ( using_rep <* using_representations(SELF)
| (
               NOT (

'SHIP_MOULDED_FORM_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT'
               IN TYPEOF(using_rep.context_of_items))) )) = 0);
  END_ENTITY; -- geometric_representation_item

  ENTITY geometric_set
    SUPERTYPE OF (geometric_curve_set)
    SUBTYPE OF (geometric_representation_item);
      elements : SET [1:?] OF geometric_set_select;
  END_ENTITY; -- geometric_set

  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_MOULDED_FORM_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_MOULDED_FORM_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_measure_with_unit
    SUBTYPE OF (measure_with_unit);
    WHERE
      wr1: ('SHIP_MOULDED_FORM_SCHEMA.LENGTH_UNIT' IN TYPEOF(SELF\
               measure_with_unit.unit_component));
  END_ENTITY; -- length_measure_with_unit

  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 manifold_solid_brep
    SUBTYPE OF (solid_model);
      outer : closed_shell;
  END_ENTITY; -- manifold_solid_brep

  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_measure_with_unit
    SUBTYPE OF (measure_with_unit);
    WHERE
      wr1: ('SHIP_MOULDED_FORM_SCHEMA.MASS_UNIT' IN TYPEOF(SELF\
                measure_with_unit.unit_component));
  END_ENTITY; -- mass_measure_with_unit

  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
    SUPERTYPE OF (ONEOF
(length_measure_with_unit,mass_measure_with_unit,

plane_angle_measure_with_unit,solid_angle_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,conversion_based_unit,
        context_dependent_unit) ANDOR ONEOF (length_unit,mass_unit,

time_unit,electric_current_unit,thermodynamic_temperature_unit,

amount_of_substance_unit,luminous_intensity_unit,plane_angle_unit,
        solid_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_MOULDED_FORM_SCHEMA.FACE_BASED_SURFACE_MODEL',
                'SHIP_MOULDED_FORM_SCHEMA.MAPPED_ITEM',
                'SHIP_MOULDED_FORM_SCHEMA.AXIS2_PLACEMENT_3D'] *
TYPEOF(it))
                = 1)) )) = 0);
      wr2 : (SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
                'SHIP_MOULDED_FORM_SCHEMA.FACE_BASED_SURFACE_MODEL',
                'SHIP_MOULDED_FORM_SCHEMA.MAPPED_ITEM'] *
TYPEOF(it)) = 1) ))
                > 0);
      wr3 : (SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_SCHEMA.MAPPED_ITEM' IN
TYPEOF(it)) ) | (
```

```
                NOT ((('SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.FACE_BASED_SURFACE_MODEL'
IN
                TYPEOF(mr_it)) )) > 0))) )) = 0);
      wr4 : (SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.FACE_SURFACE',
                'SHIP_MOULDED_FORM_SCHEMA.ORIENTED_FACE'] *
TYPEOF(fa)) = 1)) ))
                = 0)) )) = 0)) )) = 0);
      wr5 : (SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.FACE_SURFACE' IN
TYPEOF(fa)) ) |
                (NOT (('SHIP_MOULDED_FORM_SCHEMA.ADVANCED_FACE' IN
TYPEOF(
                f_sf)) OR nmsf_surface_check(f_sf\face_surface.
                face_geometry))) )) = 0)) )) = 0)) )) = 0);
      wr6 : (SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_SCHEMA.FACE_BASED_SURFACE_MODEL'
IN
                TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* fbsm\
                face_based_surface_model.fbsm_faces | (NOT (SIZEOF(
                QUERY ( o_fa <* QUERY ( fa <* cfs.cfs_faces | (
                'SHIP_MOULDED_FORM_SCHEMA.ORIENTED_FACE' IN
TYPEOF(fa)) )
                 | (NOT (('SHIP_MOULDED_FORM_SCHEMA.ADVANCED_FACE'
IN
                TYPEOF(o_fa\oriented_face.face_element)) OR
                nmsf_surface_check(o_fa\oriented_face.face_element\
                face_surface.face_geometry))) )) = 0)) )) = 0)) )) =
0);
      wr7 : (SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_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 ((
                'SHIP_MOULDED_FORM_SCHEMA.ADVANCED_FACE' IN
TYPEOF(fa)) OR
```



```
                (SIZEOF(QUERY ( bnds <* fa.bounds | (NOT (SIZEOF([
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_LOOP',
                'SHIP_MOULDED_FORM_SCHEMA.VERTEX_LOOP'] *
TYPEOF(bnds.bound))
                = 1)) )) = 0))) )) = 0)) )) = 0)) )) = 0);
      wr8 : (SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_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 ((
                'SHIP_MOULDED_FORM_SCHEMA.ADVANCED_FACE' IN
TYPEOF(fa)) OR
                (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <*
fa.bounds | (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_LOOP' IN
TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <*
elp_fbnds\path.edge_list | (
                NOT ('SHIP_MOULDED_FORM_SCHEMA.EDGE_CURVE' IN
TYPEOF(oe.
                edge_element))) )) = 0)) )) = 0))) )) = 0)) )) = 0))
)) = 0);
      wr9 : (SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_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 ((
                'SHIP_MOULDED_FORM_SCHEMA.ADVANCED_FACE' IN
TYPEOF(fa)) OR
                (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <*
fa.bounds | (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_LOOP' IN
TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe_cv <* QUERY ( oe <*
elp_fbnds\
                path.edge_list |
('SHIP_MOULDED_FORM_SCHEMA.EDGE_CURVE' IN
                TYPEOF(oe.edge_element)) ) | (NOT (SIZEOF([
                'SHIP_MOULDED_FORM_SCHEMA.B_SPLINE_CURVE',
                'SHIP_MOULDED_FORM_SCHEMA.CONIC',
                'SHIP_MOULDED_FORM_SCHEMA.CURVE_REPLICA',
                'SHIP_MOULDED_FORM_SCHEMA.LINE',
                'SHIP_MOULDED_FORM_SCHEMA.OFFSET_CURVE_3D',
                'SHIP_MOULDED_FORM_SCHEMA.PCURVE',
                'SHIP_MOULDED_FORM_SCHEMA.POLYLINE',
                'SHIP_MOULDED_FORM_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_MOULDED_FORM_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 ((
                'SHIP_MOULDED_FORM_SCHEMA.ADVANCED_FACE' IN
TYPEOF(fa)) OR
                (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <*
fa.bounds | (
                'SHIP_MOULDED_FORM_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_MOULDED_FORM_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 ((
                'SHIP_MOULDED_FORM_SCHEMA.ADVANCED_FACE' IN
TYPEOF(fa)) OR
                (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <*
fa.bounds | (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_LOOP' IN
TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <*
elp_fbnds\path.edge_list | (
                NOT (('SHIP_MOULDED_FORM_SCHEMA.VERTEX_POINT' IN
TYPEOF(oe.
                edge_element.edge_start)) AND (
                'SHIP_MOULDED_FORM_SCHEMA.VERTEX_POINT' IN
TYPEOF(oe.
                edge_element.edge_end)))) )) = 0)) )) = 0))) )) =
0)) )) =
                0)) )) = 0);
      wr12: (SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_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 ((
                'SHIP_MOULDED_FORM_SCHEMA.ADVANCED_FACE' IN
TYPEOF(fa)) OR
                (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <*
fa.bounds | (
                'SHIP_MOULDED_FORM_SCHEMA.EDGE_LOOP' IN
TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <*
elp_fbnds\path.edge_list | (
                NOT
((SIZEOF(['SHIP_MOULDED_FORM_SCHEMA.CARTESIAN_POINT',
                'SHIP_MOULDED_FORM_SCHEMA.DEGENERATE_PCURVE',
                'SHIP_MOULDED_FORM_SCHEMA.POINT_ON_CURVE',
                'SHIP_MOULDED_FORM_SCHEMA.POINT_ON_SURFACE'] *
```



```
TYPEOF(oe.

edge_element.edge_start\vertex_point.vertex_geometry)) = 1)
                AND
(SIZEOF(['SHIP_MOULDED_FORM_SCHEMA.CARTESIAN_POINT',
                'SHIP_MOULDED_FORM_SCHEMA.DEGENERATE_PCURVE',
                'SHIP_MOULDED_FORM_SCHEMA.POINT_ON_CURVE',
                'SHIP_MOULDED_FORM_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_MOULDED_FORM_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 ((
                'SHIP_MOULDED_FORM_SCHEMA.ADVANCED_FACE' IN
TYPEOF(fa)) OR
                (SIZEOF(QUERY ( vlp_fbnds <* QUERY ( bnds <*
fa.bounds | (
                'SHIP_MOULDED_FORM_SCHEMA.VERTEX_LOOP' IN
TYPEOF(bnds.bound)) )
                 | (NOT ('SHIP_MOULDED_FORM_SCHEMA.VERTEX_POINT' IN
TYPEOF(
                vlp_fbnds\vertex_loop.loop_vertex))) )) = 0))) )) =
0)) ))
                = 0)) )) = 0);
      wr14: (SIZEOF(QUERY ( fbsm <* QUERY ( it <* SELF.items | (
                'SHIP_MOULDED_FORM_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 ((
                'SHIP_MOULDED_FORM_SCHEMA.ADVANCED_FACE' IN
TYPEOF(fa)) OR
                (SIZEOF(QUERY ( vlp_fbnds <* QUERY ( bnds <*
fa.bounds | (
                'SHIP_MOULDED_FORM_SCHEMA.VERTEX_LOOP' IN
TYPEOF(bnds.bound)) )
                 | (NOT
(SIZEOF(['SHIP_MOULDED_FORM_SCHEMA.CARTESIAN_POINT',
                'SHIP_MOULDED_FORM_SCHEMA.DEGENERATE_PCURVE',
                'SHIP_MOULDED_FORM_SCHEMA.POINT_ON_CURVE',
                'SHIP_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.' +
              'NAME_ATTRIBUTE.NAMED_ITEM')) <= 1);
      wr2: (SIZEOF(USEDIN(SELF,'SHIP_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_measure_with_unit
    SUBTYPE OF (measure_with_unit);
    WHERE
      wr1: ('SHIP_MOULDED_FORM_SCHEMA.PLANE_ANGLE_UNIT' IN
TYPEOF(SELF\
                measure_with_unit.unit_component));
  END_ENTITY; -- plane_angle_measure_with_unit

  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,
        point_replica,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 point_replica
    SUBTYPE OF (point);
      parent_pt      : point;
      transformation : cartesian_transformation_operator;
    WHERE
      wr1: (transformation.dim = parent_pt.dim);
      wr2: acyclic_point_replica(SELF,parent_pt);
  END_ENTITY; -- point_replica

  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_MOULDED_FORM_SCHEMA.' +
                'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1);
  END_ENTITY; -- product_category

  ENTITY product_category_relationship;
      name         : label;
      description  : OPTIONAL text;
      category     : product_category;
      sub_category : product_category;
    WHERE
      wr1:
acyclic_product_category_relationship(SELF,[SELF.sub_category]);
  END_ENTITY; -- product_category_relationship

  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_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.CHARACTERIZED_PRODUCT_DEFINITION',
               'SHIP_MOULDED_FORM_SCHEMA.CHARACTERIZED_OBJECT'] *
TYPEOF(
              SELF\property_definition.definition)) > 0);
  END_ENTITY; -- product_definition_shape

  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_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.' +
               'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1);
      wr2: (SIZEOF(USEDIN(SELF,'SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.' +
              'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1);
      wr2: (SIZEOF(USEDIN(SELF,'SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.PCURVE' IN
TYPEOF(SELF\surface_curve
                .associated_geometry[1]));
      wr4: ('SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.' +
                'ID_ATTRIBUTE.IDENTIFIED_ITEM')) <= 1);
  END_ENTITY; -- shape_aspect

  ENTITY shape_definition_representation
    SUBTYPE OF (property_definition_representation);
    WHERE
      wr1: (('SHIP_MOULDED_FORM_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN
                TYPEOF(SELF.definition)) OR (
                'SHIP_MOULDED_FORM_SCHEMA.SHAPE_DEFINITION' IN
TYPEOF(SELF.
                definition.definition)));
      wr2: ('SHIP_MOULDED_FORM_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 solid_angle_measure_with_unit
    SUBTYPE OF (measure_with_unit);
    WHERE
      wr1: ('SHIP_MOULDED_FORM_SCHEMA.SOLID_ANGLE_UNIT' IN
TYPEOF(SELF\
               measure_with_unit.unit_component));
  END_ENTITY; -- solid_angle_measure_with_unit

  ENTITY solid_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; -- solid_angle_unit

  ENTITY solid_model
    SUPERTYPE OF (manifold_solid_brep)
    SUBTYPE OF (geometric_representation_item);
  END_ENTITY; -- solid_model

  ENTITY spherical_surface
    SUBTYPE OF (elementary_surface);
      radius : positive_length_measure;
  END_ENTITY; -- spherical_surface

  ENTITY subface
    SUBTYPE OF (face);
      parent_face : face;
    WHERE
      wr1: (NOT
mixed_loop_type_set(list_to_set(list_face_loops(SELF)) +
               list_to_set(list_face_loops(parent_face))));
  END_ENTITY; -- subface

  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_MOULDED_FORM_SCHEMA.PCURVE' IN TYPEOF(
                associated_geometry[1])) OR (master_representation <>
                pcurve_s1));
      wr3: (('SHIP_MOULDED_FORM_SCHEMA.PCURVE' IN TYPEOF(
                associated_geometry[2])) OR (master_representation <>
                pcurve_s2));
      wr4: (NOT ('SHIP_MOULDED_FORM_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 := representation_item('') ||
                     geometric_representation_item() || curve() ||
line(
                     axis_position.location,representation_item('')
||
                     geometric_representation_item() || 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_edge_set,
        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_MOULDED_FORM_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
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action_request_solution := [];
      set_3   : SET OF action_method := [];
      t2_set  : SET OF action := [];
    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) BY 1 WHILE NOT violate;
      set_3 := [];
      REPEAT j := 1 TO HIINDEX(t2_set) BY 1;
        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; -- action_request_solution_connected_to_action

  RULE action_request_solution_with_identification_assignment FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF action_request_solution := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['change plan'];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.APPLIED_IDENTI
FICATION_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; --
action_request_solution_with_identification_assignment

  RULE action_with_identification_assignment FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF action := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['change',
                   'versionable object change event','check'];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.APPLIED_IDENTI
FICATION_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; -- action_with_identification_assignment

  RULE alternative_version_relationship_has_mandatory_description
FOR (
            identification_assignment_relationship);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF identification_assignment_relationship := [];
    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; --
alternative_version_relationship_has_mandatory_description

  RULE alternative_version_relationship_has_unique_versions FOR (
            identification_assignment_relationship);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF identification_assignment_relationship := [];
    END_LOCAL;
    t1_set := QUERY ( a <* identification_assignment_relationship |
        VALUE_IN(which_class(a),'alternative version relationship')
);
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- alternative_version_relationship_has_unique_versions

  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_MOULDED_FO
RM_SCHEMA.IDENTIFICATION_ASSIGNMENT_RELATIONSHIP.'
          + 'RELATING_IDENTIFICATION_ASSIGNMENT')) > 0) OR
(SIZEOF(USEDIN(

applied_identification_assignment[i],'SHIP_MOULDED_FORM_SCHEMA.IDENT
IFICATION_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; -- alternative_version_relationship_versionable_object

  RULE approval_event_with_approval_date_time FOR (approval);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF approval := [];
      t2_set  : SET OF approval_date_time := [];
    END_LOCAL;
    t1_set := QUERY ( i <* approval | VALUE_IN(which_class(i),
        'approval event') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violate;
      t2_set := bag_to_set(USEDIN(t1_set[i],
          'SHIP_MOULDED_FORM_SCHEMA.APPROVAL_DATE_TIME.' +
'DATED_APPROVAL'));
      violate := NOT (SIZEOF(t2_set) = 1);
    END_REPEAT;

  WHERE
    wr1: (NOT violate);

  END_RULE; -- approval_event_with_approval_date_time

  RULE approval_event_with_approval_person_organization FOR
(approval);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF approval := [];
      t2_set  : SET OF approval_person_organization := [];
    END_LOCAL;
    t1_set := QUERY ( i <* approval | VALUE_IN(which_class(i),
        'approval event') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violate;
      t2_set := bag_to_set(USEDIN(t1_set[i],
          'SHIP_MOULDED_FORM_SCHEMA.APPROVAL_PERSON_ORGANIZATION.' +
          'AUTHORIZED_APPROVAL'));
      violate := NOT (SIZEOF(t2_set) = 1);
    END_REPEAT;

  WHERE
    wr1: (NOT violate);

  END_RULE; -- approval_event_with_approval_person_organization

  RULE approval_history_approves_same_definition FOR (
            applied_group_assignment, applied_approval_assignment);

    LOCAL
      violate : LOGICAL := FALSE;
      t3_set  : SET OF approval := [];
      t4_set  : SET OF group_item := [];
```



```
      t5_set  : SET OF applied_approval_assignment := [];
      t2_set  : SET OF applied_group_assignment := [];
    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_MOULDED_FORM_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) BY 1 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; -- approval_history_approves_same_definition

  RULE approval_history_has_at_least_one_member FOR (group,
            applied_group_assignment);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF group := [];
      t2_set  : SET OF applied_group_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( i <* group | VALUE_IN(which_class(i),
        'approval history') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- approval_history_has_at_least_one_member

  RULE approvals_references_approval_history FOR
(applied_group_assignment,
            group);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF group := [];
      a_set   : SET OF applied_group_assignment := [];
    END_LOCAL;
```

```
    t1_set := QUERY ( a <* group | VALUE_IN(which_class(a),
        'approval history') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- approvals_references_approval_history

  RULE author_for_change_plan FOR (
            applied_person_and_organization_assignment,
            action_request_solution);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action_request_solution := [];
      a_set   : SET OF applied_person_and_organization_assignment :=
[];
    END_LOCAL;
    t1_set := QUERY ( a <* action_request_solution |
VALUE_IN(which_class(
        a),'change plan') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- author_for_change_plan

  RULE author_for_change_realisation FOR (
            applied_person_and_organization_assignment,
executed_action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF executed_action := [];
      a_set   : SET OF applied_person_and_organization_assignment :=
[];
    END_LOCAL;
    t1_set := QUERY ( a <* executed_action |
VALUE_IN(which_class(a),
        'change realization') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- author_for_change_realisation

  RULE author_for_change_request FOR (
            applied_person_and_organization_assignment,
            versioned_action_request);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF versioned_action_request := [];
      a_set   : SET OF applied_person_and_organization_assignment :=
[];
    END_LOCAL;
    t1_set := QUERY ( a <* versioned_action_request | VALUE_IN(
        which_class(a),'change request') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- author_for_change_request

  RULE caused_by_for_check FOR
(applied_person_and_organization_assignment,
            action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action := [];
      a_set   : SET OF applied_person_and_organization_assignment :=
[];
    END_LOCAL;
    t1_set := QUERY ( a <* action | VALUE_IN(which_class(a),'check')
);
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- caused_by_for_check

  RULE caused_by_for_envisaged_version_creation FOR (
            applied_person_and_organization_assignment, action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action := [];
      a_set   : SET OF applied_person_and_organization_assignment :=
[];
    END_LOCAL;
    t1_set := QUERY ( a <* action | VALUE_IN(which_class(a),
        'envisaged version creation') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- caused_by_for_envisaged_version_creation

  RULE caused_by_for_version_creation FOR (
            applied_person_and_organization_assignment, action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action := [];
      a_set   : SET OF applied_person_and_organization_assignment :=
[];
    END_LOCAL;
    t1_set := QUERY ( a <* action | VALUE_IN(which_class(a),
        'version creation') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- caused_by_for_version_creation

  RULE caused_by_for_version_deletion FOR (
            applied_person_and_organization_assignment, action);
```



```
    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action := [];
      a_set   : SET OF applied_person_and_organization_assignment :=
[];
    END_LOCAL;
    t1_set := QUERY ( a <* action | VALUE_IN(which_class(a),
        'version deletion') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- caused_by_for_version_deletion

  RULE caused_by_for_version_modification FOR (
            applied_person_and_organization_assignment, action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action := [];
      a_set   : SET OF applied_person_and_organization_assignment :=
[];
    END_LOCAL;
    t1_set := QUERY ( a <* action | VALUE_IN(which_class(a),
        'version modification') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- caused_by_for_version_modification

  RULE caused_when_for_check FOR (applied_date_and_time_assignment,
action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action := [];
      a_set   : SET OF applied_date_and_time_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* action | VALUE_IN(which_class(a),'check')
);
```

```
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- caused_when_for_check

  RULE caused_when_for_envisaged_version_creation FOR (
            applied_date_and_time_assignment, action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action := [];
      a_set   : SET OF applied_date_and_time_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* action | VALUE_IN(which_class(a),
        'envisaged version creation') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- caused_when_for_envisaged_version_creation

  RULE caused_when_for_version_creation FOR (
            applied_date_and_time_assignment, action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action := [];
      a_set   : SET OF applied_date_and_time_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* action | VALUE_IN(which_class(a),
        'version creation') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- caused_when_for_version_creation
```



```
  RULE caused_when_for_version_deletion FOR (
            applied_date_and_time_assignment, action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action := [];
      a_set   : SET OF applied_date_and_time_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* action | VALUE_IN(which_class(a),
        'version deletion') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- caused_when_for_version_deletion

  RULE caused_when_for_version_modification FOR (
            applied_date_and_time_assignment, action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action := [];
      a_set   : SET OF applied_date_and_time_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* action | VALUE_IN(which_class(a),
        'version modification') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- caused_when_for_version_modification

  RULE centre_location_compound_representation_has_specified_name
FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['longitudinal location',
                   'transversal location','vertical location'];
      t2_set    : SET OF representation_item := [];
```

```
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'centre location') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
centre_location_compound_representation_has_specified_name

  RULE change_impact_with_versionable_object_change_event FOR (
            applied_action_request_assignment);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF applied_action_request_assignment := [];
      a_set   : SET OF action := [];
    END_LOCAL;
    t1_set := QUERY ( b <* applied_action_request_assignment |
(b.role.
        name = 'change impact') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violate;
      a_set := QUERY ( b <* t1_set[i].items | ((
          'SHIP_MOULDED_FORM_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; -- change_impact_with_versionable_object_change_event

  RULE change_plan_has_mandatory_attribute_description FOR (
            action_request_solution);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action_request_solution := [];
    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; -- change_plan_has_mandatory_attribute_description

  RULE class_and_statutory_designation_has_properties FOR (
            property_definition_representation,
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t4_list   : LIST OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t3_list   : LIST OF property_definition := [];
      t1_list   : LIST OF product_definition := [];
      t2_set    : SET OF property_definition_representation := [];
    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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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) BY 1;
      t3_list := t3_list + t2_set[i].definition;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t3_list) BY 1;
      t4_list := t4_list + t3_list[i].definition;
    END_REPEAT;
    violation := t1_list <> t4_list;

  WHERE
    wr1: (NOT violation);

  END_RULE; -- class_and_statutory_designation_has_properties

  RULE class_notation_with_named_representation_items FOR
(representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['class notations hull',
                   'class notations machinery'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
```

```
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'class notation') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- class_notation_with_named_representation_items

  RULE class_parameters_has_properties FOR (
            property_definition_representation,
            applied_classification_assignment);

    LOCAL
      t3_set    : LIST OF property_definition := [];
      violation : LOGICAL := FALSE;
      t4_set    : LIST OF product_definition := [];
      t1_set    : LIST OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'class parameters') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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) BY 1;
      t3_set := t3_set + t2_set[i].definition;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t3_set) BY 1;
      t4_set := t4_set + t3_set[i].definition;
    END_REPEAT;
    violation := t1_set <> t4_set;

  WHERE
    wr1: (NOT violation);

  END_RULE; -- class_parameters_has_properties

  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; -- compatible_dimension

  RULE compound_representation_item_with_class_id_knot FOR (
              applied_classification_assignment);

    LOCAL
      t3_set     : SET OF representation_item := [];
      violation  : LOGICAL := FALSE;
      t1_set     : SET OF compound_representation_item := [];
      c_a_set    : SET OF applied_classification_assignment := [];
      c_a_set2   : SET OF applied_classification_assignment := [];
      l_rep_item : list_representation_item;
      t2_set     : SET OF compound_representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'knot') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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 = 'ship curve') );
    REPEAT i := 1 TO HIINDEX(c_a_set2) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set2[i].items) BY 1;
        t2_set := t2_set + c_a_set2[i].items[j];
      END_REPEAT;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      l_rep_item := t1_set[i].item_element;
      REPEAT j := 1 TO HIINDEX(t1_set[i].item_element) BY 1;
        t3_set := t3_set + l_rep_item[j];
      END_REPEAT;
      violation := SIZEOF(t3_set * t2_set) < 2;
      t3_set := [];
    END_REPEAT;

  WHERE
    wr1: (NOT violation);
```

```
  END_RULE; -- compound_representation_item_with_class_id_knot

  RULE compound_representation_item_with_hydrostatic_properties FOR
(
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['hydrostatic property value'];
      t2_set    : LIST OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name =
        'hydrostatic properties for constant floating position') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
compound_representation_item_with_hydrostatic_properties

  RULE compound_representation_item_with_section_identifier FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'section of offset point table') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.' +
          'APPLIED_IDENTIFICATION_ASSIGNMENT' + '.ITEMS'));
```



```
      violation := NOT (SIZEOF(QUERY ( t2_inst <* t2_set |
(t2_inst.role.
          name = 'section identifier') )) = 1);
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; -- compound_representation_item_with_section_identifier

  RULE date_time_for_change_plan FOR
(applied_date_and_time_assignment,
            action_request_solution);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action_request_solution := [];
      a_set   : SET OF applied_date_and_time_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* action_request_solution |
VALUE_IN(which_class(
        a),'change plan') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- date_time_for_change_plan

  RULE date_time_for_change_realisation FOR (
            applied_date_and_time_assignment, executed_action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF executed_action := [];
      a_set   : SET OF applied_date_and_time_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* executed_action |
VALUE_IN(which_class(a),
        'change realisation') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- date_time_for_change_realisation

  RULE date_time_for_change_request FOR
(applied_date_and_time_assignment,
            versioned_action_request);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF versioned_action_request := [];
      a_set   : SET OF applied_date_and_time_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* versioned_action_request | VALUE_IN(
        which_class(a),'change request') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- date_time_for_change_request

  RULE document_has_at_least_one_references FOR (document);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF document := [];
      t2_set  : SET OF document_representation_type := [];
    END_LOCAL;
    t1_set := QUERY ( i <* document |
VALUE_IN(which_class(i),'document') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violate;
      t2_set := bag_to_set(USEDIN(t1_set[i],
          'SHIP_MOULDED_FORM_SCHEMA.DOCUMENT_REPRESENTATION_TYPE.' +
          'REPRESENTED_DOCUMENT'));
      violate := SIZEOF(t2_set) < 1;
    END_REPEAT;

  WHERE
    wr1: (NOT violate);

  END_RULE; -- document_has_at_least_one_references

  RULE document_has_exactly_one_author FOR (document);

    LOCAL
      violate : LOGICAL := FALSE;
      bag_3   : BAG OF applied_organization_assignment := [];
      bag_2   : BAG OF applied_person_and_organization_assignment :=
[];
      bag_1   : BAG OF applied_person_assignment := [];
    END_LOCAL;
```



```
    REPEAT i := 1 TO SIZEOF(document) BY 1 WHILE NOT violate;
      bag_1 := USEDIN(document[i],'SHIP_MOULDED_FORM_SCHEMA.' +
          'APPLIED_PERSON_ASSIGNMENT.ITEMS');
      bag_1 := QUERY ( assign <* bag_1 | (assign.role.name =
'author') );
      bag_2 := USEDIN(document[i],'SHIP_MOULDED_FORM_SCHEMA.' +
          'APPLIED_PERSON_AND_ORGANIZATION_ASSIGNMENT.ITEMS');
      bag_2 := QUERY ( assign <* bag_2 | (assign.role.name =
'author') );
      bag_3 := USEDIN(document[i],'SHIP_MOULDED_FORM_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; -- document_has_exactly_one_author

  RULE document_reference_with_address_has_at_least_one_references
FOR (
            document);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF document := [];
      t2_set  : SET OF applied_external_identification_assignment :=
[];
    END_LOCAL;
    t1_set := QUERY ( i <* document | VALUE_IN(which_class(i),
        'document reference with address') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violate;
      t2_set :=
bag_to_set(USEDIN(t1_set[i],'SHIP_MOULDED_FORM_SCHEMA.APPLIED_EXTERN
AL_IDENTIFICATION_ASSIGNMENT.ITEMS'));
      violate := SIZEOF(t2_set) < 1;
    END_REPEAT;

  WHERE
    wr1: (NOT violate);

  END_RULE; --
document_reference_with_address_has_at_least_one_references

  RULE
envisaged_version_creation_has_mandatory_attribute_description FOR (
            action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action := [];
    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; --
envisaged_version_creation_has_mandatory_attribute_description

  RULE executed_action_with_identification_assignment FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF executed_action := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['change realization'];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.APPLIED_IDENTI
FICATION_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; -- executed_action_with_identification_assignment

  RULE external_instance_reference_has_same_identifier FOR (
            applied_external_identification_assignment);

    LOCAL
      aia_set    : SET OF applied_identification_assignment := [];
      violation  : LOGICAL := FALSE;
      extref_set : SET OF applied_external_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_MOULDED_FORM_SCHEMA.APPLIED_IDEN
TIFICATION_ASSIGNMENT.ITEMS');
      violation := NOT (aia_set[1].assigned_id =
extref_set[i].assigned_id);
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; -- external_instance_reference_has_same_identifier

  RULE floating_position_compound_representation_with_name FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['moulded form displacement',
                   'draught at amidships','length of waterline',
                   'breadth of waterline','angle of trim','angle of
heel'];
      t2_set    : SET OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'floating position') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- floating_position_compound_representation_with_name

  RULE global_axis_placement_has_properties FOR (
            property_definition_representation, group,
            applied_classification_assignment);

    LOCAL
      t3_set    : LIST OF property_definition := [];
```

```
      violation : LOGICAL := FALSE;
      t4_set    : LIST OF product_definition := [];
      t1_set    : LIST OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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) BY 1;
      t3_set := t3_set + t2_set[i].definition;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t3_set) BY 1;
      t4_set := t4_set + t3_set[i].definition;
    END_REPEAT;
    violation := t1_set <> t4_set;

  WHERE
    wr1: (NOT violation);

  END_RULE; -- global_axis_placement_has_properties

  RULE global_id_is_unique FOR (applied_identification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      set_1     : SET OF applied_identification_assignment := [];
      bag_2     : BAG OF STRING := [];
    END_LOCAL;
    set_1 := QUERY ( i <* applied_identification_assignment |
(i.role.name
        = 'globally unambiguous identifier') );
    REPEAT i := 1 TO HIINDEX(set_1) BY 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; -- global_id_is_unique

  RULE hull_moulded_form_design_parameter_with_class_references FOR
(
            applied_classification_assignment);
```



```
    LOCAL
      t3_set    : SET OF representation := [];
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'hull moulded form design parameter')
);
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
          'DEFINITION'));
      REPEAT j := 1 TO HIINDEX(t2_set) BY 1;
        t3_set := t3_set + t2_set[j].used_representation;
      END_REPEAT;
      violation := SIZEOF(QUERY ( t2_inst <* t3_set | ('midship
tumble' IN
           which_class(t2_inst)) )) > 1;
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; --
hull_moulded_form_design_parameter_with_class_references

  RULE hydrostatic_properties_with_specified_class FOR (
            applied_classification_assignment);

    LOCAL
      t3_set     : SET OF representation_item := [];
      violation  : LOGICAL := FALSE;
      t1_set     : SET OF compound_representation_item := [];
      c_a_set    : SET OF applied_classification_assignment := [];
      c_a_set2   : SET OF applied_classification_assignment := [];
      l_rep_item : list_representation_item;
      t2_set     : SET OF compound_representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name =
        'hydrostatic properties for constant floating position') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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 = 'floating position') );
    REPEAT i := 1 TO HIINDEX(c_a_set2) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set2[i].items) BY 1;
        t2_set := t2_set + c_a_set2[i].items[j];
      END_REPEAT;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      l_rep_item := t1_set[i].item_element;
      REPEAT j := 1 TO HIINDEX(t1_set[i].item_element) BY 1;
        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; -- hydrostatic_properties_with_specified_class

  RULE hydrostatic_property_with_specified_name FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['property type'];
      t2_set    : SET OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'hydrostatic property') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- hydrostatic_property_with_specified_name

  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; --
identification_role_optional_attribute_description_required

  RULE initiator_for_change_request FOR (
            applied_person_and_organization_assignment,
            versioned_action_request);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF versioned_action_request := [];
      a_set   : SET OF applied_person_and_organization_assignment :=
[];
    END_LOCAL;
    t1_set := QUERY ( a <* versioned_action_request | VALUE_IN(
        which_class(a),'change request') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- initiator_for_change_request

  RULE mandatory_entity_type_for_external_instance_reference FOR (
            external_source, external_source_relationship);

    LOCAL
      violate : LOGICAL := FALSE;
      bag_1   : BAG OF external_source := [];
    END_LOCAL;
    bag_1 := QUERY ( a <* external_source | (a.description = 'schema
name') );
    REPEAT i := 1 TO SIZEOF(bag_1) BY 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; -- mandatory_entity_type_for_external_instance_reference

  RULE members_is_referenced_by_at_least_one_revision FOR (
```

```
            applied_group_assignment, group);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF group := [];
      a_set   : SET OF applied_group_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* group |
VALUE_IN(which_class(a),'revision') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- members_is_referenced_by_at_least_one_revision

  RULE no_approvals_except_in_approval_history FOR (approval);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF approval := [];
      t2_set  : SET OF applied_group_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* approval | VALUE_IN(which_class(a),
        'approval event') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violate;
      t2_set := bag_to_set(USEDIN(t1_set[i],

'SHIP_MOULDED_FORM_SCHEMA.APPLIED_GROUP_ASSIGNMENT.ITEMS'));
      violate := SIZEOF(t2_set) = 0;
      REPEAT k := 1 TO HIINDEX(t2_set) BY 1 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; -- no_approvals_except_in_approval_history

  RULE offset_point_table_model_compound_representation_has_name FOR
(
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
```



```
      arg_list  : LIST OF STRING := ['offset point table type'];
      t2_set    : SET OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'offset point table model') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
offset_point_table_model_compound_representation_has_name

  RULE principal_characteristics_has_properties FOR (
            property_definition_representation,
            applied_classification_assignment);

    LOCAL
      t3_set    : LIST OF property_definition := [];
      violation : LOGICAL := FALSE;
      t4_set    : LIST OF product_definition := [];
      t1_set    : LIST OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'principal characteristics') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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) BY 1;
      t3_set := t3_set + t2_set[i].definition;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t3_set) BY 1;
      t4_set := t4_set + t3_set[i].definition;
    END_REPEAT;
    violation := t1_set <> t4_set;

  WHERE
```

```
    wr1: (NOT violation);

  END_RULE; -- principal_characteristics_has_properties

  RULE product_definition_for_call_sign FOR (
             applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'ship designation') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.APPLIED_IDENTI
FICATION_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; -- product_definition_for_call_sign

  RULE product_definition_for_class_notation FOR (
             applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition := [];
    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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_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; -- product_definition_for_class_notation

  RULE product_definition_for_flag_state FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'ship designation') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.APPLIED_IDENTI
FICATION_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; -- product_definition_for_flag_state

  RULE
product_definition_for_hydrostatic_definition_requires_reference FOR
(
            applied_classification_assignment);

    LOCAL
      t3_set    : SET OF property_definition_representation := [];
      violation : LOGICAL := FALSE;
      t4_set    : SET OF property_definition := [];
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      c2_a_set  : SET OF applied_classification_assignment := [];
      t2_set    : SET OF representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'hydrostatic definition') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
```

```
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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 = 'hydrostatic table') );
    REPEAT i := 1 TO HIINDEX(c2_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c2_a_set[i].items) BY 1;
        t2_set := t2_set + c2_a_set[i].items[j];
      END_REPEAT;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t2_set) BY 1;
      t3_set := t3_set +
bag_to_set(USEDIN(t2_set[i],'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFIN
ITION_REPRESENTATION.USED_REPRESENTATION'));
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t3_set) BY 1;
      t4_set := t4_set + t3_set[i].definition;
    END_REPEAT;
    violation := t1_set <> t4_set;

  WHERE
    wr1: (NOT violation);

  END_RULE; --
product_definition_for_hydrostatic_definition_requires_reference

  RULE product_definition_for_managing_company FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF applied_organization_assignment := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'owner designation') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_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; -- product_definition_for_managing_company

  RULE product_definition_for_ordering_company FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF applied_organization_assignment := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'owner designation') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_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; -- product_definition_for_ordering_company

  RULE product_definition_for_owning_company FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF applied_organization_assignment := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'owner designation') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_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; -- product_definition_for_owning_company

  RULE product_definition_for_port_of_registration FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'ship designation') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.APPLIED_IDENTI
FICATION_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; -- product_definition_for_port_of_registration

  RULE product_definition_for_regulation FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition := [];
    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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_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; -- product_definition_for_regulation

  RULE product_definition_for_shipyard FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF applied_organization_assignment := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'shipyard designation') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_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; -- product_definition_for_shipyard

  RULE product_definition_for_stability_definition FOR (
            applied_classification_assignment);

    LOCAL
      t3_set    : SET OF property_definition_representation := [];
      violation : LOGICAL := FALSE;
      t4_set    : SET OF property_definition := [];
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      c2_a_set  : SET OF applied_classification_assignment := [];
      t2_set    : SET OF representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
```

```
        assigned_class.name = 'stability definition') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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) BY 1;
      REPEAT j := 1 TO HIINDEX(c2_a_set[i].items) BY 1;
        t2_set := t2_set + c2_a_set[i].items[j];
      END_REPEAT;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t2_set) BY 1;
      t3_set := t3_set +
bag_to_set(USEDIN(t2_set[i],'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFIN
ITION_REPRESENTATION.USED_REPRESENTATION'));
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t3_set) BY 1;
      t4_set := t4_set + t3_set[i].definition;
    END_REPEAT;
    violation := t1_set <> t4_set;

  WHERE
    wr1: (NOT violation);

  END_RULE; -- product_definition_for_stability_definition

  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) BY 1
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; --
product_definition_relationship_references_are_distinct

  RULE product_definition_relationship_related_to_class_moulded_form
FOR (
            applied_classification_assignment);

    LOCAL
      t3_set    : SET OF product_definition := [];
```



```
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition_relationship := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      c2_a_set  : SET OF applied_classification_assignment := [];
      t2_set    : SET OF product_definition := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'moulded form relationship') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        t1_set := t1_set + c_a_set[i].items[j];
      END_REPEAT;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1;
      t2_set := t2_set + t1_set[i].related_product_definition;
      t2_set := t2_set + t1_set[i].relating_product_definition;
    END_REPEAT;
    c2_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'moulded form') );
    REPEAT i := 1 TO HIINDEX(c2_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c2_a_set[i].items) BY 1;
        t3_set := t3_set + c2_a_set[i].items[j];
      END_REPEAT;
    END_REPEAT;
    violation := NOT (t2_set <= t3_set);

  WHERE
    wr1: (NOT violation);

  END_RULE; --
product_definition_relationship_related_to_class_moulded_form

  RULE
product_definition_relationship_with_identification_assignment FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition_relationship := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['item relationship'];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.APPLIED_IDENTI
FICATION_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; --
product_definition_relationship_with_identification_assignment

  RULE product_definition_shape_with_identification_assignment FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition_shape := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['definition'];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.APPLIED_IDENTI
FICATION_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; --
product_definition_shape_with_identification_assignment

  RULE product_definition_with_identification_assignment FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
```



```
      arg_list  : LIST OF STRING := ['definition','definable
object'];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.APPLIED_IDENTI
FICATION_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; -- product_definition_with_identification_assignment

  RULE
product_related_product_category_with_identification_assignment FOR
(
             applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_related_product_category := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['Shiptype'];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.APPLIED_IDENTI
FICATION_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; --
product_related_product_category_with_identification_assignment

  RULE product_with_identification_assignment FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['ship'];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.APPLIED_IDENTI
FICATION_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; -- product_with_identification_assignment

  RULE propeller_moulded_form_design_parameter_with_class_references
FOR (
            applied_classification_assignment);

    LOCAL
      t3_set    : SET OF representation := [];
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
```



```
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'propeller moulded form design
parameter') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
          'DEFINITION'));
      REPEAT j := 1 TO HIINDEX(t2_set) BY 1;
        t3_set := t3_set + t2_set[j].used_representation;
      END_REPEAT;
      violation := SIZEOF(QUERY ( t2_inst <* t3_set | (
          'propeller location' IN which_class(t2_inst)) )) > 1;
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; --
propeller_moulded_form_design_parameter_with_class_references

  RULE property_definition_appendage_moulded_form_design_parameter
FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'appendage moulded form design
parameter') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.' +
          'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
      violation := NOT (SIZEOF(QUERY ( t2_inst <* t2_set |
(t2_inst.name =
           'appendage moulded form design parameter') )) = 1);
    END_REPEAT;

  WHERE
    wr1: (NOT violation);
```

```
  END_RULE; --
property_definition_appendage_moulded_form_design_parameter

  RULE property_definition_for_bottom_moulded_form_design_parameter
FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'bottom moulded form design
parameter') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.' +
          'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
      violation := NOT (SIZEOF(QUERY ( t2_inst <* t2_set |
(t2_inst.name =
           'bottom moulded form design parameter') )) = 1);
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; --
property_definition_for_bottom_moulded_form_design_parameter

  RULE property_definition_for_bulb_moulded_form_design_parameter
FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'bulb moulded form design parameter')
);
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.' +
          'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
      violation := NOT (SIZEOF(QUERY ( t2_inst <* t2_set |
(t2_inst.name =
           'bulb moulded form design parameter') )) = 1);
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; --
property_definition_for_bulb_moulded_form_design_parameter

  RULE property_definition_for_class_notation FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'class notation') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.PROPERTY_DEFIN
ITION_REPRESENTATION.DEFINITION'));
      violation := NOT (SIZEOF(QUERY ( t2_inst <* t2_set |
(t2_inst.name =
           'class notation') )) = 1);
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; -- property_definition_for_class_notation

  RULE property_definition_for_class_society FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF applied_organization_assignment := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'class notation') );
```

```
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_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; -- property_definition_for_class_society

  RULE property_definition_for_deck_moulded_form_design_parameter
FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
       assigned_class.name = 'deck moulded form design parameter')
);
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.' +
          'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
      violation := NOT (SIZEOF(QUERY ( t2_inst <* t2_set |
(t2_inst.name =
           'deck moulded form design parameter') )) = 1);
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; --
property_definition_for_deck_moulded_form_design_parameter

  RULE property_definition_for_hull_moulded_form_design_parameter
FOR (
            applied_classification_assignment);
```



```
    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'hull moulded form design parameter')
);
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.' +
          'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
      violation := NOT (SIZEOF(QUERY ( t2_inst <* t2_set |
(t2_inst.name =
           'hull moulded form design parameter') )) = 1);
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; --
property_definition_for_hull_moulded_form_design_parameter

  RULE property_definition_for_local_coordinate_system FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.' +
          'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
      violation := NOT (SIZEOF(QUERY ( t2_inst <* t2_set |
(t2_inst.name =
           'local co-ordinate system') )) = 1);
    END_REPEAT;

  WHERE
```

```
    wr1: (NOT violation);

  END_RULE; -- property_definition_for_local_coordinate_system

  RULE property_definition_for_local_coordinate_system_with_position
FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.PROPERTY_DEFIN
ITION_REPRESENTATION.DEFINITION'));
      violation := NOT (SIZEOF(QUERY ( t2_inst <* t2_set |
(t2_inst.name =
           'local co-ordinate system with position reference') )) =
1);
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; --
property_definition_for_local_coordinate_system_with_position

  RULE property_definition_for_moulded_form_function_parameters FOR
(
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'moulded form functional definition')
);
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.' +
          'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
      violation := NOT (SIZEOF(QUERY ( t2_inst <* t2_set |
(t2_inst.name =
           'moulded form function parameters') )) = 1);
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; --
property_definition_for_moulded_form_function_parameters

  RULE property_definition_for_rudder_moulded_form_design_parameter
FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'rudder moulded form design
parameter') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.' +
          'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
      violation := NOT (SIZEOF(QUERY ( t2_inst <* t2_set |
(t2_inst.name =
           'rudder moulded form design parameter') )) = 1);
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; --
property_definition_for_rudder_moulded_form_design_parameter

  RULE
property_definition_for_thruster_moulded_form_design_parameter FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
```

```
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'thruster moulded form design
parameter') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.' +
          'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
      violation := NOT (SIZEOF(QUERY ( t2_inst <* t2_set |
(t2_inst.name =
           'thruster moulded form design parameter') )) = 1);
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; --
property_definition_for_thruster_moulded_form_design_parameter

  RULE property_definition_for_thruster_propeller_parameter FOR (
             applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_relationship := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'thruster moulded form design
parameter') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.' +
          'PROPERTY_DEFINITION_RELATIONSHIP' +
          '.RELATING_PROPERTY_DEFINITION'));
      violation := NOT (SIZEOF(QUERY ( t2_inst <* t2_set |
(t2_inst.name =
           'thruster propeller parameter') )) = 1);
    END_REPEAT;

  WHERE
```



```
    wr1: (NOT violation);

  END_RULE; -- property_definition_for_thruster_propeller_parameter

  RULE
property_definition_of_propeller_moulded_form_design_parameter FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
       assigned_class.name = 'propeller moulded form design
parameter') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.' +
          'PROPERTY_DEFINITION_REPRESENTATION' + '.DEFINITION'));
      violation := NOT (SIZEOF(QUERY ( t2_inst <* t2_set |
(t2_inst.name =
           'propeller moulded form design parameter') )) = 1);
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; --
property_definition_of_propeller_moulded_form_design_parameter

  RULE property_definition_with_identification_assignment FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF property_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := [
                    'moulded form characteristics definition',
                    'moulded form functional definition',
                    'local co-ordinate system','spacing table',
                    'hydrostatic definition','stability definition'];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.APPLIED_IDENTI
FICATION_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; -- property_definition_with_identification_assignment

  RULE representation_for_appendage_moulded_form_design_parameter
FOR (
            representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['appendage length',
                  'appendage breadth','appendage depth',
                  'type of appendage'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'appendage moulded form design parameter') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_for_appendage_moulded_form_design_parameter

  RULE representation_for_bottom_moulded_form_design_parameter FOR (
            representation);

    LOCAL
```



```
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['bilge radius','rise of floor',
                   'aft end of flat of bottom',
                   'front end of flat of bottom','flat of bottom
breadth',
                   'rake of keel'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <*
bag_to_set(USEDIN(temp_rep,'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINI
TION_REPRESENTATION.USED_REPRESENTATION'))
         | (temp_prop_def_rep.name = 'bottom moulded form design
parameter') ))
        > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_for_bottom_moulded_form_design_parameter

  RULE representation_for_bulb_moulded_form_design_parameter FOR (
            representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['bulb length','bulb length from
pp',
                   'bulb breadth','bulb breadth pp','bulb depth',
                   'bulb depth pp','bulb frame section area at pp',
                   'bulb location'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <*
bag_to_set(USEDIN(temp_rep,'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINI
TION_REPRESENTATION.USED_REPRESENTATION'))
         | (temp_prop_def_rep.name = 'bulb moulded form design
parameter') ))
        > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- representation_for_bulb_moulded_form_design_parameter

  RULE representation_for_class_and_statutory_designation FOR (
            representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['class number'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'class and statutory designation') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- representation_for_class_and_statutory_designation

  RULE representation_for_deck_moulded_form_design_parameter FOR (
            representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['camber','sheer at ap','sheer
at fp'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <*
bag_to_set(USEDIN(temp_rep,'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINI
TION_REPRESENTATION.USED_REPRESENTATION'))
         | (temp_prop_def_rep.name = 'deck moulded form design
parameter') ))
        > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- representation_for_deck_moulded_form_design_parameter

  RULE representation_for_global_axis_placement FOR
(representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['global axes and origin',
                  'after perpendicular offset','orientation'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'global axis placement') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- representation_for_global_axis_placement

  RULE representation_for_hull_moulded_form_design_parameter FOR (
            representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := [
                  'aft end of parallel midbody at design draught',
                  'front end of parallel midbody at design
draught',
                  'aft end of flat of side','front end of flat of
side',
                  'block coefficient','prismatic coefficient',
                  'max wetted frame section area','waterplane
coefficient'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <*
bag_to_set(USEDIN(temp_rep,'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINI
TION_REPRESENTATION.USED_REPRESENTATION'))
         | (temp_prop_def_rep.name = 'hull moulded form design
parameter') ))
```

```
        > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- representation_for_hull_moulded_form_design_parameter

  RULE representation_for_hydrostatic_table_constrained FOR (
           applied_classification_assignment);

    LOCAL
      t3_set    : SET OF representation_item := [];
      violation : LOGICAL := FALSE;
      t1_set    : SET OF representation := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      c2_a_set  : SET OF applied_classification_assignment := [];
      t2_set    : SET OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'hydrostatic table') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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 =
        'hydrostatic properties for constant floating position') );
    REPEAT i := 1 TO HIINDEX(c2_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c2_a_set[i].items) BY 1;
        t2_set := t2_set + c2_a_set[i].items[j];
      END_REPEAT;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(t1_set[i].items) BY 1;
        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; -- representation_for_hydrostatic_table_constrained

  RULE representation_for_hydrostatic_table_restricted FOR (
           applied_classification_assignment);
```



```
    LOCAL
      t3_set    : SET OF representation_item := [];
      violation : LOGICAL := FALSE;
      t1_set    : SET OF representation := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      c2_a_set  : SET OF applied_classification_assignment := [];
      t2_set    : SET OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'hydrostatic table') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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 = 'hydrostatic property') );
    REPEAT i := 1 TO HIINDEX(c2_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c2_a_set[i].items) BY 1;
        t2_set := t2_set + c2_a_set[i].items[j];
      END_REPEAT;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(t1_set[i].items) BY 1;
        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; -- representation_for_hydrostatic_table_restricted

  RULE representation_for_hydrostatic_table_restricted_by_class_id
FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF representation := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['mean shell thickness'];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'hydrostatic table') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_for_hydrostatic_table_restricted_by_class_id

  RULE representation_for_local_coordinate_system FOR
(representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['local axes and origin'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'local co-ordinate system') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- representation_for_local_coordinate_system

  RULE representation_for_midship_tumble_restricted_by_class_id FOR
(
             applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF representation := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['tumble out at bottom',
                  'tumble in at top','tumble out at side',
                  'tumble in at side'];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'midship tumble') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_for_midship_tumble_restricted_by_class_id

  RULE representation_for_moulded_form_function_parameters FOR (
            representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['function'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
       QUERY ( temp_prop_def_rep <*
bag_to_set(USEDIN(temp_rep,'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINI
TION_REPRESENTATION.USED_REPRESENTATION'))
         | (temp_prop_def_rep.name = 'moulded form function
parameters') ))
       > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- representation_for_moulded_form_function_parameters

  RULE representation_for_offset_point_table_model_for_point FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['section point'];
      t2_set    : SET OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
       assigned_class.name = 'section of offset point table') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
```

```
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- representation_for_offset_point_table_model_for_point

  RULE representation_for_offset_point_table_model_for_section FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['offset point table section'];
      t2_set    : SET OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'offset point table model') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_for_offset_point_table_model_for_section

  RULE
representation_for_offset_table_shape_representation_restricted FOR
(
            applied_classification_assignment);

    LOCAL
      t3_set    : SET OF representation_item := [];
```



```
      violation : LOGICAL := FALSE;
      t1_set    : SET OF representation := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      c2_a_set  : SET OF applied_classification_assignment := [];
      t2_set    : SET OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'offset table shape representation')
);
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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 = 'offset point table model') );
    REPEAT i := 1 TO HIINDEX(c2_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c2_a_set[i].items) BY 1;
        t2_set := t2_set + c2_a_set[i].items[j];
      END_REPEAT;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(t1_set[i].items) BY 1;
        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; --
representation_for_offset_table_shape_representation_restricted

  RULE representation_for_propeller_location_restricted_by_class_id
FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF representation := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['shaft line inclination x',
                   'shaft line inclination y','shaft line location',
                   'propeller location'];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'propeller location') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
```

```
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_for_propeller_location_restricted_by_class_id

  RULE representation_for_propeller_moulded_form_design_parameter
FOR (
             representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['type of propulsion',
                  'propeller diameter','chord length at 0 7
radius',
                  'thickness at 0 7 radius','number of propeller
blades',
                  'expanded area ratio','hub diameter ratio',
                  'nominal design pitch ratio','type of propeller
blades',
                  'rake','skew','design sense of rotation'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <*
bag_to_set(USEDIN(temp_rep,'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINI
TION_REPRESENTATION.USED_REPRESENTATION'))
         | (temp_prop_def_rep.name =
        'propeller moulded form design parameter') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_for_propeller_moulded_form_design_parameter

  RULE representation_for_rudder_moulded_form_design_parameter FOR (
             representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['rudder height','rudder mean
```



```
height',
                    'rudder length','rudder mean length','rudder
thickness',
                    'projected rudder area','type of the rudder',
                    'rudder location'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <*
bag_to_set(USEDIN(temp_rep,'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINI
TION_REPRESENTATION.USED_REPRESENTATION'))
         | (temp_prop_def_rep.name = 'rudder moulded form design
parameter') ))
        > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_for_rudder_moulded_form_design_parameter

  RULE representation_for_stability_table_restricted FOR (
            applied_classification_assignment);

    LOCAL
      t3_set    : SET OF representation_item := [];
      violation : LOGICAL := FALSE;
      t1_set    : SET OF representation := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      c2_a_set  : SET OF applied_classification_assignment := [];
      t2_set    : SET OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'stability table') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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) BY 1;
      REPEAT j := 1 TO HIINDEX(c2_a_set[i].items) BY 1;
        t2_set := t2_set + c2_a_set[i].items[j];
      END_REPEAT;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
```

```
      REPEAT j := 1 TO HIINDEX(t1_set[i].items) BY 1;
        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; -- representation_for_stability_table_restricted

  RULE representation_for_stability_table_restricted_by_class_id FOR
(
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF representation := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['mean shell thickness'];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'stability table') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_for_stability_table_restricted_by_class_id

  RULE representation_for_thruster_moulded_form_design_parameter FOR
(
            representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['thruster tunnel diameter',
                   'thruster tunnel min length',
                   'thruster tunnel max length',
                   'geometric thruster location','thruster
location'];
      reps      : BAG OF representation := [];
    END_LOCAL;
```



```
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <*
bag_to_set(USEDIN(temp_rep,'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINI
TION_REPRESENTATION.USED_REPRESENTATION'))
         | (temp_prop_def_rep.name =
        'thruster moulded form design parameter') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_for_thruster_moulded_form_design_parameter

  RULE representation_has_global_uncertainty_assigned_context FOR (
           shape_representation);

    LOCAL
      has_gunac : LOGICAL := TRUE;
    END_LOCAL;
    REPEAT i := 1 TO HIINDEX(shape_representation) BY 1 WHILE
has_gunac;
      has_gunac :=

'SHIP_MOULDED_FORM_SCHEMA.GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT' IN
          TYPEOF(shape_representation[i].context_of_items);
    END_REPEAT;

  WHERE
    wr1: has_gunac;

  END_RULE; --
representation_has_global_uncertainty_assigned_context

  RULE representation_has_global_unit_assigned_context FOR
(representation);

    LOCAL
      has_guac : LOGICAL := TRUE;
    END_LOCAL;
    REPEAT i := 1 TO HIINDEX(representation) BY 1 WHILE has_guac;
      REPEAT j := 1 TO SIZEOF(representation[i].items) BY 1 WHILE
has_guac;
        IF ('SHIP_MOULDED_FORM_SCHEMA.VALUE_REPRESENTATION_ITEM' IN
            TYPEOF(representation[i].items[j])) OR (
            'SHIP_MOULDED_FORM_SCHEMA.GEOMETRIC_REPRESENTATION_ITEM'
IN
            TYPEOF(representation[i].items[j])) THEN
          has_guac :=
```

```
'SHIP_MOULDED_FORM_SCHEMA.GLOBAL_UNIT_ASSIGNED_CONTEXT' IN
              TYPEOF(representation[i].context_of_items);
        END_IF;
      END_REPEAT;
    END_REPEAT;

  WHERE
    wr1: has_guac;

  END_RULE; -- representation_has_global_unit_assigned_context

  RULE representation_item_for_transformation_to_parent FOR (
            applied_classification_assignment);

    LOCAL
      t3_set     : SET OF representation := [];
      t4_set     : SET OF representation_map := [];
      t1_set     : SET OF property_definition := [];
      t5_set     : SET OF mapped_item := [];
      c_a_set    : SET OF applied_classification_assignment := [];
      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;
      t2_set     : SET OF property_definition_representation := [];
    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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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 violation1;
      t2_set :=
bag_to_set(USEDIN(t1_set[i],'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFIN
ITION_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) BY 1 WHILE NOT violation1;
      t4_set :=
bag_to_set(USEDIN(t3_set[i],'SHIP_MOULDED_FORM_SCHEMA.REPRESENTATION
_MAP.MAPPED_REPRESENTATION'));
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation1;
      t5_set := bag_to_set(USEDIN(t4_set[i],
          'SHIP_MOULDED_FORM_SCHEMA.MAPPED_ITEM.MAPPING_SOURCE'));
```



```
      REPEAT j := 1 TO 3 BY 1 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; -- representation_item_for_transformation_to_parent

  RULE representation_items_appendage_moulded_form_design_parameter
FOR (
             representation);

    LOCAL
      found    : LOGICAL := FALSE;
      arg_list : LIST OF STRING := ['moulded form outer surface',
                 'moulded form displacement','user def appendage
type'];
      reps     : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'appendage moulded form design parameter') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT found;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_items_appendage_moulded_form_design_parameter

  RULE representation_items_for_bottom_moulded_form_design_parameter
FOR (
             representation);

    LOCAL
      found    : LOGICAL := FALSE;
      arg_list : LIST OF STRING := ['moulded form outer surface',
                 'moulded form displacement'];
      reps     : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
```



```
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'bottom moulded form design parameter') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT found;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_items_for_bottom_moulded_form_design_parameter

  RULE representation_items_for_bulb_moulded_form_design_parameter
FOR (
            representation);

    LOCAL
      found    : LOGICAL := FALSE;
      arg_list : LIST OF STRING := ['moulded form outer surface',
                 'moulded form displacement'];
      reps     : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'bulb moulded form design parameter') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT found;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_items_for_bulb_moulded_form_design_parameter

  RULE representation_items_for_deck_moulded_form_design_parameter
FOR (
            representation);

    LOCAL
      found    : LOGICAL := FALSE;
```



```
      arg_list : LIST OF STRING := ['moulded form outer surface',
                 'moulded form displacement'];
      reps     : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'deck moulded form design parameter') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT found;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_items_for_deck_moulded_form_design_parameter

  RULE representation_items_for_hull_moulded_form_design_parameter
FOR (
             representation);

    LOCAL
      found    : LOGICAL := FALSE;
      arg_list : LIST OF STRING := ['moulded form outer surface',
                 'moulded form displacement',
                 'waterline angle of entrance at stern',
                 'waterline angle of entrance at bow',
                 'max frame section area location','hull length
pp',
                 'hull length waterline','hull breadth','hull
depth',
                 'hull design draught','gunwale radius'];
      reps     : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'hull moulded form design parameter') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT found;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_items_for_hull_moulded_form_design_parameter

  RULE representation_items_for_moulded_form_design_parameters FOR (
            representation);

    LOCAL
      found    : LOGICAL := FALSE;
      arg_list : LIST OF STRING := ['status'];
      reps     : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'moulded form design parameters') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT found;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_items_for_moulded_form_design_parameters

  RULE representation_items_for_moulded_form_function_parameters FOR
(
            representation);

    LOCAL
      found    : LOGICAL := FALSE;
      arg_list : LIST OF STRING := ['user def function'];
      reps     : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <*
bag_to_set(USEDIN(temp_rep,'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINI
TION_REPRESENTATION.USED_REPRESENTATION'))
         | (temp_prop_def_rep.name = 'moulded form function
parameters') ))
        > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT found;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_items_for_moulded_form_function_parameters

  RULE representation_items_for_rudder_moulded_form_design_parameter
FOR (
            representation);

    LOCAL
      found    : LOGICAL := FALSE;
      arg_list : LIST OF STRING := ['moulded form outer surface',
                 'moulded form displacement'];
      reps     : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'rudder moulded form design parameter') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT found;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_items_for_rudder_moulded_form_design_parameter

  RULE
representation_items_of_thruster_moulded_form_design_parameter FOR (
            representation);

    LOCAL
      found    : LOGICAL := FALSE;
      arg_list : LIST OF STRING := ['moulded form outer surface',
                 'moulded form displacement'];
      reps     : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'thruster moulded form design parameter') )) > 0) );
```

```
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT found;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_items_of_thruster_moulded_form_design_parameter

  RULE representation_items_optional_for_class_notation FOR (
            representation);

    LOCAL
      found    : LOGICAL := FALSE;
      arg_list : LIST OF STRING := ['ice class notation','service
factor',
                 'approval required for heavy cargo'];
      reps     : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'class notation') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT found;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- representation_items_optional_for_class_notation

  RULE representation_items_optional_for_owner_designation FOR (
            representation);

    LOCAL
      found    : LOGICAL := FALSE;
      arg_list : LIST OF STRING := ['owner approval'];
      reps     : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
```



```
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'owner designation') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT found;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- representation_items_optional_for_owner_designation

  RULE representation_items_optional_for_principal_characteristics
FOR (
             representation);

    LOCAL
      found    : LOGICAL := FALSE;
      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'];
      reps     : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'principal characteristics') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT found;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_items_optional_for_principal_characteristics

  RULE representation_items_propeller_moulded_form_design_parameter
FOR (
             representation);

    LOCAL
      found    : LOGICAL := FALSE;
      arg_list : LIST OF STRING := ['blade_mean_height',
```

```
                  'moulded form outer surface','moulded form
displacement'];
      reps     : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'propeller moulded form design parameter') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT found;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_items_propeller_moulded_form_design_parameter

  RULE
representation_of_local_coordinate_system_with_position_reference
FOR (
             representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['local axes and origin'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'local co-ordinate system with position reference') )) > 0)
);
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_of_local_coordinate_system_with_position_reference
```



```
  RULE representation_restricted_by_name_class_notation FOR (
            representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['service area',
                  'approval required for oil cargo',
                  'approval required for loading unloading
aground',
                  'approval required for unloading grabs'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'class notation') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- representation_restricted_by_name_class_notation

  RULE representation_restricted_by_name_class_parameters FOR (
            representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['length class','length solas',
                  'scantlings draught','block coefficient class',
                  'design speed ahead','design speed astern'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'class parameters') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- representation_restricted_by_name_class_parameters

  RULE representation_restricted_by_name_principal_characteristics
FOR (
            representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['length between
perpendiculars',
                  'moulded breadth','moulded depth'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'principal characteristics') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_restricted_by_name_principal_characteristics

  RULE representation_restricted_by_name_ship_overall_dimensions FOR
(
            representation);

    LOCAL
      violation : LOGICAL := FALSE;
      arg_list  : LIST OF STRING := ['overall breadth','overall
depth',
                  'overall length','stem overhang','stern
overhang'];
      reps      : BAG OF representation := [];
    END_LOCAL;
    reps := QUERY ( temp_rep <* representation | (SIZEOF(
        QUERY ( temp_prop_def_rep <* bag_to_set(USEDIN(temp_rep,

'SHIP_MOULDED_FORM_SCHEMA.PROPERTY_DEFINITION_REPRESENTATION.' +
        'USED_REPRESENTATION')) | (temp_prop_def_rep.name =
        'ship overall dimensions') )) > 0) );
    REPEAT i := 1 TO HIINDEX(reps) BY 1 WHILE NOT violation;
```



```
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
representation_restricted_by_name_ship_overall_dimensions

  RULE revision_has_mandatory_attribute_description FOR (group);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF group := [];
    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; -- revision_has_mandatory_attribute_description

  RULE revision_with_context_referenced_for_context_of_revision FOR
(
            applied_group_assignment, group);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF group := [];
      a_set   : SET OF applied_group_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* group | VALUE_IN(which_class(a),
        'revision with context') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; --
revision_with_context_referenced_for_context_of_revision

  RULE ship_curve_has_name FOR (applied_classification_assignment);
```

```
    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['side condition','curve
shape'];
      t2_set    : SET OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'ship curve') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- ship_curve_has_name

  RULE ship_curve_segment_has_class FOR
(applied_classification_assignment);

    LOCAL
      t3_set     : SET OF representation_item := [];
      violation  : LOGICAL := FALSE;
      t1_set     : SET OF compound_representation_item := [];
      c_a_set    : SET OF applied_classification_assignment := [];
      c_a_set2   : SET OF applied_classification_assignment := [];
      l_rep_item : list_representation_item;
      t2_set     : SET OF compound_representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'ship curve segment') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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 = 'ship curve') );
    REPEAT i := 1 TO HIINDEX(c_a_set2) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set2[i].items) BY 1;
        t2_set := t2_set + c_a_set2[i].items[j];
      END_REPEAT;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
```



```
    REPEAT j := 1 TO HIINDEX(t1_set[i].item_element) BY 1;
      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; -- ship_curve_segment_has_class

RULE ship_curve_with_spacing_position_has_class FOR (
           applied_classification_assignment);

  LOCAL
    t3_set     : SET OF representation_item := [];
    violation  : LOGICAL := FALSE;
    t1_set     : SET OF compound_representation_item := [];
    c_a_set    : SET OF applied_classification_assignment := [];
    c_a_set2   : SET OF applied_classification_assignment := [];
    l_rep_item : list_representation_item;
    t2_set     : SET OF compound_representation_item := [];
  END_LOCAL;
  c_a_set := QUERY ( i <* applied_classification_assignment | (i.
      assigned_class.name = 'ship curve with spacing position') );
  REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
    REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
      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) BY 1;
    REPEAT j := 1 TO HIINDEX(c_a_set2[i].items) BY 1;
      t2_set := t2_set + c_a_set2[i].items[j];
    END_REPEAT;
  END_REPEAT;
  REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
    REPEAT j := 1 TO HIINDEX(t1_set[i].item_element) BY 1;
      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; -- ship_curve_with_spacing_position_has_class

RULE ship_designation_has_one_specified_names FOR (
           applied_classification_assignment);
```

```
    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'ship designation') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.APPLIED_IDENTI
FICATION_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; -- ship_designation_has_one_specified_names

  RULE ship_moulded_form_revision_has_description FOR (
            product_definition_relationship);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF product_definition_relationship := [];
    END_LOCAL;
    t1_set := QUERY ( i <* product_definition_relationship |
VALUE_IN(
        which_class(i),'ship moulded form revision') );
    violate := SIZEOF(QUERY ( k <* t1_set | (NOT
EXISTS(k.description)) ))
        > 0;

  WHERE
    wr1: (NOT violate);

  END_RULE; -- ship_moulded_form_revision_has_description

  RULE ship_overall_dimensions_has_properties FOR (
            property_definition_representation,
            applied_classification_assignment);

    LOCAL
      t3_set    : LIST OF property_definition := [];
      violation : LOGICAL := FALSE;
      t4_set    : LIST OF product_definition := [];
```



```
      t1_set    : LIST OF product_definition := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      t2_set    : SET OF property_definition_representation := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'ship overall dimensions') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        t1_set := t1_set + c_a_set[i].items[j];
      END_REPEAT;
    END_REPEAT;
    t2_set := QUERY ( i <* property_definition_representation |
(i.name =
        'ship overall dimensions') );
    REPEAT i := 1 TO HIINDEX(t2_set) BY 1;
      t3_set := t3_set + t2_set[i].definition;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t3_set) BY 1;
      t4_set := t4_set + t3_set[i].definition;
    END_REPEAT;
    violation := t1_set <> t4_set;

  WHERE
    wr1: (NOT violation);

  END_RULE; -- ship_overall_dimensions_has_properties

  RULE ship_point_compound_representation_has_name FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['point shape'];
      t2_set    : SET OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'ship point') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- ship_point_compound_representation_has_name

  RULE ship_surface_compound_representation_has_name FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['surface shape'];
      t2_set    : SET OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'ship surface') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- ship_surface_compound_representation_has_name

  RULE spacing_position_compound_representation_has_name FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['position number','position'];
      t2_set    : SET OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'spacing position') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- spacing_position_compound_representation_has_name

  RULE
spacing_position_with_offset_compound_representation_has_class FOR (
            applied_classification_assignment);

    LOCAL
      t3_set    : SET OF representation_item := [];
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      c_a_set2  : SET OF applied_classification_assignment := [];
      t2_set    : SET OF compound_representation_item := [];
    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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set2[i].items) BY 1;
        t2_set := t2_set + c_a_set2[i].items[j];
      END_REPEAT;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(t1_set[i].item_element) BY 1;
        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; --
spacing_position_with_offset_compound_representation_has_class

  RULE spacing_position_with_offset_compound_representation_has_name
FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
```

```
      arg_list  : LIST OF STRING := ['offset'];
      t2_set    : SET OF representation_item := [];
    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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; --
spacing_position_with_offset_compound_representation_has_name

  RULE stability_properties_for_floating_position_has_class FOR (
             applied_classification_assignment);

    LOCAL
      t3_set     : SET OF representation_item := [];
      violation  : LOGICAL := FALSE;
      t1_set     : SET OF compound_representation_item := [];
      c_a_set    : SET OF applied_classification_assignment := [];
      c_a_set2   : SET OF applied_classification_assignment := [];
      l_rep_item : list_representation_item;
      t2_set     : SET OF compound_representation_item := [];
    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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set2[i].items) BY 1;
        t2_set := t2_set + c_a_set2[i].items[j];
      END_REPEAT;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(t1_set[i].item_element) BY 1;
        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; -- stability_properties_for_floating_position_has_class

  RULE stability_properties_for_floating_position_has_name FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['centre of gravity above keel',
                   'definition of starting floating position'];
      t2_set    : SET OF representation_item := [];
    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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- stability_properties_for_floating_position_has_name

  RULE stability_property_has_name FOR
(applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF compound_representation_item := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['angle of heel','righting arm',
                   'centre of buoyancy'];
      t2_set    : SET OF representation_item := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'stability property') );
```

```
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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;
      REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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; -- stability_property_has_name

  RULE unique_approvals_in_approval_history FOR (group,
            applied_group_assignment);

    LOCAL
      violate : LOGICAL := FALSE;
      t3_set  : SET OF approval := [];
      t1_set  : SET OF group := [];
      t2_set  : SET OF applied_group_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( i <* group | VALUE_IN(which_class(i),
        'approval history') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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_MOULDED_FORM_SCHEMA.APPROVAL' IN TYPEOF(b)) );
      violate := NOT VALUE_UNIQUE(t3_set);
    END_REPEAT;

  WHERE
    wr1: (NOT violate);

  END_RULE; -- unique_approvals_in_approval_history

  RULE user_def_appendage_type_description_required FOR
(representation);

    LOCAL
      violation : LOGICAL := FALSE;
    END_LOCAL;
    REPEAT i := 1 TO HIINDEX(representation) BY 1 WHILE NOT
violation;
      violation := (SIZEOF(QUERY ( r <* representation[i].items | ((
          'SHIP_MOULDED_FORM_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
IN
          TYPEOF(r)) AND (r.name = 'type of appendage') AND (r\
```



```
          descriptive_representation_item.description = 'user
defined')) ))
          > 0) AND (SIZEOF(QUERY ( r <* representation[i].items |
(r.name =
           'user def appendage type') )) = 0);
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; -- user_def_appendage_type_description_required

  RULE user_def_function_description_required FOR (representation);

    LOCAL
      violation : LOGICAL := FALSE;
    END_LOCAL;
    REPEAT i := 1 TO HIINDEX(representation) BY 1 WHILE NOT
violation;
      violation := (SIZEOF(QUERY ( r <* representation[i].items | ((
          'SHIP_MOULDED_FORM_SCHEMA.DESCRIPTIVE_REPRESENTATION_ITEM'
IN
          TYPEOF(r)) AND (r.name = 'function') 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; -- user_def_function_description_required

  RULE valid_product_definition_for_class_moulded_form FOR (
            applied_classification_assignment,
applied_group_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF product_definition := [];
      violate1  : LOGICAL;
      violate2  : LOGICAL;
      c_a_set   : SET OF applied_classification_assignment := [];
      gr_ass    : SET OF applied_group_assignment := [];
      groups    : SET OF group := [];
    END_LOCAL;
    c_a_set := QUERY ( i <* applied_classification_assignment | (i.
        assigned_class.name = 'ship moulded form') );
    REPEAT i := 1 TO HIINDEX(c_a_set) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        t1_set := t1_set + c_a_set[i].items[j];
      END_REPEAT;
    END_REPEAT;
```



```
    gr_ass := QUERY ( i <* applied_group_assignment | (i.role.name =
        'equivalence') );
    REPEAT i := 1 TO HIINDEX(gr_ass) BY 1;
      REPEAT j := 1 TO HIINDEX(gr_ass[i].items) BY 1;
        IF gr_ass[i].items[j] IN t1_set THEN
          groups := groups + gr_ass[i].assigned_group;
        END_IF;
      END_REPEAT;
    END_REPEAT;
    gr_ass := QUERY ( i <* applied_group_assignment |
((SIZEOF(i.items) <>
         0) AND (i.role.name = 'item structure') AND
(i.assigned_group IN
        groups)) );
    REPEAT i := 1 TO HIINDEX(gr_ass) BY 1 WHILE NOT violation;
      REPEAT j := 1 TO HIINDEX(gr_ass[i].items) BY 1 WHILE NOT
violation;
        violate1 := VALUE_IN(which_class(gr_ass[i].items[j]),
            'moulded form');
        violate2 := VALUE_IN(which_class(gr_ass[i].items[j]),
            'moulded form relationship');
        violation := NOT (violate1 OR violate2);
      END_REPEAT;
    END_REPEAT;

  WHERE
    wr1: (NOT violation);

  END_RULE; -- valid_product_definition_for_class_moulded_form

  RULE version_creation_has_mandatory_attribute_description FOR
(action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action := [];
    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; -- version_creation_has_mandatory_attribute_description

  RULE version_deletion_has_mandatory_attribute_description FOR
(action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action := [];
    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; -- version_deletion_has_mandatory_attribute_description

  RULE version_history_has_exactly_one_assigned_group FOR (
            applied_group_assignment, group);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF group := [];
      set_1   : SET OF applied_group_assignment := [];
      set_3   : SET OF group_item := [];
      set_2   : SET OF applied_group_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* group | VALUE_IN(which_class(a),
        'version history') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- version_history_has_exactly_one_assigned_group

  RULE version_history_is_referenced_by_at_least_one_versions FOR (
            applied_group_assignment, group);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF group := [];
      a_set   : SET OF applied_group_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* group | VALUE_IN(which_class(a),
        'version history') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; --
version_history_is_referenced_by_at_least_one_versions

  RULE version_history_referenced_by_exactly_one_current_version FOR
(
            applied_group_assignment, group);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF group := [];
      a_set   : SET OF applied_group_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* group | VALUE_IN(which_class(a),
        'version history') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; --
version_history_referenced_by_exactly_one_current_version

  RULE version_history_referenced_by_multiple_roles FOR (
            applied_group_assignment, group);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF group := [];
      a_set   : SET OF applied_group_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* group | VALUE_IN(which_class(a),
        'version history') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- version_history_referenced_by_multiple_roles

  RULE version_modification_has_mandatory_attribute_description FOR
(
            action);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF action := [];
    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; --
version_modification_has_mandatory_attribute_description

  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_MOULDED_FO
RM_SCHEMA.IDENTIFICATION_ASSIGNMENT_RELATIONSHIP.'
          + 'RELATING_IDENTIFICATION_ASSIGNMENT')) > 0) OR
(SIZEOF(USEDIN(

applied_identification_assignment[i],'SHIP_MOULDED_FORM_SCHEMA.IDENT
IFICATION_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; --
```

```
version_relationship_associates_with_versionable_object

  RULE version_relationship_has_mandatory_attribute_description FOR
(
            identification_assignment_relationship);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF identification_assignment_relationship := [];
    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; --
version_relationship_has_mandatory_attribute_description

  RULE version_relationship_has_unique_versions FOR (
            identification_assignment_relationship);

    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF identification_assignment_relationship := [];
    END_LOCAL;
    t1_set := QUERY ( a <* identification_assignment_relationship |
        VALUE_IN(which_class(a),'version relationship') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; -- version_relationship_has_unique_versions

  RULE versionable_object_has_one_version_id FOR (
            applied_identification_assignment);

    LOCAL
      version_ids         : SET OF applied_identification_assignment
:= [];
      duplicate           : LOGICAL := FALSE;
      versionable_objects : BAG OF identification_item := [];
    END_LOCAL;
    version_ids := QUERY ( i <* applied_identification_assignment |
(i.
        role.name = 'version identifier') );
    REPEAT i := 1 TO HIINDEX(version_ids) BY 1;
```



```
      versionable_objects := versionable_objects +
version_ids[i].items;
    END_REPEAT;
    REPEAT i := 1 TO HIINDEX(versionable_objects) BY 1 WHILE NOT
duplicate;
      REPEAT j := i + 1 TO HIINDEX(versionable_objects) BY 1 WHILE
NOT
          duplicate;
        duplicate := versionable_objects[i] :=:
versionable_objects[j];
      END_REPEAT;
    END_REPEAT;

  WHERE
    wr1: (NOT duplicate);

  END_RULE; -- versionable_object_has_one_version_id

  RULE versioned_action_request_with_identification_assignment FOR (
            applied_classification_assignment);

    LOCAL
      violation : LOGICAL := FALSE;
      t1_set    : SET OF versioned_action_request := [];
      c_a_set   : SET OF applied_classification_assignment := [];
      arg_list  : LIST OF STRING := ['change request'];
      t2_set    : SET OF applied_identification_assignment := [];
    END_LOCAL;
    REPEAT j := 1 TO HIINDEX(arg_list) BY 1 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) BY 1;
      REPEAT j := 1 TO HIINDEX(c_a_set[i].items) BY 1;
        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_MOULDED_FORM_SCHEMA.APPLIED_IDENTI
FICATION_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; --
versioned_action_request_with_identification_assignment

  RULE versions_is_referenced_by_at_least_one_version_history FOR (
            applied_group_assignment, group);
```

```
    LOCAL
      violate : LOGICAL := FALSE;
      t1_set  : SET OF group := [];
      a_set   : SET OF applied_group_assignment := [];
    END_LOCAL;
    t1_set := QUERY ( a <* group |
VALUE_IN(which_class(a),'versions') );
    REPEAT i := 1 TO HIINDEX(t1_set) BY 1 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; --
versions_is_referenced_by_at_least_one_version_history

  FUNCTION acyclic_curve_replica(
              rep: curve_replica;
              parent: curve
      ): BOOLEAN;
    IF NOT ('SHIP_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_point_replica(
              rep: point_replica;
              parent: point
      ): BOOLEAN;
    IF NOT ('SHIP_MOULDED_FORM_SCHEMA.POINT_REPLICA' IN
TYPEOF(parent))
         THEN
      RETURN(TRUE);
    END_IF;
    IF parent :=: rep THEN
      RETURN(FALSE);
    ELSE

RETURN(acyclic_point_replica(rep,parent\point_replica.parent_pt));
    END_IF;

  END_FUNCTION; -- acyclic_point_replica

  FUNCTION acyclic_product_category_relationship(
              relation: product_category_relationship;
              children: SET OF product_category
      ): BOOLEAN;

    LOCAL
      x              : SET OF product_category_relationship;
      local_children : SET OF product_category;
    END_LOCAL;
    REPEAT i := 1 TO HIINDEX(children) BY 1;
      IF relation.category :=: children[i] THEN
        RETURN(FALSE);
      END_IF;
```

```
    END_REPEAT;
    x :=
bag_to_set(USEDIN(relation.category,'SHIP_MOULDED_FORM_SCHEMA.' +
         'PRODUCT_CATEGORY_RELATIONSHIP.SUB_CATEGORY'));
    local_children := children + relation.category;
    IF SIZEOF(x) > 0 THEN
      REPEAT i := 1 TO HIINDEX(x) BY 1;
        IF NOT
acyclic_product_category_relationship(x[i],local_children)
            THEN
          RETURN(FALSE);
        END_IF;
      END_REPEAT;
    END_IF;
    RETURN(TRUE);

  END_FUNCTION; -- acyclic_product_category_relationship

  FUNCTION acyclic_surface_replica(
              rep: surface_replica;
              parent: surface
      ): BOOLEAN;
    IF NOT ('SHIP_MOULDED_FORM_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_MOULDED_FORM_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: GENERIC:item;
              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_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.PCURVE' IN TYPEOF(c\
          composite_curve.segments[k].parent_curve))) AND (NOT (
          'SHIP_MOULDED_FORM_SCHEMA.SURFACE_CURVE' IN TYPEOF(c\
          composite_curve.segments[k].parent_curve))) AND (NOT (
          'SHIP_MOULDED_FORM_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: INTEGER;
              up_knots: INTEGER;
              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_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.CARTESIAN_POINT' IN TYPEOF(item)
THEN
        dim := SIZEOF(item\cartesian_point.coordinates);
        RETURN(dim);
      END_IF;
      IF 'SHIP_MOULDED_FORM_SCHEMA.DIRECTION' IN TYPEOF(item) THEN
        dim := SIZEOF(item\direction.direction_ratios);
        RETURN(dim);
      END_IF;
      IF 'SHIP_MOULDED_FORM_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));
      OTHERWISE     :         RETURN(?);
      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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.PCURVE' IN TYPEOF(c) THEN
      surfs := [c\pcurve.basis_surface];
    ELSE
      IF 'SHIP_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.REPRESENTATION.ITEMS')
        * cntxt.representations_in_context) > 0 THEN
      RETURN(TRUE);
    ELSE
      y := QUERY ( z <* USEDIN(item,'') | (
          'SHIP_MOULDED_FORM_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: INTEGER;
```

```
              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]];
    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: INTEGER;
              u1: INTEGER;
              low2: INTEGER;
              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)];
  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_MOULDED_FORM_SCHEMA.POLY_LOOP' IN
TYPEOF(l[1]);
    REPEAT i := 2 TO SIZEOF(l) BY 1;
      IF ('SHIP_MOULDED_FORM_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: representation_item
      ): BOOLEAN;
    IF SIZEOF(['SHIP_MOULDED_FORM_SCHEMA.BOUNDED_CURVE',
        'SHIP_MOULDED_FORM_SCHEMA.CONIC',
        'SHIP_MOULDED_FORM_SCHEMA.CURVE_REPLICA',
        'SHIP_MOULDED_FORM_SCHEMA.LINE',
        'SHIP_MOULDED_FORM_SCHEMA.OFFSET_CURVE_3D'] * TYPEOF(cv)) >
1 THEN
      RETURN(FALSE);
    ELSE
      IF (('SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.CONIC',
            'SHIP_MOULDED_FORM_SCHEMA.LINE'] * TYPEOF(cv)) = 1 THEN
          RETURN(TRUE);
        ELSE
          IF 'SHIP_MOULDED_FORM_SCHEMA.CURVE_REPLICA' IN TYPEOF(cv)
THEN
            RETURN(nmsf_curve_check(cv\curve_replica.parent_curve));
          ELSE
```

```
            IF ('SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.POLYLINE' IN TYPEOF(cv\
                offset_curve_3d.basis_curve))) THEN

RETURN(nmsf_curve_check(cv\offset_curve_3d.basis_curve));
            ELSE
              IF 'SHIP_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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;
    RETURN(FALSE);

  END_FUNCTION; -- nmsf_curve_check

  FUNCTION nmsf_surface_check(
              surf: surface
      ): BOOLEAN;
    IF 'SHIP_MOULDED_FORM_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF(surf)
THEN
      RETURN(TRUE);
    ELSE
      IF 'SHIP_MOULDED_FORM_SCHEMA.SWEPT_SURFACE' IN TYPEOF(surf)
THEN
        RETURN(nmsf_curve_check(surf\swept_surface.swept_curve));
      ELSE
        IF (('SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.SURFACE_REPLICA' IN
TYPEOF(surf)
              THEN

RETURN(nmsf_surface_check(surf\surface_replica.parent_surface));
          ELSE
            IF (('SHIP_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.OPEN_SHELL' IN TYPEOF(a_shell) THEN
      RETURN(open_shell_reversed(a_shell));
    ELSE
      IF 'SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.EDGE' IN TYPEOF(an_item) THEN
    RETURN(edge_reversed(an_item));
  END_IF;
  IF 'SHIP_MOULDED_FORM_SCHEMA.PATH' IN TYPEOF(an_item) THEN
    RETURN(path_reversed(an_item));
  END_IF;
  IF 'SHIP_MOULDED_FORM_SCHEMA.FACE_BOUND' IN TYPEOF(an_item) THEN
    RETURN(face_bound_reversed(an_item));
  END_IF;
  IF 'SHIP_MOULDED_FORM_SCHEMA.FACE' IN TYPEOF(an_item) THEN
    RETURN(face_reversed(an_item));
  END_IF;
  IF 'SHIP_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z))
OR (
        'SHIP_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_SCHEMA.CELSIUS_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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_MOULDED_FORM_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 valid_wireframe_edge_curve(
              crv: curve
      ): BOOLEAN;
    IF SIZEOF(['SHIP_MOULDED_FORM_SCHEMA.LINE',
        'SHIP_MOULDED_FORM_SCHEMA.CONIC',
        'SHIP_MOULDED_FORM_SCHEMA.B_SPLINE_CURVE',
        'SHIP_MOULDED_FORM_SCHEMA.POLYLINE'] * TYPEOF(crv)) = 1 THEN
      RETURN(TRUE);
    ELSE
      IF 'SHIP_MOULDED_FORM_SCHEMA.CURVE_REPLICA' IN TYPEOF(crv)
THEN

RETURN(valid_wireframe_edge_curve(crv\curve_replica.parent_curve));
      ELSE
        IF 'SHIP_MOULDED_FORM_SCHEMA.OFFSET_CURVE_3D' IN TYPEOF(crv)
THEN
          RETURN(valid_wireframe_edge_curve(crv\offset_curve_3d.
              basis_curve));
        END_IF;
      END_IF;
    END_IF;
    RETURN(FALSE);

  END_FUNCTION; -- valid_wireframe_edge_curve

  FUNCTION valid_wireframe_vertex_point(
              pnt: point
```

```
      ): BOOLEAN;
    IF 'SHIP_MOULDED_FORM_SCHEMA.CARTESIAN_POINT' IN TYPEOF(pnt)
THEN
      RETURN(TRUE);
    ELSE
      IF 'SHIP_MOULDED_FORM_SCHEMA.POINT_REPLICA' IN TYPEOF(pnt)
THEN

RETURN(valid_wireframe_vertex_point(pnt\point_replica.parent_pt));
      END_IF;
    END_IF;
    RETURN(FALSE);

  END_FUNCTION; -- valid_wireframe_vertex_point

  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_MOULDED_FORM_SCHEMA.VECTOR' IN TYPEOF(arg1) THEN
          mag1 := arg1.magnitude;
          vec1 := arg1.orientation;
        ELSE
          mag1 := 1;
          vec1 := arg1;
        END_IF;
        IF 'SHIP_MOULDED_FORM_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_MOULDED_FORM_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_moulded_form_schema
```

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 — AIM short names of entities

Entity name	Short name
ACTION	ACTION
ACTION_ASSIGNMENT	ACTASS
ACTION_METHOD	ACTMTH
ACTION_RELATIONSHIP	ACTRLT
ACTION_REQUEST_ASSIGNMENT	ACRQAS
ACTION_REQUEST_SOLUTION	ACRQSL
ADDRESS	ADDRSS
ADVANCED_FACE	ADVFC
AMOUNT_OF_SUBSTANCE_UNIT	AOSU
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

Entity name	Short name
APPLIED_ORGANIZATION_ASSIGNMENT	APORAS
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
CLASS	CLASS

Entity name	Short name
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_EDGE_SET	CNEDST
CONNECTED_FACE_SET	CNFCST
CONTEXT_DEPENDENT_UNIT	CNDPUN
CONVERSION_BASED_UNIT	CNBSUN
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
DIMENSIONAL_EXPONENTS	DMNEXP

Entity name	Short name
DIRECTION	DRCTN
DOCUMENT	DCMNT
DOCUMENT_REFERENCE	DCMRFR
DOCUMENT_REPRESENTATION_TYPE	DCRPTY
DOCUMENT_TYPE	DCMTYP
DOCUMENT_USAGE_CONSTRAINT	DCUSCN
EDGE	EDGE
EDGE_BASED_WIREFRAME_MODEL	EBWM
EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION	EBWSR
EDGE_CURVE	EDGCRV
EDGE_LOOP	EDGLP
EFFECTIVITY	EFFCTV
EFFECTIVITY_ASSIGNMENT	EFFASS
ELECTRIC_CURRENT_UNIT	ELCRUN
ELEMENTARY_SURFACE	ELMSRF
ELLIPSE	ELLPS
EVALUATED_DEGENERATE_PCURVE	EVDGPC
EXECUTED_ACTION	EXCACT
EXTERNAL_IDENTIFICATION_ASSIGNMENT	EXIDAS
EXTERNAL_SOURCE	EXTSRC
EXTERNAL_SOURCE_RELATIONSHIP	EXSRRL
EXTERNALLY_DEFINED_ITEM	EXDFIT
FACE	FACE
FACE_BASED_SURFACE_MODEL	FBSM
FACE_BOUND	FCBND
FACE_OUTER_BOUND	FCOTBN
FACE_SURFACE	FCSRF
FACETED_BREP	FCTBR
FOUNDED_ITEM	FNDITM

Entity name	Short name
FUNCTIONALLY_DEFINED_TRANSFORMATION	FNDFTR
GEOMETRIC_CURVE_SET	GMCRST
GEOMETRIC_REPRESENTATION_CONTEXT	GMRPCN
GEOMETRIC_REPRESENTATION_ITEM	GMRPIT
GEOMETRIC_SET	GMTST
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_MEASURE_WITH_UNIT	LMWU
LENGTH_UNIT	LNGUNT
LINE	LINE
LOCAL_TIME	LCLTM
LOOP	LOOP
LUMINOUS_INTENSITY_UNIT	LMINUN
MANIFOLD_SOLID_BREP	MNSLBR
MAPPED_ITEM	MPPITM
MASS_MEASURE_WITH_UNIT	MMWU
MASS_UNIT	MSSUNT
MEASURE_WITH_UNIT	MSWTUN
NAME_ATTRIBUTE	NMATT

Entity name	Short name
NAMED_UNIT	NMDUNT
NON_MANIFOLD_SURFACE_SHAPE_REPRESENTATION	NMSSR
OBJECT_ROLE	OBJRL
OFFSET_CURVE_3D	OFCR3D
OFFSET_SURFACE	OFFSRF
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

Entity name	Short name
PLACEMENT	PLCMNT
PLANE	PLANE
PLANE_ANGLE_MEASURE_WITH_UNIT	PAMWU
PLANE_ANGLE_UNIT	PLANUN
POINT	POINT
POINT_ON_CURVE	PNONCR
POINT_ON_SURFACE	PNONSR
POINT_REPLICA	PNTRPL
POLY_LOOP	PLYLP
POLYLINE	PLYLN
PRODUCT	PRDCT
PRODUCT_CATEGORY	PRDCTG
PRODUCT_CATEGORY_RELATIONSHIP	PRCTRL
PRODUCT_CONTEXT	PRDCNT
PRODUCT_DEFINITION	PRDDFN
PRODUCT_DEFINITION_CONTEXT	PRDFCN
PRODUCT_DEFINITION_FORMATION	PRDFFR
PRODUCT_DEFINITION_RELATIONSHIP	PRDFRL
PRODUCT_DEFINITION_SHAPE	PRDFSH
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

Entity name	Short name
REPRESENTATION_CONTEXT	RPRCNT
REPRESENTATION_ITEM	RPRITM
REPRESENTATION_MAP	RPRMP
REPRESENTATION_RELATIONSHIP	RPRRLT
ROLE_ASSOCIATION	RLASS
SEAM_CURVE	SMCRV
SERIAL_NUMBERED_EFFECTIVITY	SRNMEF
SHAPE_ASPECT	SHPASP
SHAPE_DEFINITION_REPRESENTATION	SHDFRP
SHAPE_REPRESENTATION	SHPRPR
SI_UNIT	SUNT
SOLID_ANGLE_MEASURE_WITH_UNIT	SAMWU
SOLID_ANGLE_UNIT	SLANUN
SOLID_MODEL	SLDMDL
SPHERICAL_SURFACE	SPHSRF
SUBFACE	SBFC
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

Entity name	Short name
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)

Implementation method specific requirements

The implementation method defines what types of exchange behaviour 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 ISO 10303-21 and in the AIM defined in annex A of this part of ISO 10303. The header of the exchange structure shall identify use of this part of ISO 10303 by the schema name 'ship moulded form 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 the subclauses of this annex.

This annex is in the form of a questionnaire. This questionnaire is intended to be filled out by the implementer and may be used in preparation for conformance testing by a testing laboratory. The completed PICS proforma is referred to as a PICS.

A number of options are identified in this standard for possible use by conforming implementations. Some of these options may be dynamically (run-time) selected for use/non-use, for instance, OPTIONAL attributes of an entity. Others shall be statically (configuration-time) selected for use/non-use, such as a particular style of geometry as defined in a conformance class.

Questions:

1. For simplicity of reference, an identifier for the product or system with which the tested STEP implementation is packaged in and/or procured by is required.

Product/system identifier (or name): _____

2. There are ten classes defined in this international standard. Each class specifies a subset of ISO 10303-216 AIM constructs. These classes are detailed in 6 of this document. Conformance to this part of ISO 10303 requires conformance to at least one of the primary conformance classes 1 through 4.

Claimed classes of conformance (functionality) - circle choices:

Class 1: Support for exchange of hydrostatic data;
Class 2: Support for exchange of basic moulded form geometry as an offset table;
Class 3: Support for exchange of Class 2 plus wireframe shape representations;
Class 4: Support for exchange of Class 3 plus surface shape representations.
Class 5: Support for exchange of Class 4 plus hull applicability.

3. Conformance to this international standard may be realized in one or more of several different implementation methods. The implementation methods define what types of exchange behavior are required with respect to this international standard.

Claimed implementation forms - circle choices:

exchange structure (ISO 10303-21);

exchange structure (ISO 10303-22);

exchange structure (ISO 10303-28).

4. If the exchange structure used is ISO 10303-28, which one?

ISO 10303-28 exchange structure? : _____

5. If the implementation receives data, which does not comply with the requirements in this international standard for the selected conformance class(es), or with the requirements of the 20's series of Parts for the selected implementation method, it shall execute a default response. A default response shall be statically set.

Default Response: _____

6. A conforming implementation shall maintain the static options selected throughout subsequent dynamic assessment (testing) without requiring modification. In a user environment, a conforming implementation shall permanently maintain the provision of selected static options, or it shall provide users discretionary control over the changing and setting of the static options, or both (depending on the option).

Does the IUT provide some user discretion over the changing and setting of static options?
Yes or No

7. If yes, which ones?

(a)Conformance class(es): _____

(b) Default Response: _____

8. A statement of conformance shall include identification of at least one party deeming conformance for the implementation.

Evaluator(s) (tester/certifier/accrediter):_____

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(216) 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 ship_moulded_form_schema in an open system, the object identifier

{iso standard 10303 part(216) version(1) object(1) ship-moulded-form-schema(1)}

is assigned to ship_moulded_form_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.

Annex F
(informative)

Application activity model

The application activity model (AAM) is provided as an aid to 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. Activities and data flows that are out of scope are marked with an asterisk.

The AAM covers activities which go beyond the subject of this application protocol. The diagrams use a modified IDEF0 notation [9]. Figure F.1 gives the basic notation. Each activity may be decomposed to provide more detail. If an activity has been decomposed, a separate figure is included.

As with any IDEF0 model, the application activity model is dependent on a particular viewpoint and purpose. The viewpoint of the application activity model is from a manufacturing engineer. The purpose of the application activity model is to clarify the context and scope of this application protocol.

This is an activity model of life cycle activities across all shipbuilding. There are several activity diagrams that have all activities out of scope but they are important in illustrating how the manufacture of a part process was developed and how the in-scope requirements were derived

NOTE The viewpoint of the application activity model is of the global ship development process. This activity model identifies the life cycle activities across all shipbuilding APs. Activities related to the shipbuilding lifecycle that are not expanded in this activity model are detailed in other shipbuilding application protocols.

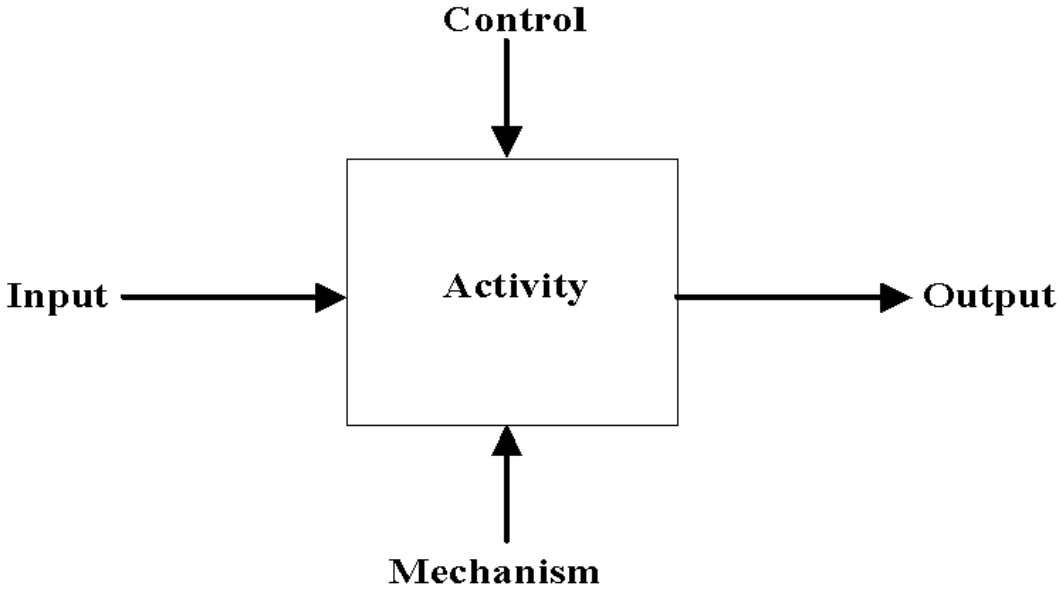


Figure F.1 — IDEF0 Basic notation

F. 1 Application activity model definitions and abbreviations

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 given in this annex do not supersede the definitions given in the main body of the text.

F.1.1 approved design: the release state of the design for production and inspection

F.1.2 arrangements*: the ship's compartments and spaces

NOTE Any description of arrangements will include associated definitions of purpose for the compartment or space.

F.1.3 assemble ship*: the production stage of the ship that assembles the modular units and the additional parts

NOTE The result is an assembled ship, that still has to be tested.

F.1.4 availability, reliability and maintainability information*: the information required for installation and planned maintenance for all components on the ship

F.1.5 basic hull parameters: principal dimensions and proportions based on estimations from historical data or from preliminary design development

F.1.6 budget*: the monetary constraint on the design, construction, and maintenance of the ship

F.1.7 calculate CFD*: calculation of fluid flow around the hull

NOTE The results contribute to the hydrodynamics and powering requirements calculations.

F.1.8 calculate cost of ship*: the description of creating the negotiating documents based on technical product data and the calculation of their estimated manufacturing cost

NOTE The results of this activity may contain sale price documents, financing support plan and documents describing funding and possible loans.

F.1.9 calculate hydrostatic properties: calculate the theoretical floating portion of the ship based on the ship design parameters

F.1.10 certificates*: documents, licenses, and permits issued for the completed ship

F.1.11 check design against rules and regulations: review of the design to assure that national and international classification requirements are met

NOTE 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.1.12 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 the ships' life cycle

F.1.13 complete and approve design of machinery*: the selection, arrangement, and approval of all machinery for the ship

F.1.14 complete and approve design of outfitting and distribution systems*: selection and approval of the necessary outfitting equipment

NOTE The selection is based mainly on former designs and in accordance with the ship's specified requirements. It also contains the layout of the different types of distribution systems such as piping, electrical, and HVAC.

F.1.15 complete and approve design of ship structure*: certify the final structural design of the ship before construction begins

F.1.16 complete and approve ship design: the production, validation, and certification of ship design product data, required rules and regulations, documents, and the classification drawings using the preliminary design

NOTE The result of this activity is the approved design and the production and delivery schedule.

F.1.17 conduct acceptance trials: the final trials for delivery the ship

F.1.18 conduct contractor sea trials: the sea trials to test if the built ship meets the contract requirements

F.1.19 completed sectional area curve: the curve created by representing the area of each cross-section, as a point on a curve, at specified intervals through the hull for the length of the ship.l(see Figure 43)

NOTE A completed sectional area curve is used to assess the suitability of the hull moulded form.

F.1.20 consultants: organizations and agencies that provide specific services to shipyards, ship owners and classification societies during the ship life-cycle

F.1.21 contract: the legal agreement that documents and authorizes the requisition for the ship

NOTE The contract is used as a constraint in subsequent activities such as final design and approval and production

F.1.22 cost*: the calculated monetary outlay for the ship based on the expenditure for material, labour, and overhead

F.1.23 create preliminary design: the early design stage for a ship taking into consideration the classification rules, national and international demands, shipyard constraints, and owner requirements

F.1.24 create preliminary general arrangements*:
produce the early compartmentation plans from the preliminary hull form definition

F.1.25 create preliminary hull form: the early hull form design stage for a ship

NOTE The first step of designing a ship by using parent ships main dimensions and form parameters. One or more preliminary hull forms will be generated.

F.1.26 create preliminary machinery design*: the early selection and arrangement of main machinery for the ship

NOTE It includes the prime propulsion system, shaft system, power systems and cargo handling equipment.

F.1.27 create preliminary outfitting design*: produce the early design for the ship's outfitting equipment

NOTE It includes distributed systems, such as piping and electrical systems.

F.1.28 create preliminary structure design*: produce the early design of the steel structure and the arrangement of the primary structural members

F.1.29 decide post-sales & maintenance support*: develop a proposal for the maintenance of the ship

NOTE This is part of the bid package documentation and includes the post sales support.

F.1.30 decommission and disassemble*: the phase of the ship’s life cycle during which the ship is taken out of service or dismantled

F.1.31 design schedule: data that controls the time span from the design phase to production

F.1.32 details of flow: this is the results of the Computational Fluid Dynamics (CFD) analysis, which models the flow of fluid along the hull. This is used to estimate resistance of the hull through the water and contributes to estimating the powering requirements for the ship

F.1.33 distribution and outfitting design*: the design of the distribution systems and the outfitting

EXAMPLE HVAC, electrical, and piping systems are types of distribution. Doors, ladders, pumps, motors, and control panels are types of outfitting.

F.1.34 do parametric variations: evaluate alternative hull design solutions created by varying hull moulded form parameters

NOTE Changing the speed, length, or beam of the ship, will result in varying hull forms for the ship.

F.1.35 equipment certificates: the certificates issued by the Classification Society on completing the equipment items which will be assembled to create the final product

F.1.36 estimate form parameters: establish preliminary hull form parameters from historical parent ship data and estimated main dimensions

F.1.37 estimate hydrodynamics and powering*: calculations approximating hydrodynamic properties data.for the preliminary hull form

EXAMPLE Types of estimates are resistance, propulsion, seakeeping and maneuverability.

F.1.38 estimate main dimensions and parameters: parameters and measurements that satisfy the clients usage requirements

EXAMPLE Speed, loading, mission, and capacities are types of client reqirements

F.1.39 estimate manoeuvrability*: approximation of the manoeuvrability of the ship and comparison of the model testing results with design requirements

NOTE The proof of the ship's manoeuvrability will principally be given in practice or by model testing. Measuring rudder forces and rudder moments as well as the radius of the turning circle during model tests will be done either in circulating water channels or manoeuvring basins.

F.1.40 estimate resistance and powering*: calculations based on historical data for producing powering and resistance data for the initial preliminary design

F.1.41 estimate sea-keeping*: calculations for the theoretical behaviour of a ship in seaway

NOTE The resultant coefficients of the equations of motion may be obtained either by analytical or numerical methods. The natural periods of the ship will be calculated for the rolling, pitching, and heaving motions.

F.1.42 evaluate request and schedule bid: shipyard function to analyze and consider the inquiry from the ship owner for a new ship

F.1.43 fair hull: adjustments to the hull moulded form to correct any surface aberrations

F.1.44 feedback: the return of information from an activity

F.1.45 finalise and approve general arrangements*: the final design of space allocation and arrangement of compartments and accesses

F.1.46 finalise and approve hull form: achieve an authorized complete hull form with the aid of preliminary design

NOTE The result is an approved hull form design.

F.1.47 finalise and approve hydrodynamics and powering*: complete the hydrodynamic analysis for the final design

NOTE Includes all relevant hydrodynamic calculations such as resistance, propulsion, seakeeping and manoeuvrability.

F.1.48 finalise main dimensions and parameters: refinement of the model tests, analysis, and usage requirements

F.1.49 fore-body definition: the hull moulded form design of the fore-body of the ship

NOTE The fore-body is that portion of the ship that extends from the fore part of the mid-body to the stem of the ship.

F.1.50 general arrangements*: the design of space allocation and arrangement of compartments and accesses

NOTE The ship's general arrangement is described by a compartment and access drawing showing the location, the access, and the size of the different compartments.

F.1.51 generate initial aft-body definition: establish the primary hull moulded form of the aft-body of the ship from the main dimensions, the preliminary hull form, and performance parameters considering the number and size of propellers

NOTE The aft-body is that portion of the ship aft of the mid-body (parallel mid-body) or mid-ship (non-parallel mid-body).

F.1.52 generate initial deck definition: determine the position and shape of the main weather deck of the ship from the main dimensions, the preliminary hull form, and the performance parameters

F.1.53 generate initial fore-body definition: establish the primary hull moulded form of the fore-body of the ship from the main dimensions, the preliminary hull form, and the performance parameters

NOTE The fore-body is that portion of the ship forward of the mid-body (parallel mid-body) or mid-ship (non-parallel mid-body).

F.1.54 generate initial hull form definition: establish the primary hull moulded form shape based on the calculated main dimensions, the preliminary hull form, and the performance parameters

F.1.55 generate initial mid-body definition: establish the primary hull moulded form of the mid-body of the ship from the main dimensions, the preliminary hull form, and the performance parameters

F.1.56 historical data from previous designs: data held by the shipyard or model basin on previous ship designs

NOTE This data is used to estimate the hydrodynamics, powering requirements and sea-keeping.

F.1.57 hull form parameters: the block coefficient, prismatic coefficient and Froude number that are used during the preliminary design phase to estimate the hydrostatic and hydrodynamic properties of the ship

F.1.58 hull form sections: divisions of the hull moulded form between planar sections along the longitudinal axis of the ship

F.1.59 hull moulded form with deck definition: the shape of the hull moulded form with the enclosing watertight deck of the ship

F.1.60 hull moulded form: the surface that defines the shape of the ship hull

NOTE Includes the aft-body, mid-body, and fore-body hull shape definitions, but does not take into account the thickness of the material from which the hull is made.

F.1.61 hydrodynamics & powering results*: engineering design calculations and model basin test data

NOTE Containing resistance, propulsion, propeller performance, brake power, service speed, sea keeping and manoeuvrability data.

F.1.62 hydrostatics table: tabular data that describes the hydrostatic properties for the ship that result from calculations at the initial and final design stages

F.1.63 knowledge and experience: a companies previous history of performance, comprehension of ship information, understanding problem areas, and analysis ability related to a ship throughout its life cycle

F.1.64 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.1.65 list of required certificates*: certificates for the ship issued by the Classification Society, national, and international regulatory bodies

F.1.66 loading conditions*: quantities of cargo, ballast water, and consumables identified for each space or compartment that are used as a basis for design

EXAMPLE Types of consumables are food, fuel, and medical supplies.

F.1.67 loading and stability manual: a booklet placed on board the ship specifying the prescribed limits for loading and unloading the ship as related to strength and stability of the ship

F.1.68 machinery design*: design drawings, prototypes, and electronic models of the ship mechanical systems

NOTE An output from the final design process.

F.1.69 main dimension: the length, breadth, depth and draught of the ship

NOTE These are estimated during the preliminary hull moulded form design and then finalised during the main design phase.

F.1.70 machinery weights: estimated or calculated design weights for all machinery on board the ship

NOTE Result of several calculations and design activities

F.1.71 manoeuvring results*: the estimated zigzagging, planned moves, and turning results based on previous model basin tests or sea trials

F.1.72 manufacturing restrictions: constraints of the ship construction, fabrication, and design processes governed by available technology and shipyard facilities

F.1.73 material list*: the list of all raw materials needed to manufacture the ship

NOTE A result of the final design process.

F.1.74 mid-body definition: the hull moulded form design of the parallel body portion, that includes the mid-ship section design, of the ship that extends between the aft-body and the fore-body of the ship or the hull moulded form design of the mid-ship section design for ships that have no parallel body portion

EXAMPLE Oil tankers and dry cargo carriers have a parallel mid-body portion, but Naval ships seldom have a parallel mid-body.

F.1.75 model basin consultants: organizations that perform model basin testing to calculate hydrodynamics and powering data

F.1.76 model basin theory*: the formulas, along with experience, knowledge, and empirical data, used by the model basin consultants to calculate the hydrodynamics and powering information

F.1.77 modifications from machinery*: revisions or adjustments to the hydrodynamics and powering due to feedback from the preliminary machinery design

F.1.78 modifications to hull form: revisions or adjustments to the hull shape due to feedback from hydrodynamics and powering results and the final design process

F.1.79 modifications to hull form parameters: revisions or adjustments to the form parameters due to the result of performing parametric variations in the preliminary design stage

F.1.80 modifications to main dimensions: revisions or adjustments to the main dimensions of the hull that are made during the hull moulded form design process

F.1.81 modular units*: sub-sections of the ship complete with machinery and outfitting which will be assembled to create the final product

F.1.82 offer: a proposal from the shipyard to produce the requested ship

NOTE This is shipyard's data that results from the preliminary design process.

F.1.83 offer guidelines: the data necessary to make an unconditional offer to the ship owner

F.1.84 operate and maintain a ship*: procedures and documentation for running, servicing, and repairing the ship during its useful lifetime

F.1.85 operational information*: the document resulting from operating, maintaining, and surveying a ship to give information about a ship's condition

NOTE The document includes accumulated information recorded during the functional and serviceable phase of the ship used for maintenance and in the final scrapping stage.

F.1.86 owner: the organization that requests, orders, takes delivery of, and operates the ship

F.1.87 owner request with requirements: the requirements document submitted to the shipyard by the owner with the invitation to submit a bid

F.1.88 perform model tests*: conduct model testing in a towing tank or model basin

NOTE The results contribute to the hydrodynamics and powering design

F.1.89 perform ship lifecycle: the completed stages of a ship's existence that include conception, design, construction, operation, and disposal

F.1.90 place order*: to acknowledge and accept a bid proposal from a shipyard to produce a ship

NOTE From this a contract is awarded to the shipyard.

F.1.91 planned maintenance system*: data created during the final design process and used during the operation and maintenance of the ship

F.1.92 power requirements for engine*: hydrodynamics and propulsion data used in the selection of the main engine

F.1.93 pre layout: the first layout of the ship that is produced during the bid evaluation stage and is the basis for the preliminary design

F.1.94 predict brake power and service speed*: estimate the required braking power from the engine and the required power delivered to the propeller for determining the size of the main engine

F.1.95 predict propeller performance*: estimate propulsion requirements data to produce a preliminary propeller design

F.1.96 predict propulsion data*: calculate the propulsion requirements of the ship

NOTE An approximate definition of driving power can be made using the displacement equation for specific Froude numbers (see 4.2.21.4) in the preliminary design stage.

F.1.97 predict resistance*: estimate the wet-able surface area of the ship for seaway resistance

NOTE The resistance calculation uses historical data related to the hull moulded form geometry.

F.1.98 preliminary design: the design that leads to the submission of a bid proposal

F.1.99 preliminary general arrangements: the early definition of spaces and compartments for the ship

NOTE General arrangements are approximated or calulated as a result of the preliminary design process.

F.1.100 preliminary hull form: the early definition of the hull moulded form, as a result of the preliminary design process

NOTE This is used in the bid proposal documents, for preliminary compartment design, hydrodynamics, and powering calculations.

F.1.101 preliminary machinery design*: the early definition of the ship's mechanical systems

NOTE This is used early to estimate the noise, speed and vibration, and to estimate the machinery weights.

F.1.102 preliminary machinery, structure and outfitting design*: early design information fed back from the preliminary design for machinery, structure, and outfitting and furnishing

NOTE This assists in the preliminary general arrangements development.

F.1.103 preliminary outfitting design*: the early definition of the ship's outfitting and accommodation

NOTE This contributes to part of the preliminary design process.

F.1.104 preliminary structure design*: the early definition of the steel structure for the ship

NOTE This contributes to part of the preliminary design process.

F.1.105 prepare bid: the preparation of all documents and data necessary for submitting a proposal to the owner for building the ship

F.1.106 present offer*: submit all required bid proposal documents, for producing the requested ship, to the prospective owner

F.1.107 produce and approve reference documents: compose and assemble technical documentation authorized for the ship using production information

NOTE The output includes the loading and stability manual that is approved by the Classification Society.

F.1.108 produce and inspect a ship: build and perform quality assurance at each stage of construction or fabrication of the ship

NOTE Inspect means, the controlling of all activities throughout the whole production life cycle of a ship.

F.1.109 produce modular build units*: fabricate steel subsections and erected into large detached certified units that when assembled or attached will make up the completed ship

NOTE Their production is controlled by the schedule, contract, the approved design, any manufacturing restrictions, and the classification society. The results are the modular units, which are assembled into the ship.

F.1.110 produce steel sub-sections*: construct and join structural steel members that when assembled and certified make up a modular unit

NOTE Their production is controlled by the schedule, contract, the approved design, any manufacturing restrictions, and the classification society. The results are the steel sub-sections, which are assembled into modular units for completing the ship.

F.1.111 product component information*: the technical data for all parts and items that will be incorporated into the ship

NOTE These are taken into consideration when the preliminary designs are being made.

F.1.112 production and delivery schedule: the timetable to which the ship is to be manufactured and delivered

F.1.113 production information: data describing the manufacturing and construction details of the product

EXAMPLE Types of information are: dimensions, mechanical properties, materials, and workshop information.

F.1.114 propeller design*: the design of the propeller or propulsor as a result of the hydrodynamics and powering calculations

NOTE The design controls some of the ship's machinery design.

F.1.115 propeller theory*: knowledge based on experience, historical data, and research studies of ship propeller performance

F.1.116 quality assurance: the rules applied by an organization within the shipyard that has the task to audit the shipyard organization and applied processes in a manner such that the quality of the resulting product is assured

F.1.117 regular wave theory*: the knowledge and experience related to the motion response of a ship in waves of constant height and period

F.1.118 request a ship: the first action or contact of a ship owner when intending to order a ship

NOTE Having definite ideas regarding appearance and functionality of the ship, the owner expresses these ideas in an inquiry to the shipyard.

F.1.119 request for production changes: changes that are requested to the ship design as a result of production experience or difficulties with the realization of the ship design

F.1.120 resistance and shaft power*: the opposition to motion that results in hydrodynamics and powering estimates

NOTE Resistance and shaft power is a constraint on the creation of the preliminary hull form.

F.1.121 resistance theory*: the principles, theorems, and rules used to predict the resistance of the hull to forward motion in the sea

F.1.122 resources: the shipyard, classification society, and outside consultants

F.1.123 schedule: the plan for governing the timing of the production phases

NOTE Formed as a part of the final design process.

F.1.124 scrapping plan*: the document used to schedule the time and resources required to dismantle the ship

F.1.125 ship: a large waterborne vessel whose design, manufacture and lifecycle operation is governed by the principles of naval architecture and in accordance with international and classification society regulations

F.1.126 ship product model data: the product data of the ship accumulated throughout its lifecycle

NOTE Since scrapping is part of the lifecycle the ship is not an output, only the documented information and knowledge about the ship survives.

F.1.127 ship weight modifications*: revisions to ship weight due to the preliminary structure design

NOTE This is fed back to modify the preliminary hull form and revise the preliminary general arrangements.

F.1.128 shipyard: an organization that designs, builds, maintains, and repairs ships

F.1.129 short and long term responses*: an approximation of sea keeping ability with calculations based on regular waves of a specified height and period

NOTE The results are used to predict the maximum response in irregular waves over a long period of time.

F.1.130 specify ship: the production of precise specifications for a ship prior to a contract being placed

F.1.131 steel sub-sections*: sub-sections of the steel structure which are outfitted with the machinery and distribution systems before assembly

F.1.132 structural design*: design of the ship's foundation and framework

EXAMPLE The design includes the keel, hull, bulkheads, decks, superstructures, girders, stiffeners, etc..

F.1.133 technical documentation: the detailed information about material parts needed for producing the ship and system

NOTE In the case of maintenance the technical documentation of a system means part of the product description required to perform preventative maintenance, repair and failure of that system.

F.1.134 technical requirements: the owner's specifications that must be realized by the completed ship

F.1.135 test results*: the documented evaluations and conclusions for each inspection, check, or analysis against design specifications, rules, and regulations

NOTE As part of the ship's maintenance throughout its lifecycle, test results are the results of functional tests carried out after the execution of maintenance actions.

F.1.136 test ship: inspect and evaluate the actual ship against the design specifications, rules and regulations, and contract requirements

NOTE The structure is tested and sea trials are carried out. The test results are documented.

F.1.137 test structures: the steel structures are inspected and evaluated against the design specifications, rules and regulations, and contract requirements

NOTE The output is the test result documentation.

F.1.138 test systems*: the ship's systems including outfitting, machinery, and mission systems are inspected, put into service, and evaluated against the design specifications, rules and regulations, and contract requirements

NOTE The output is the test result documentation.

F.1.139 total resistance*: the counteracting forces on the hull due to forward motion

EXAMPLE Total resistance includes frictional resistance, wave resistance, eddy resistance, and air resistance.

F.1.140 towing tank model: a scale model of the ship used in a towing tank for estimating hydrodynamic properties

F.1.141 transportation need: a constraint which determines the specification for the ship construction

F.1.142 weights and centres of gravity*: the amount that things weigh and the centres of mass for those things

NOTE Weights and centres of gravity necessary for further calculations.

F.1.143 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 Figure F.2 through Figure F.15. The graphical form of the Application Activity Model is presented in the IDEF0 [9] activity modeling format. Activities and data flows that are out of scope are marked with asterisks.

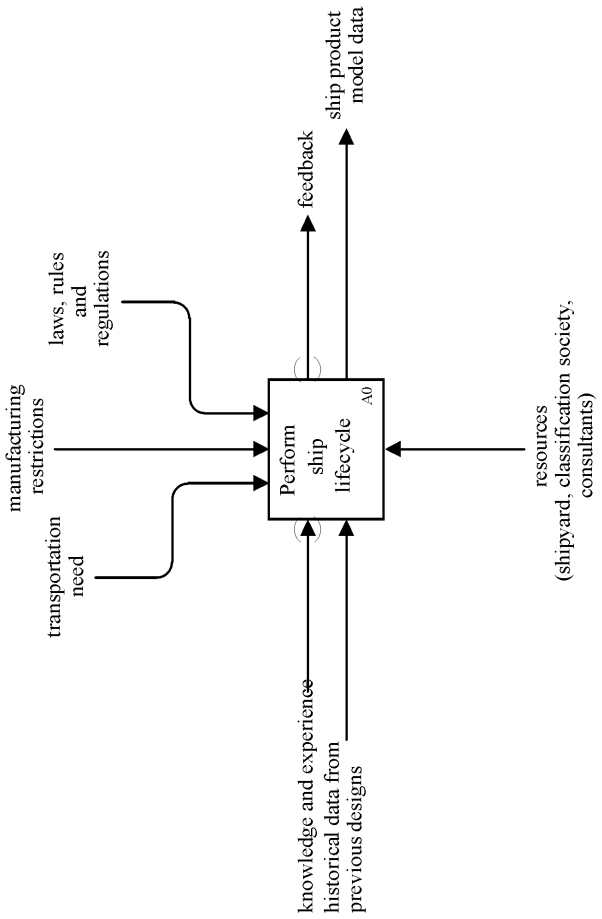


Figure F.2 — Node A0 - moulded form life cycle

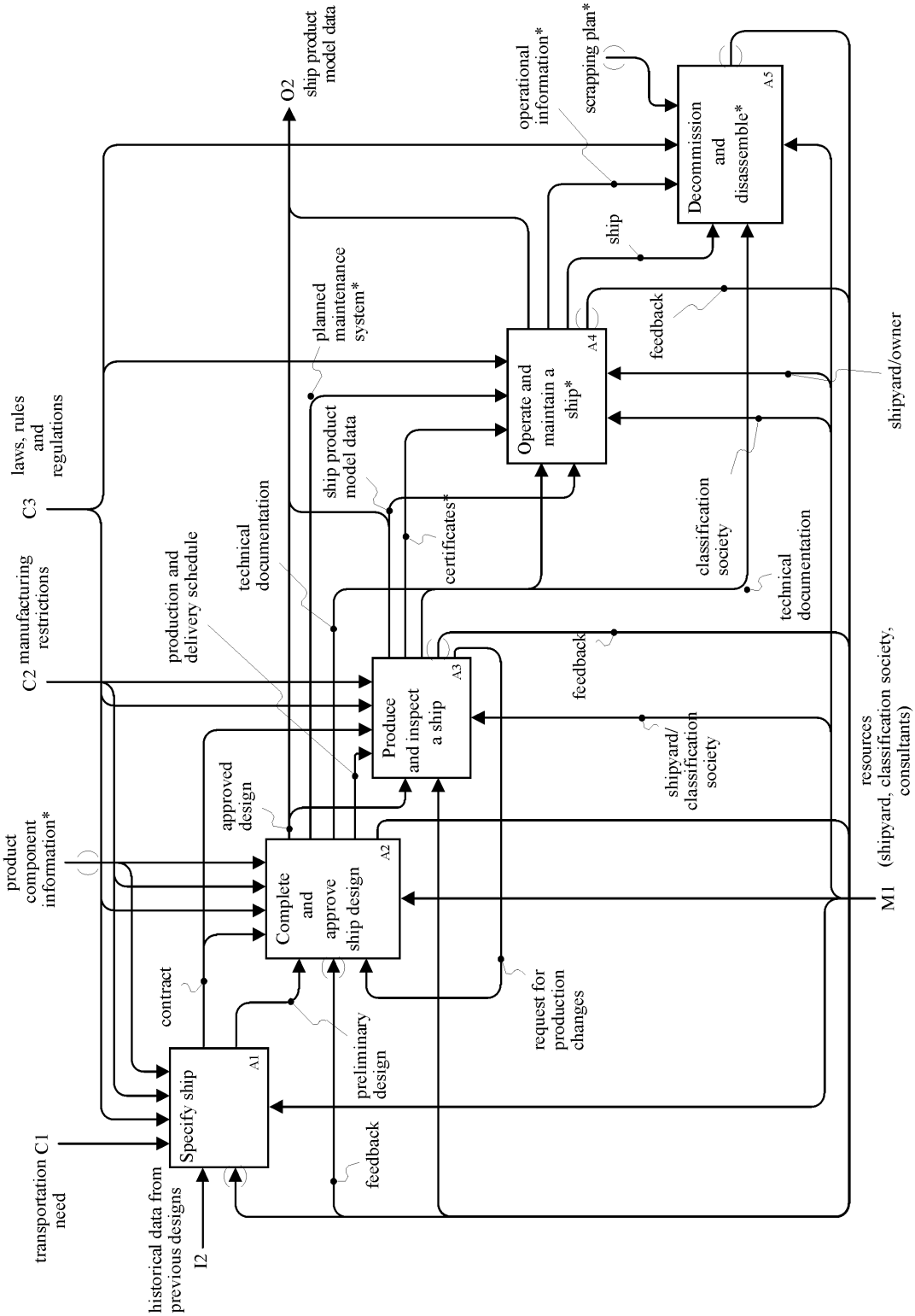


Figure F.3 — Node A0 - perform ship life cycle

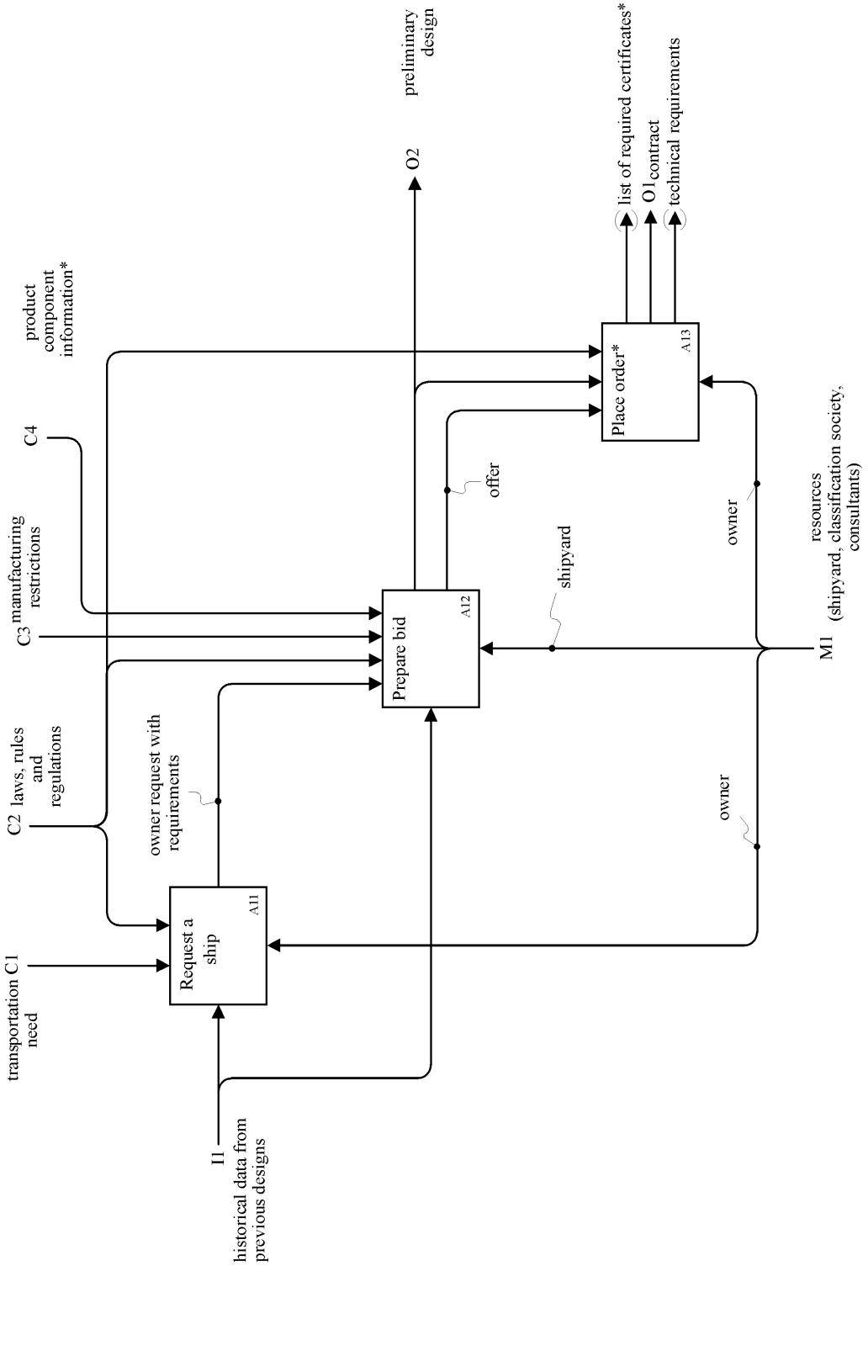


Figure F.4 — Node A1 - specify ship

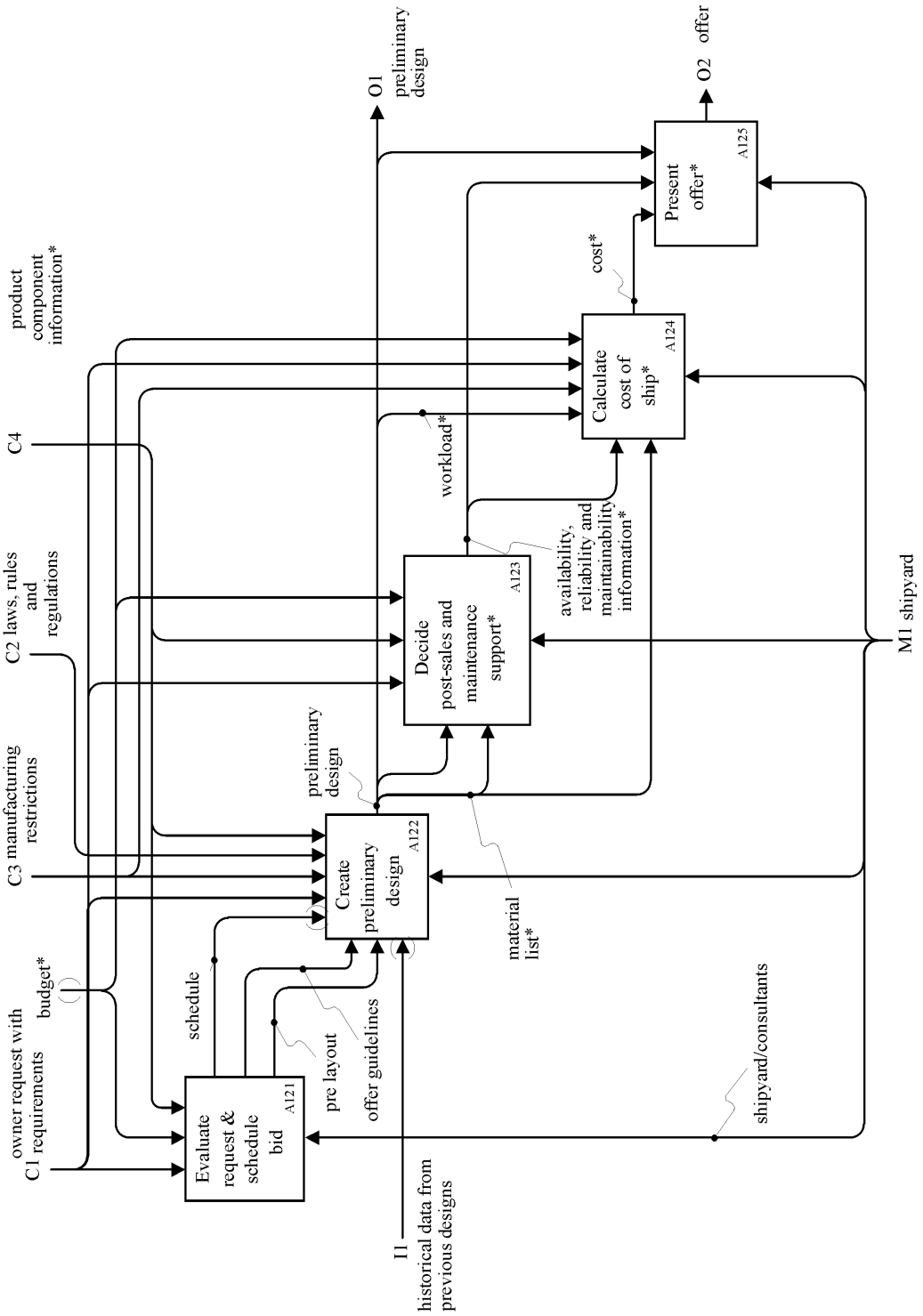


Figure F.5 — Node A12 - prepare bid

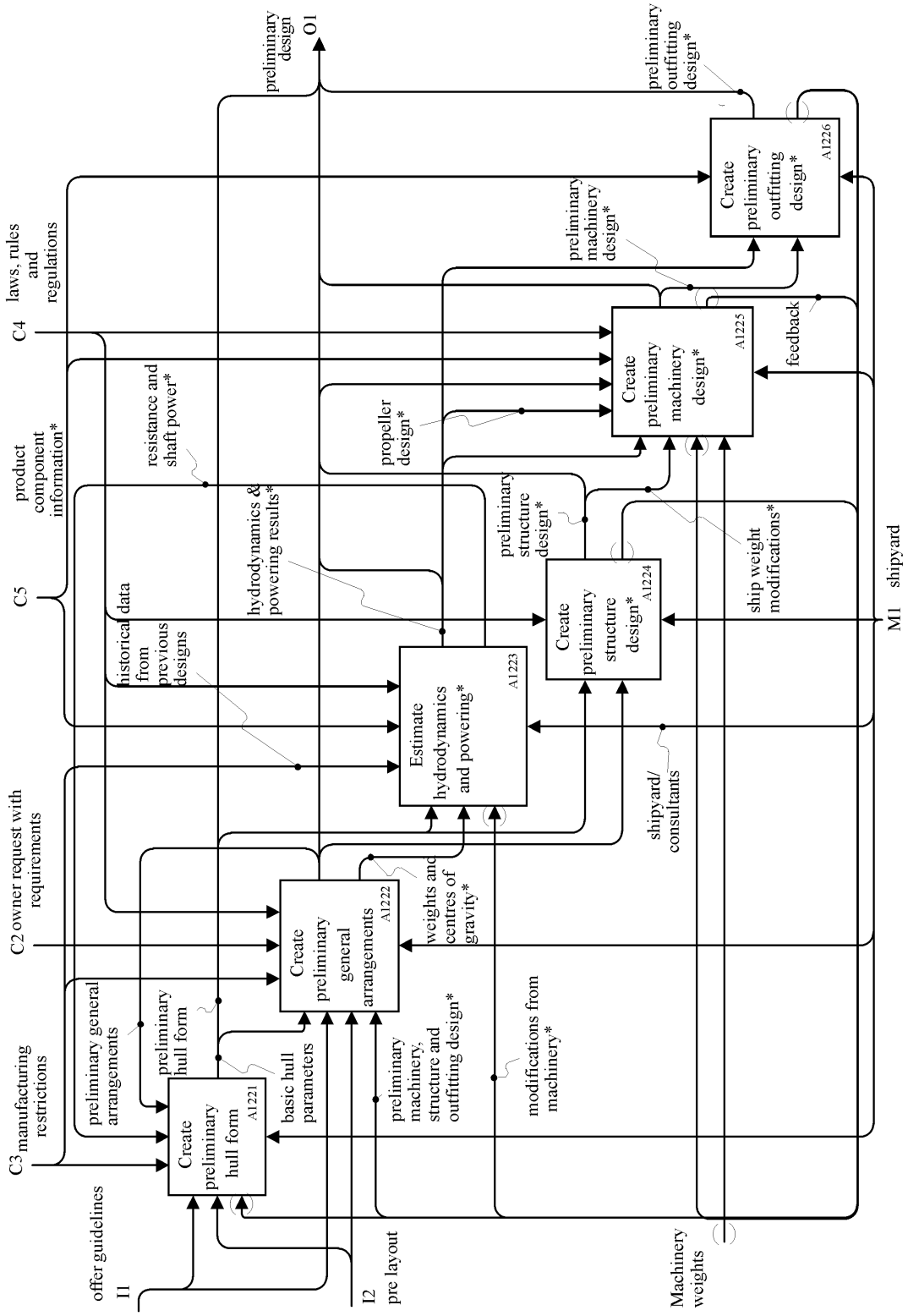


Figure F.6 — Node A122 - create preliminary design

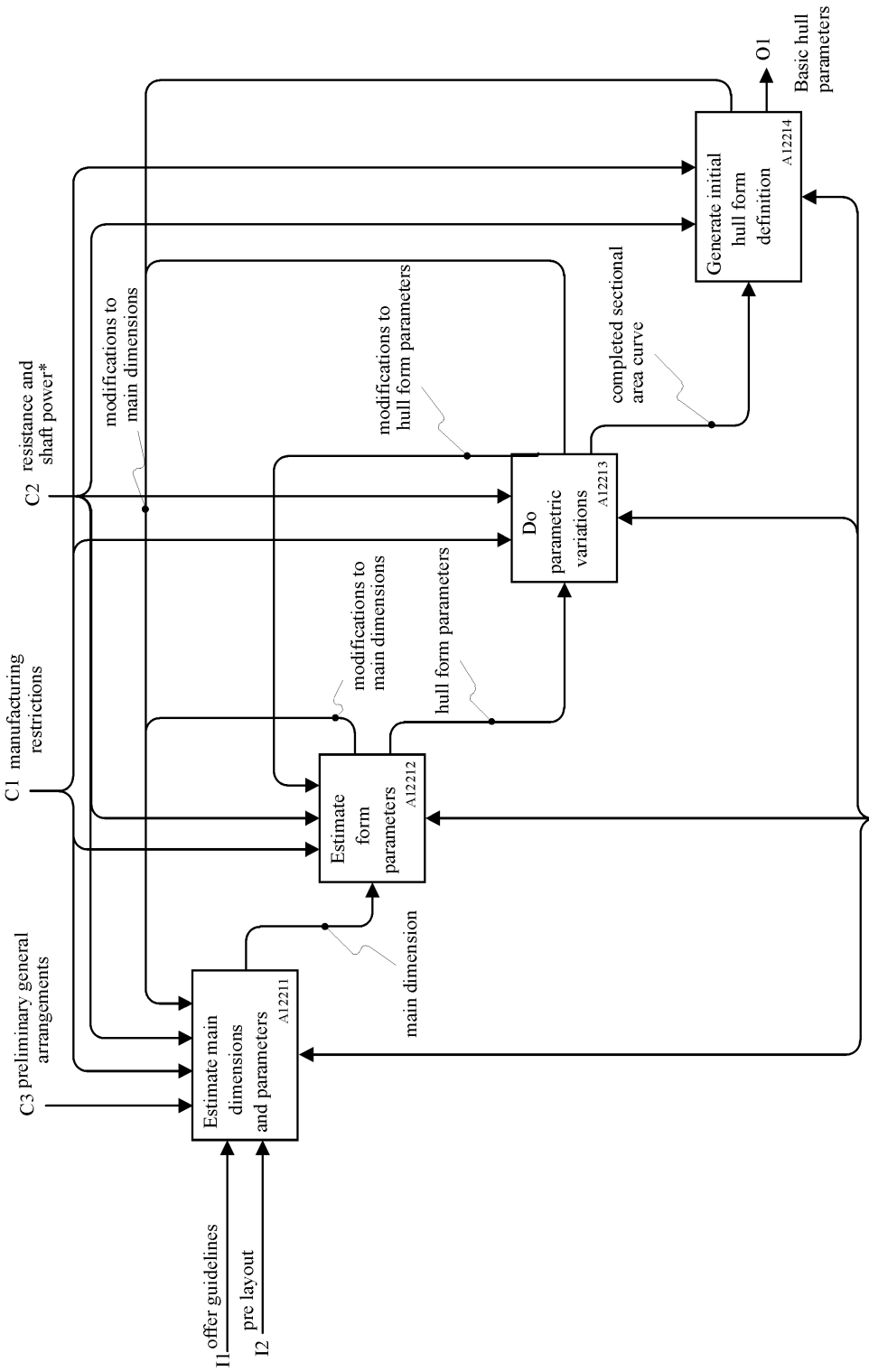


Figure F.7 — Node A1221 - create preliminary hull form

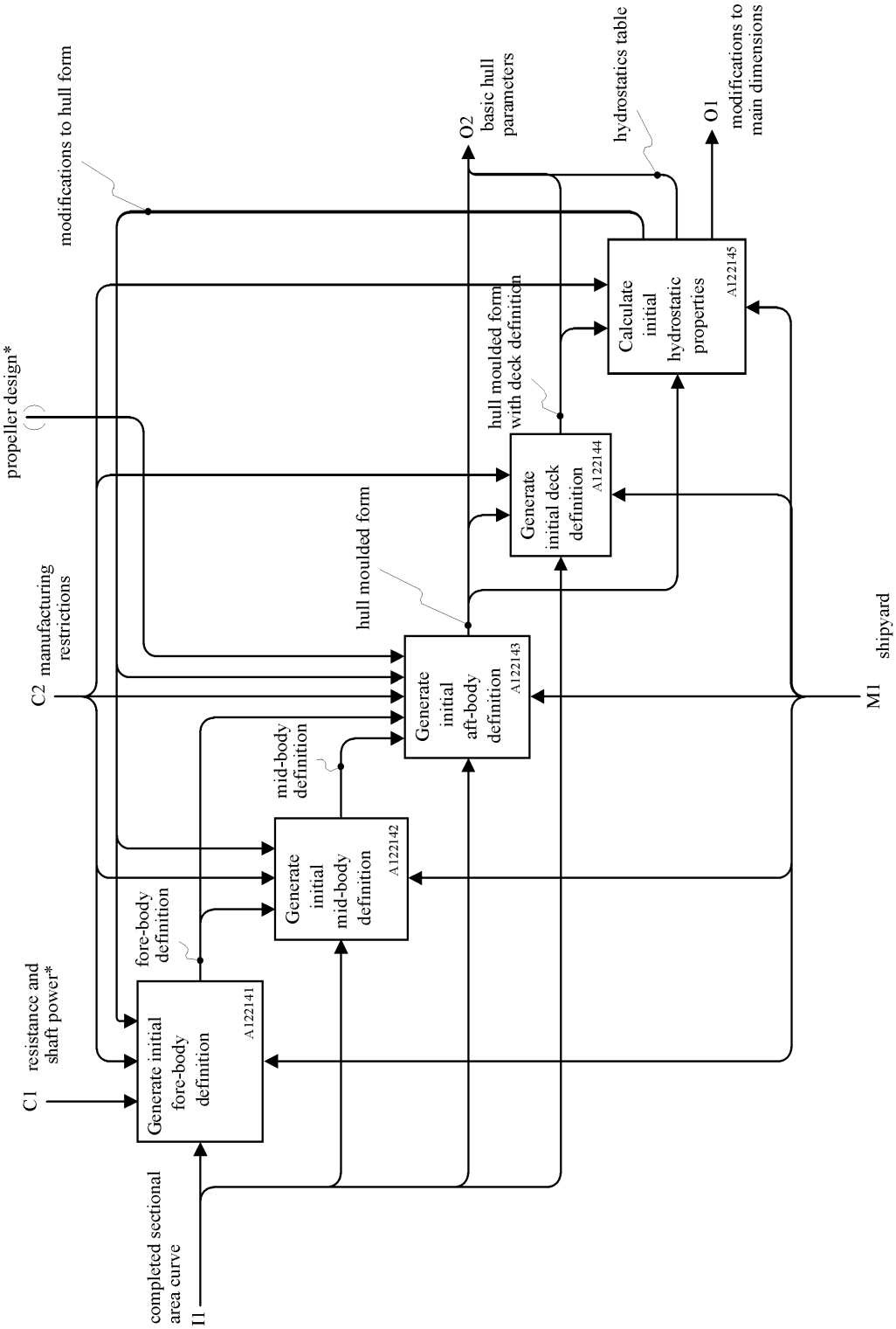


Figure F.8 — Node A12214 - generate initial hull form definition

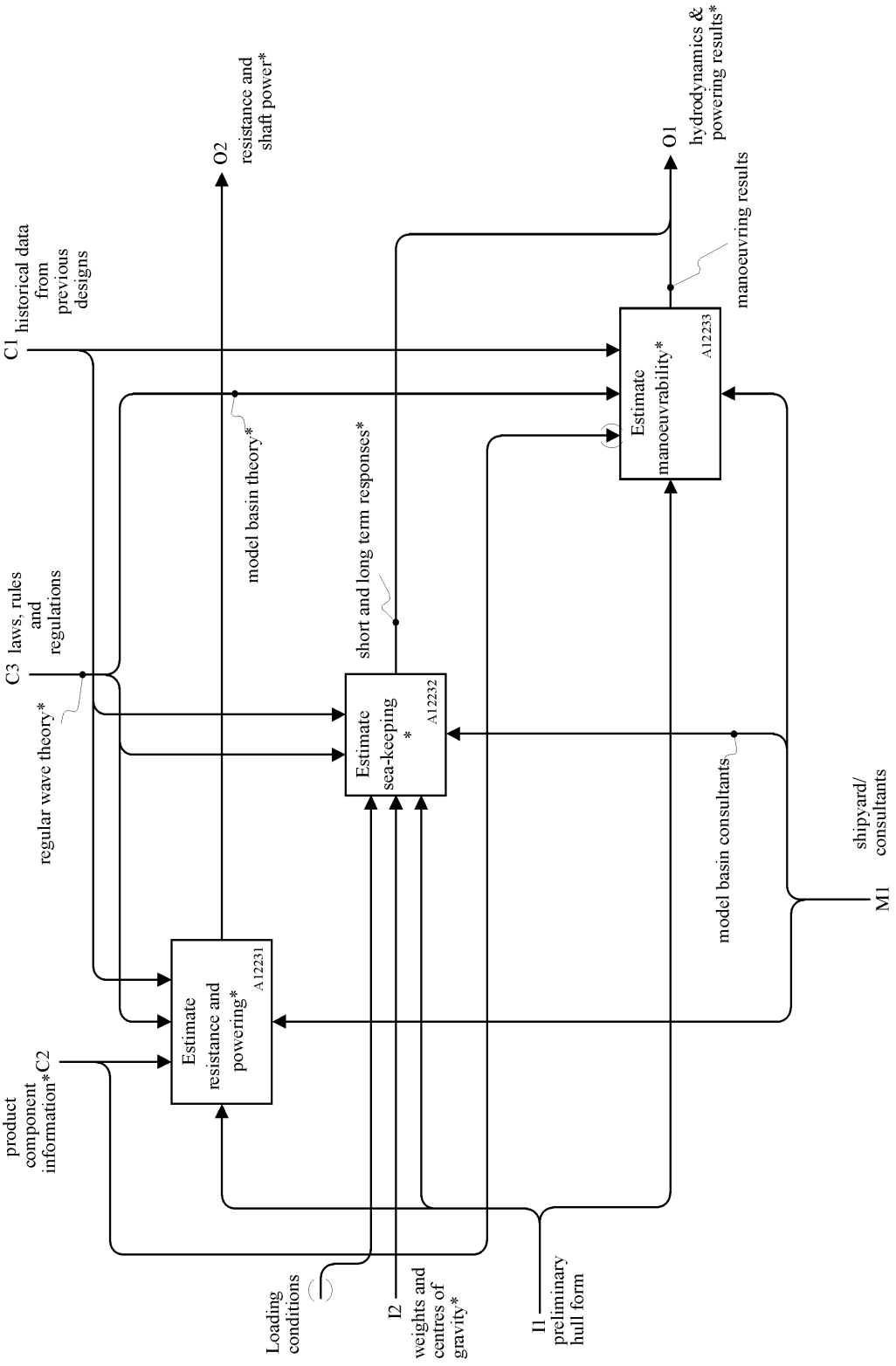


Figure F.9 — Node A1223 - estimate hydrodynamics and powering

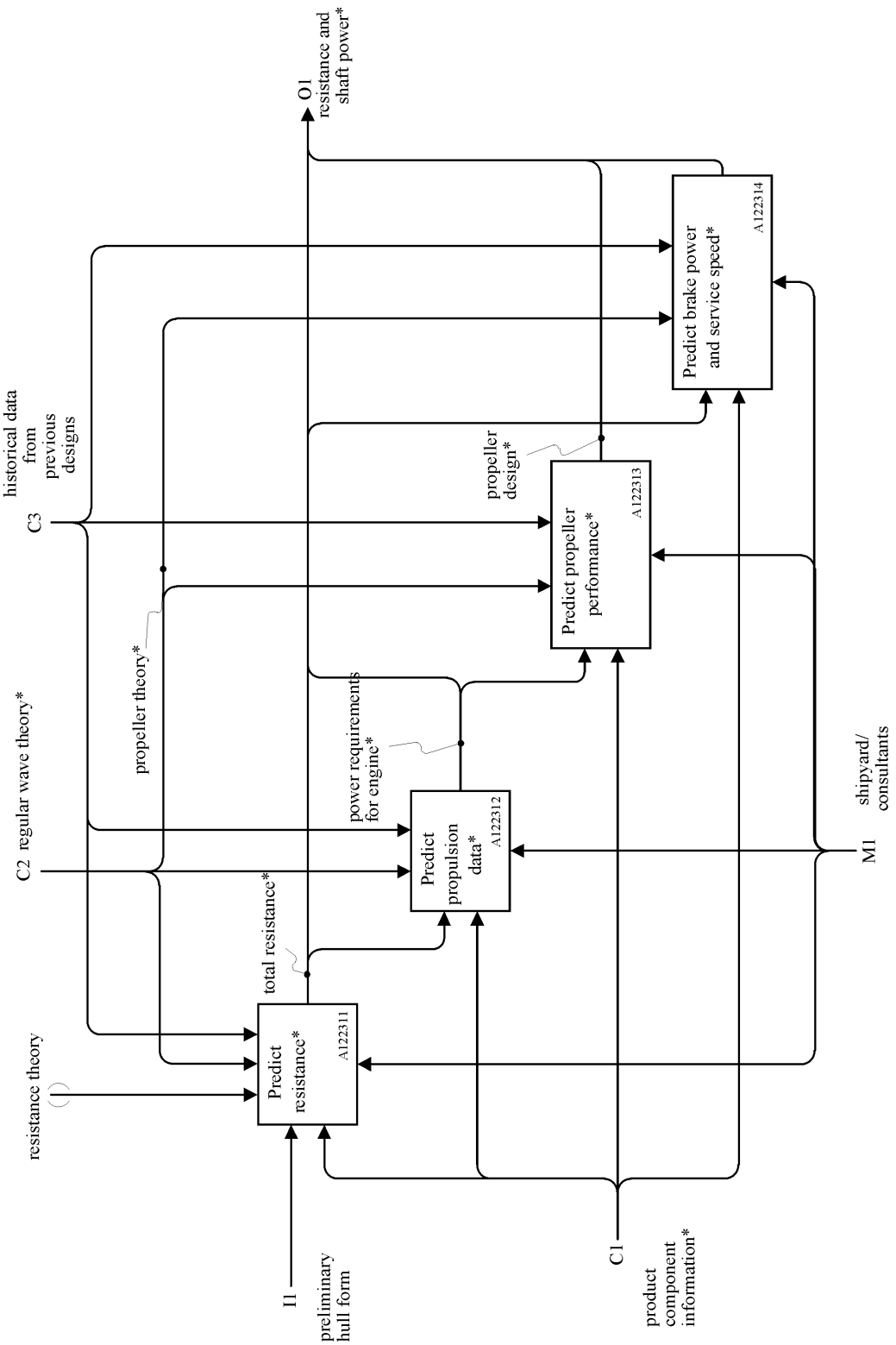


Figure F.10 — Node 12231 - estimate resistance and powering

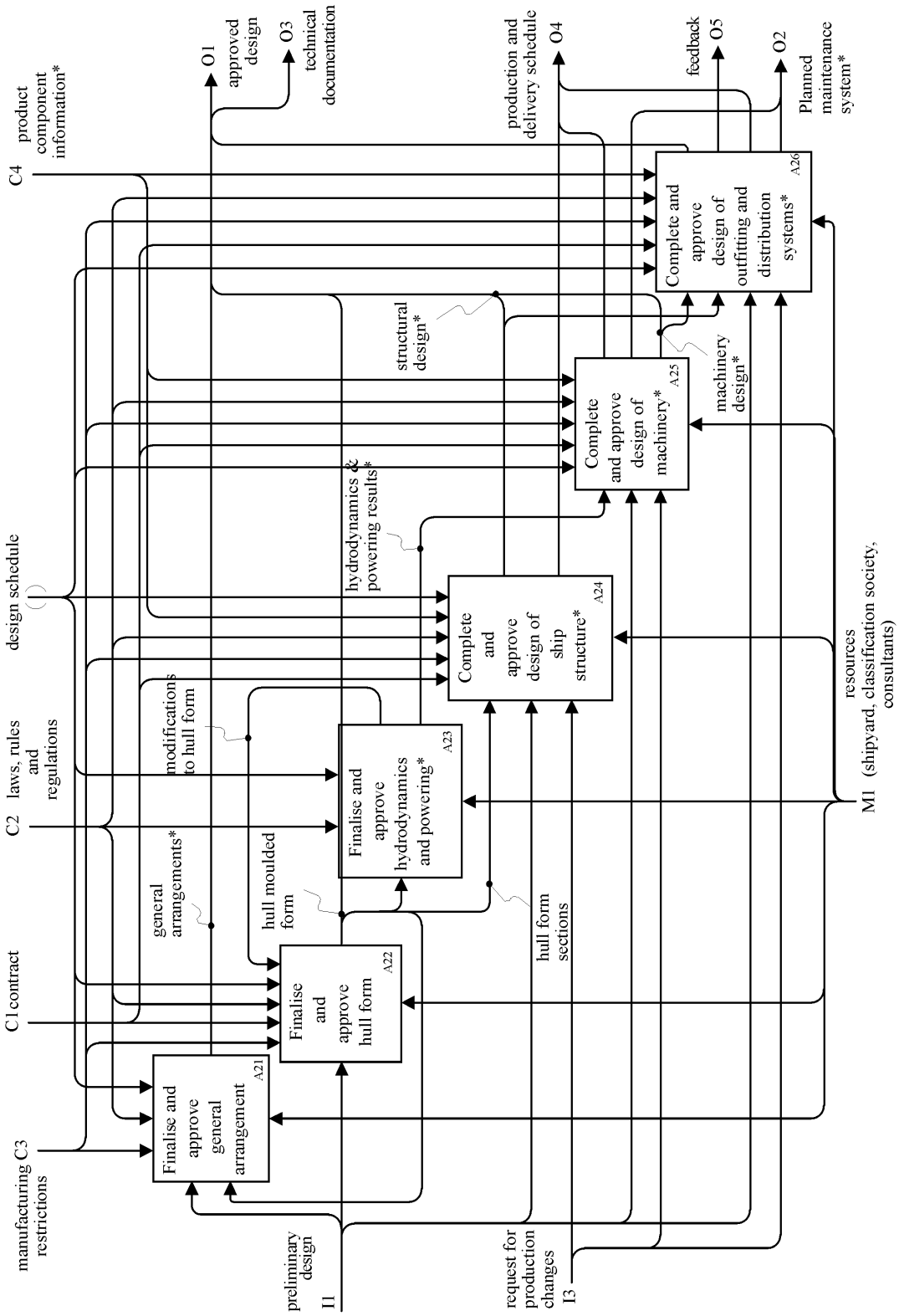


Figure F.11 — Node A2 - complete and approve ship design

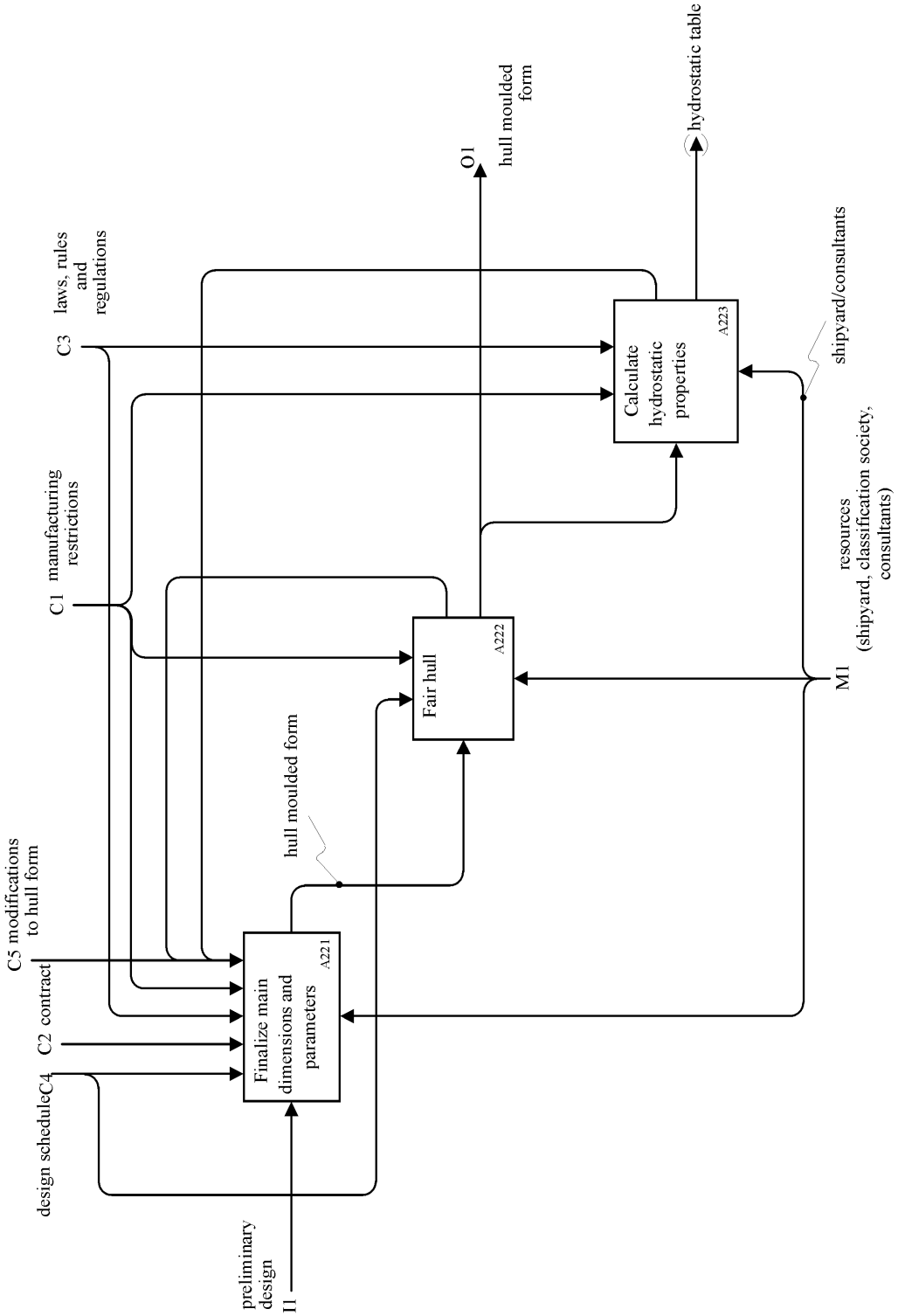


Figure F.12 — Node A22 - finalize and approve hull form

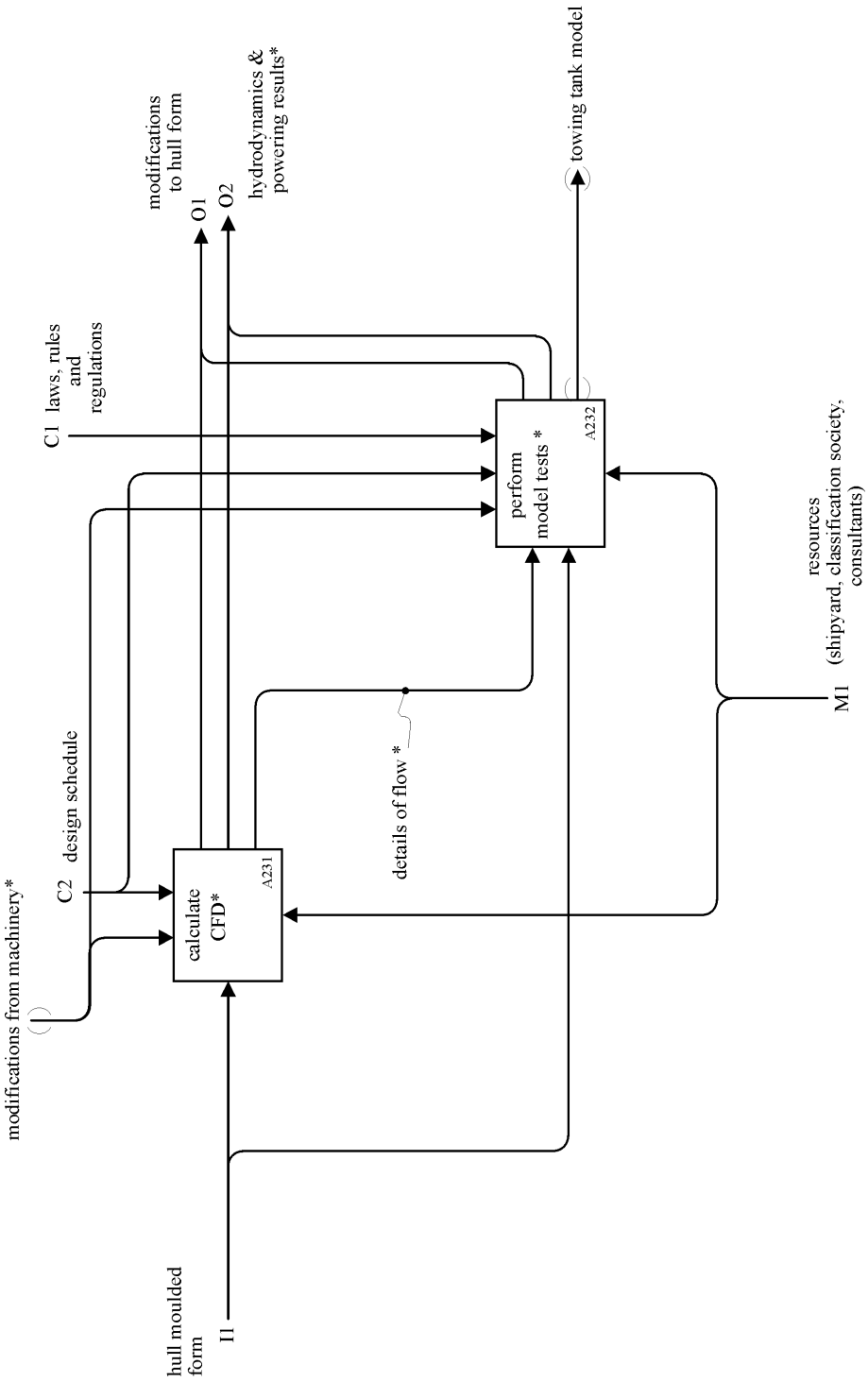


Figure F.13 — Node A23 - finalize and approve hydrodynamics and powering

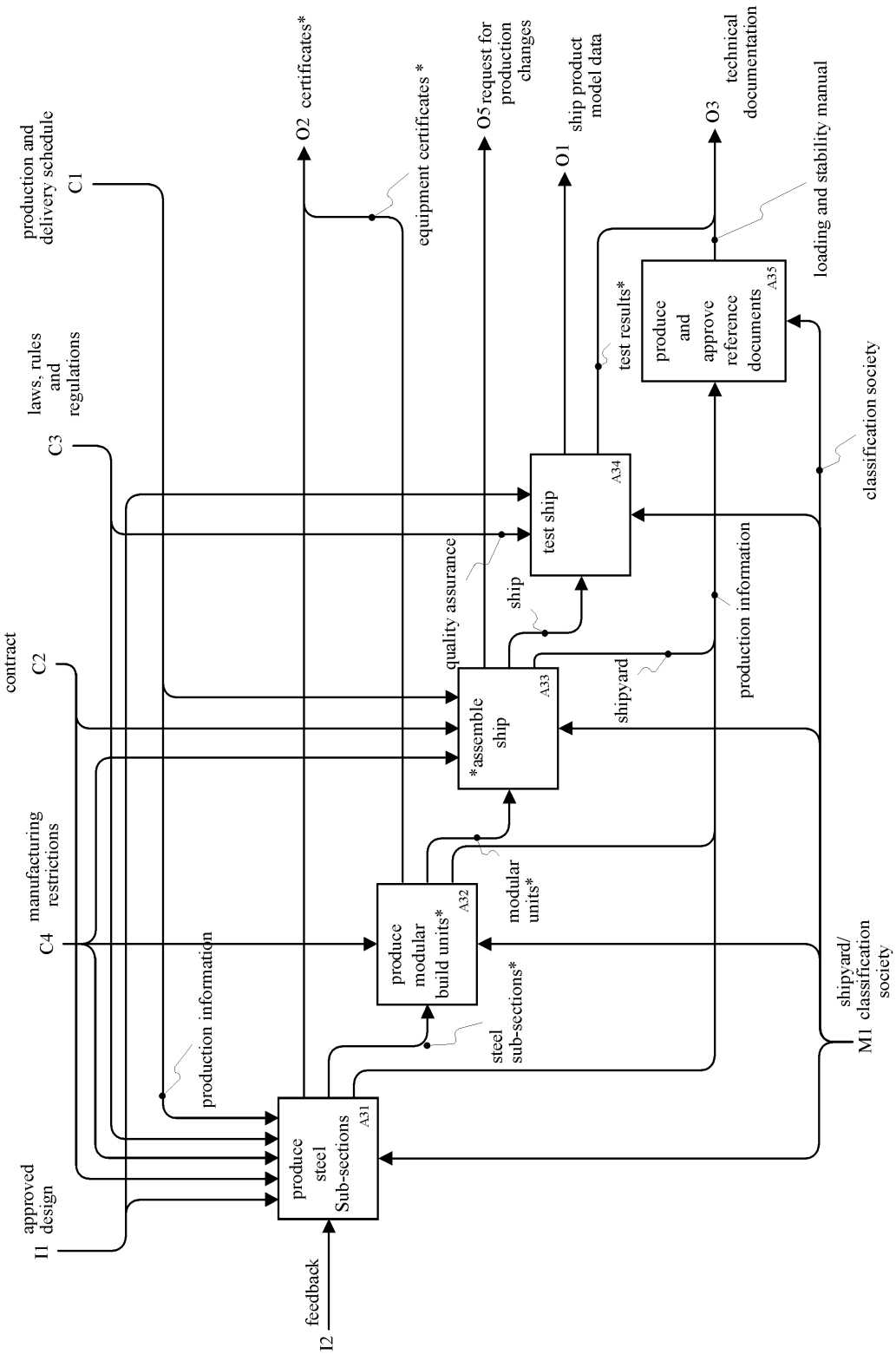


Figure F.14 — Node A3 - produce and inspect a ship

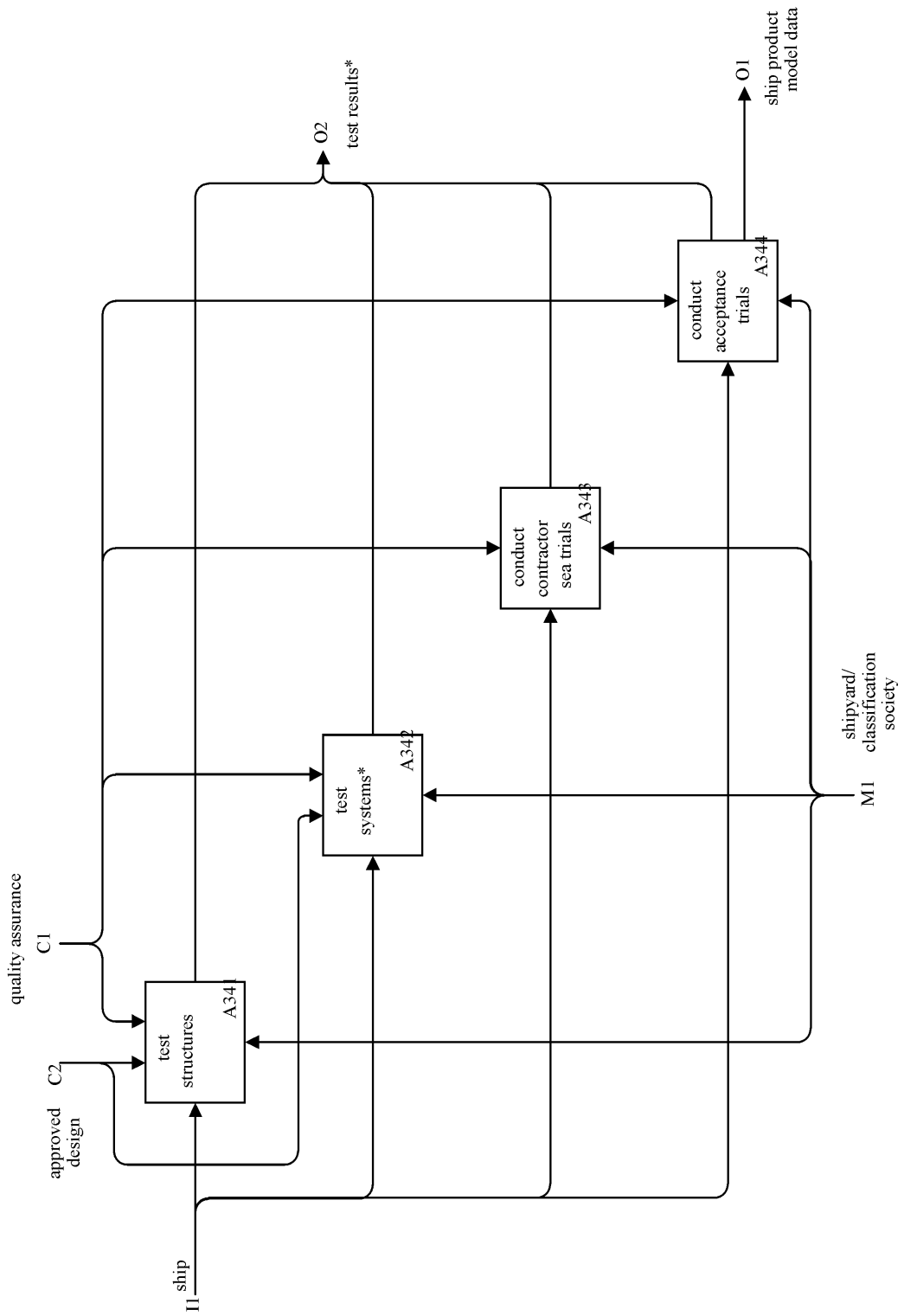


Figure F.15 — Node A34 - test ship

Annex G
(informative)

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 D of ISO 10303-11.

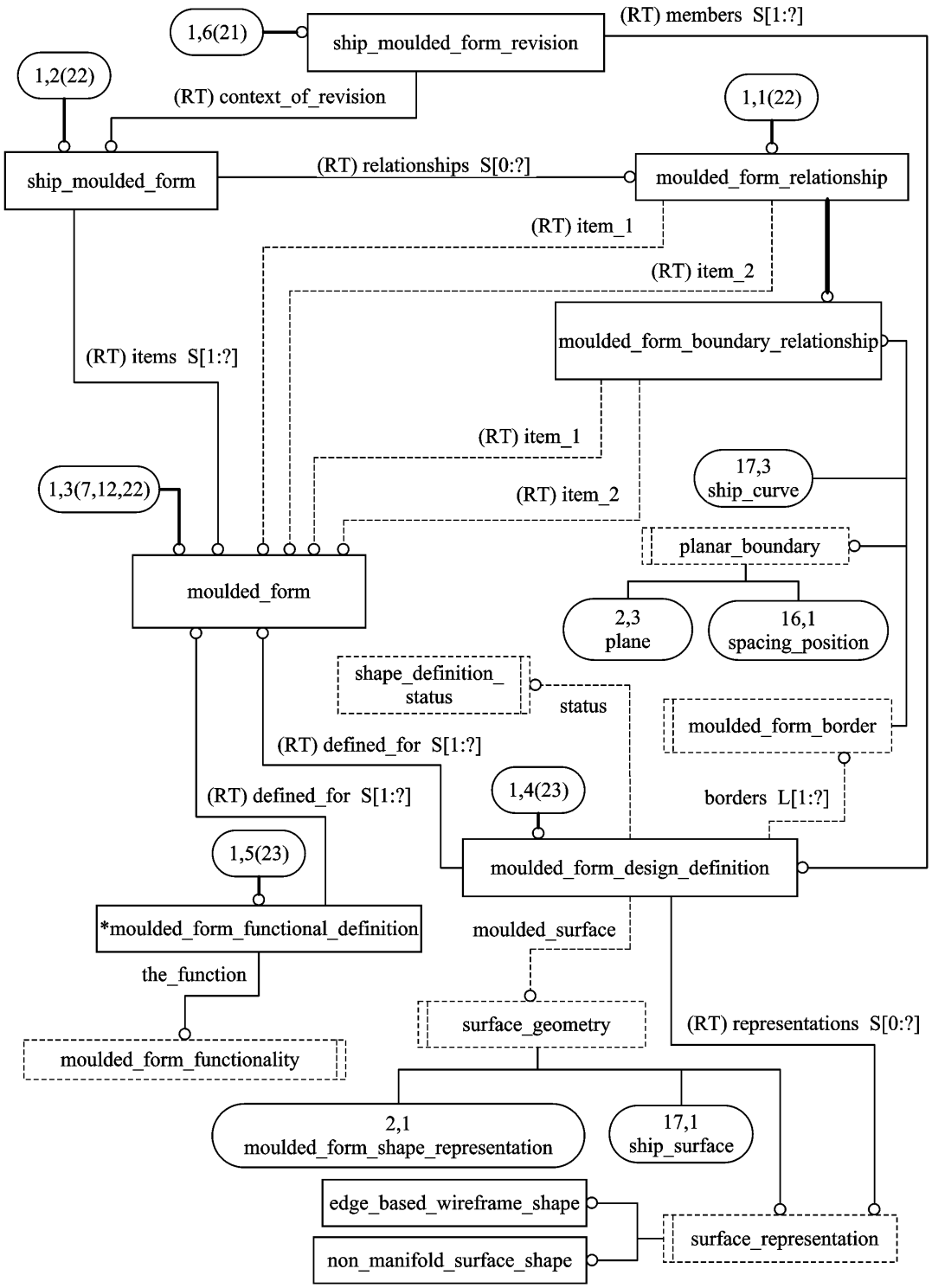


Figure G.1 — ARM EXPRESS-G diagram 1 of 25

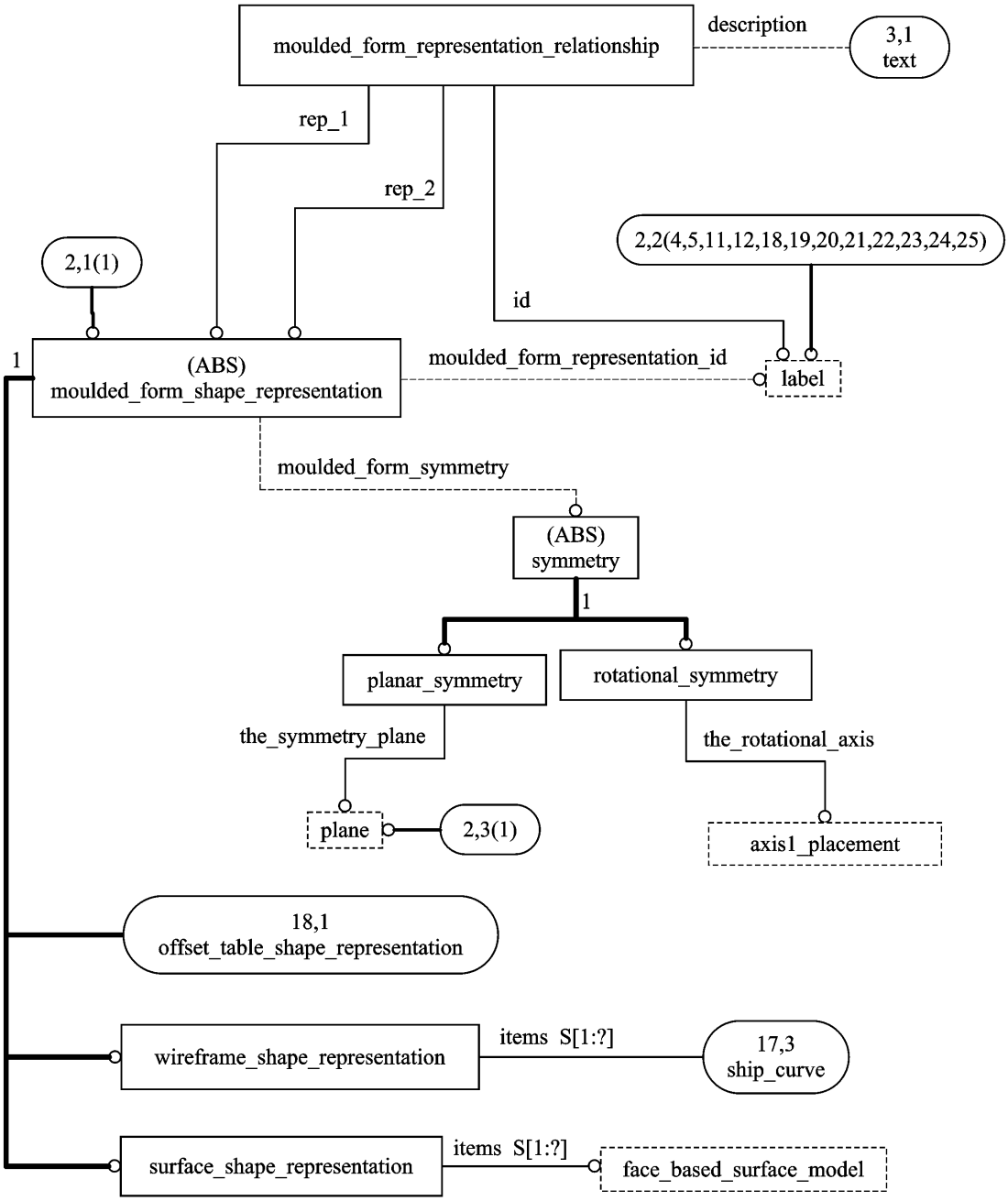


Figure G.2 — ARM EXPRESS-G diagram 2 of 25

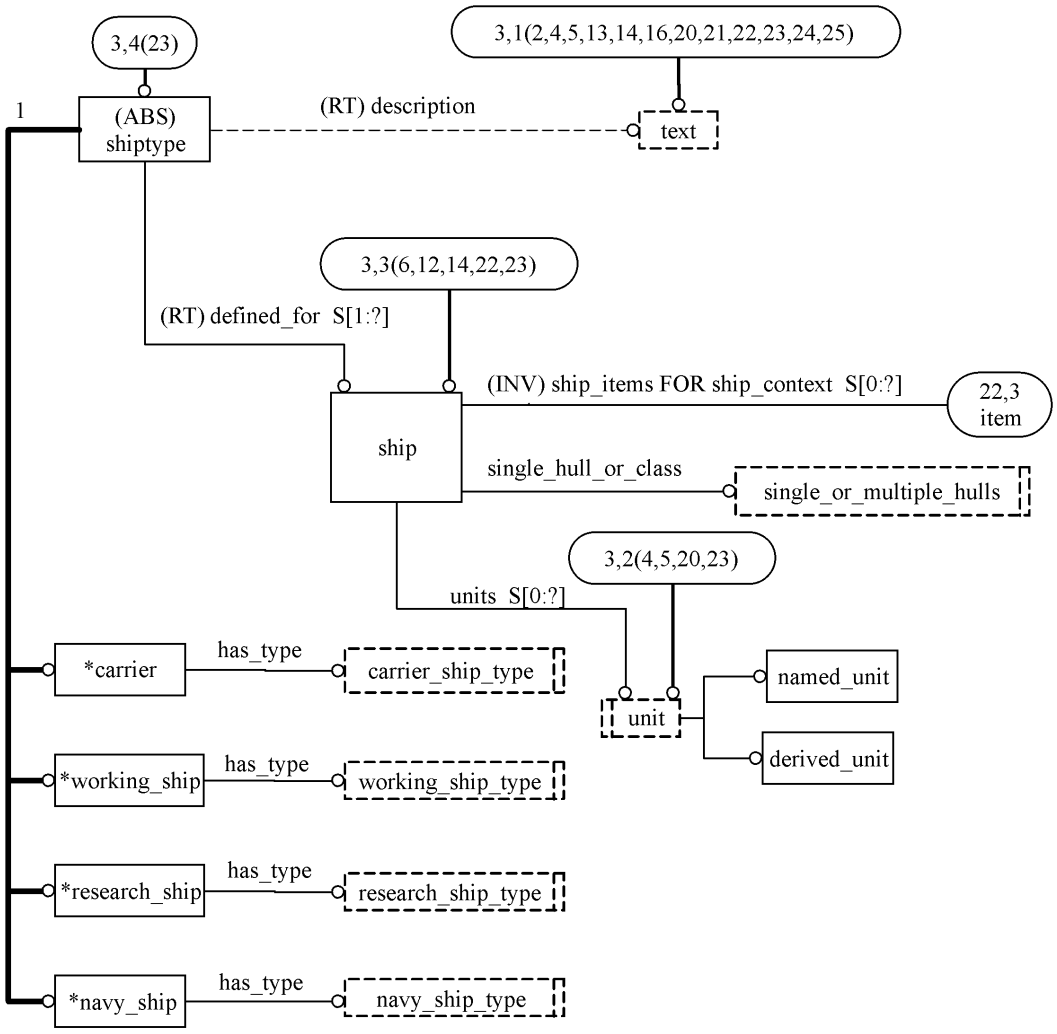


Figure G.3 — ARM EXPRESS-G diagram 3 of 25

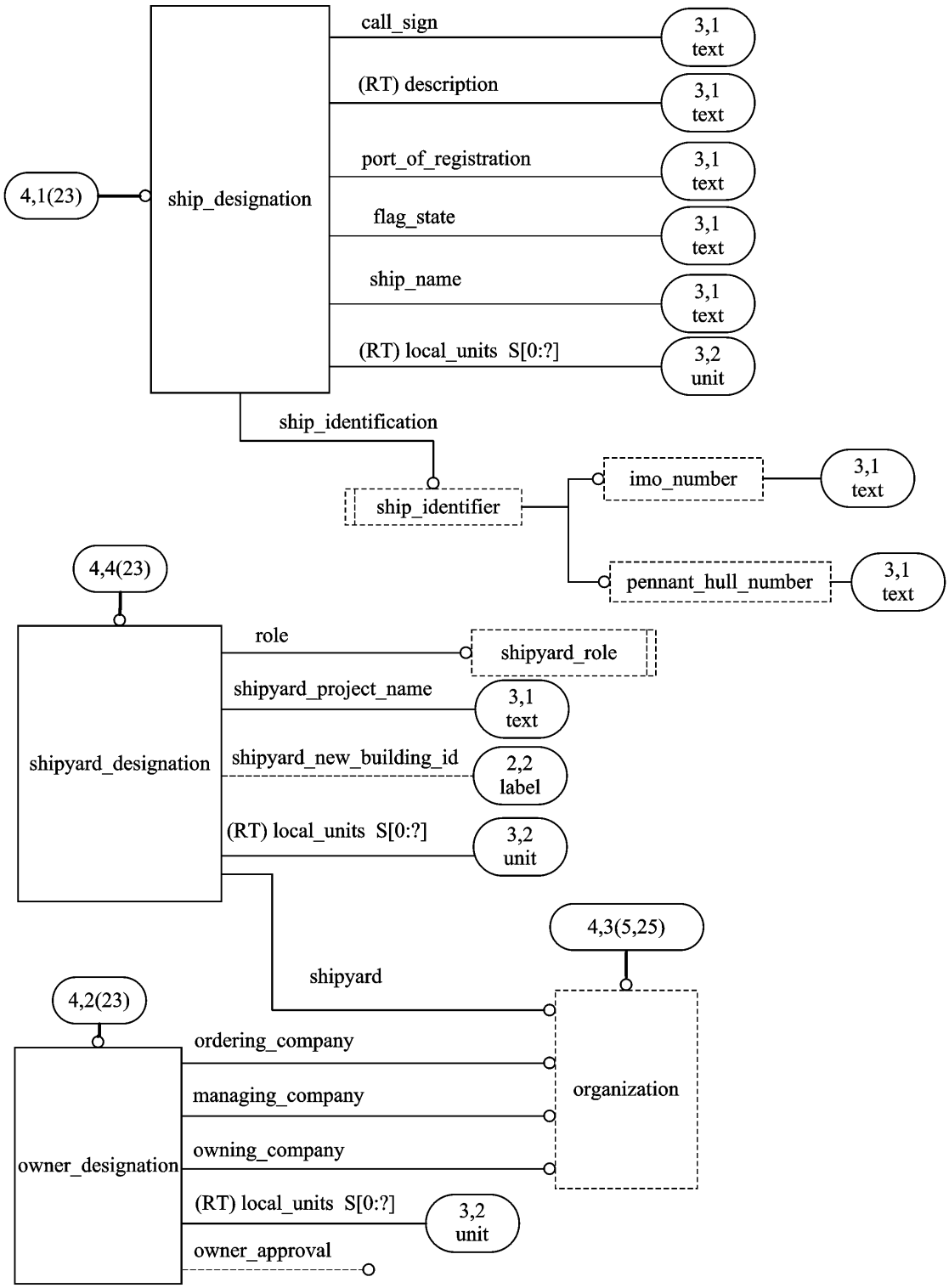


Figure G.4 — ARM EXPRESS-G diagram 4 of 25

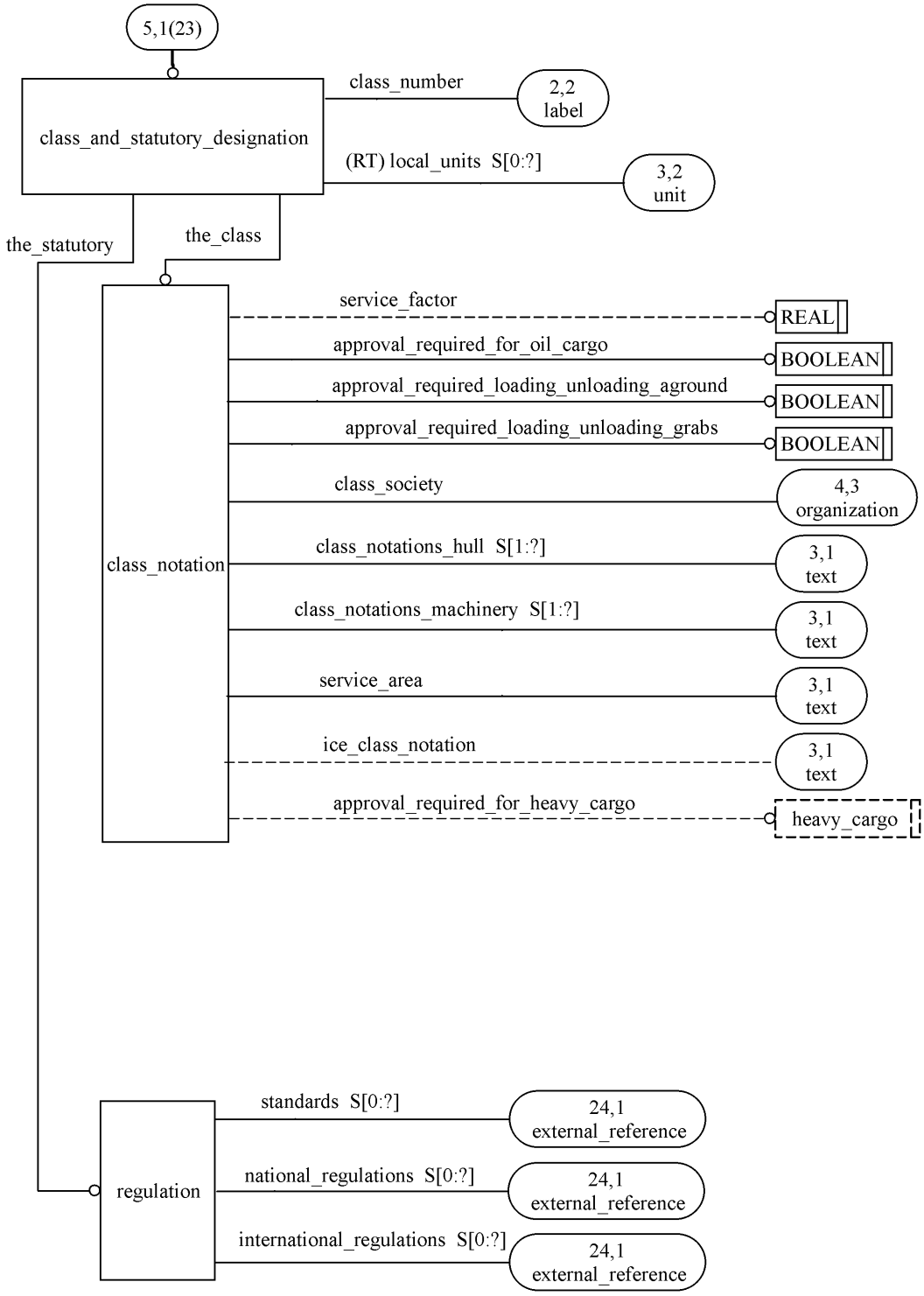


Figure G.5 — ARM EXPRESS-G diagram 5 of 25

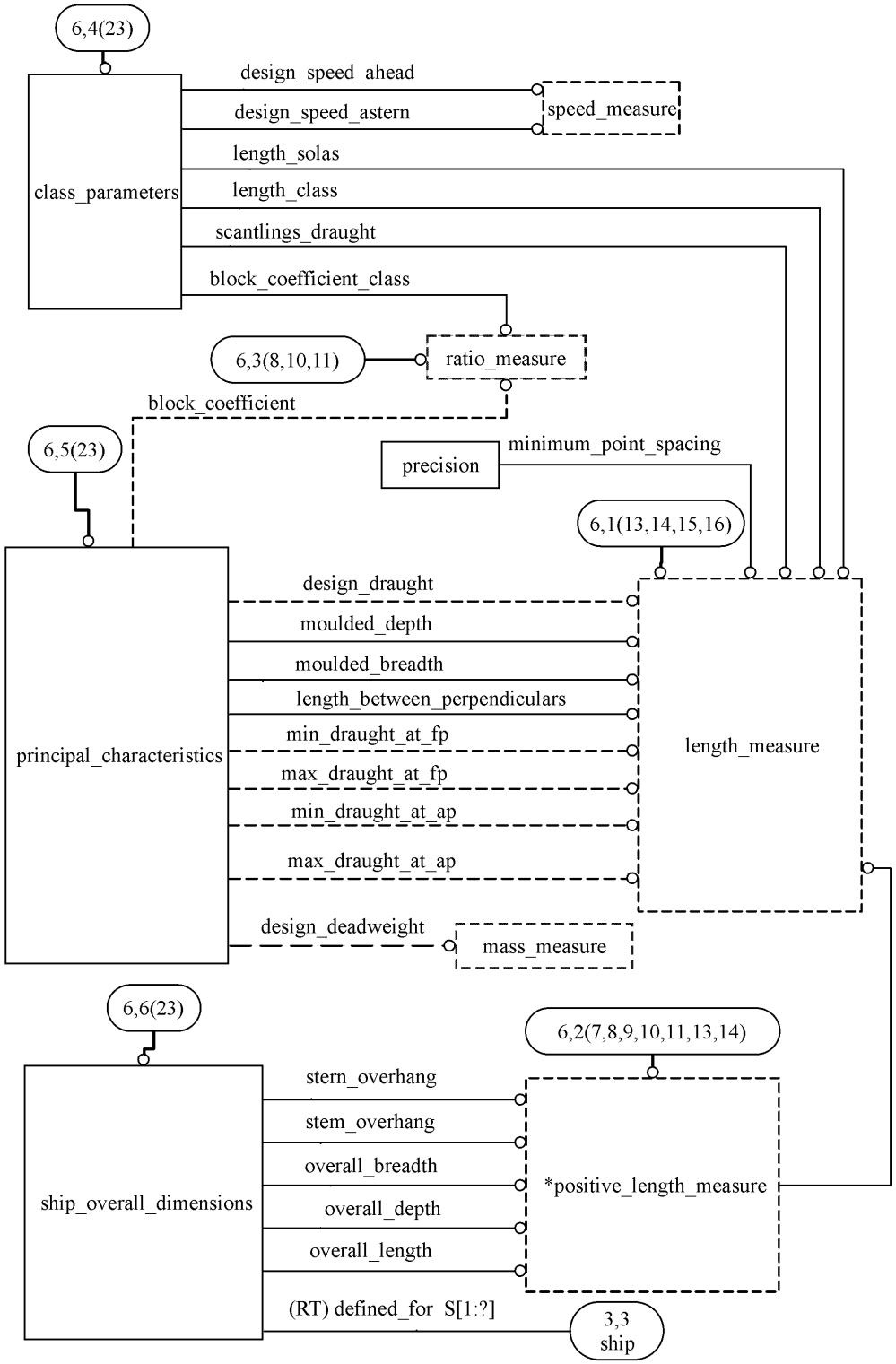


Figure G.6 — ARM EXPRESS-G diagram 6 of 25

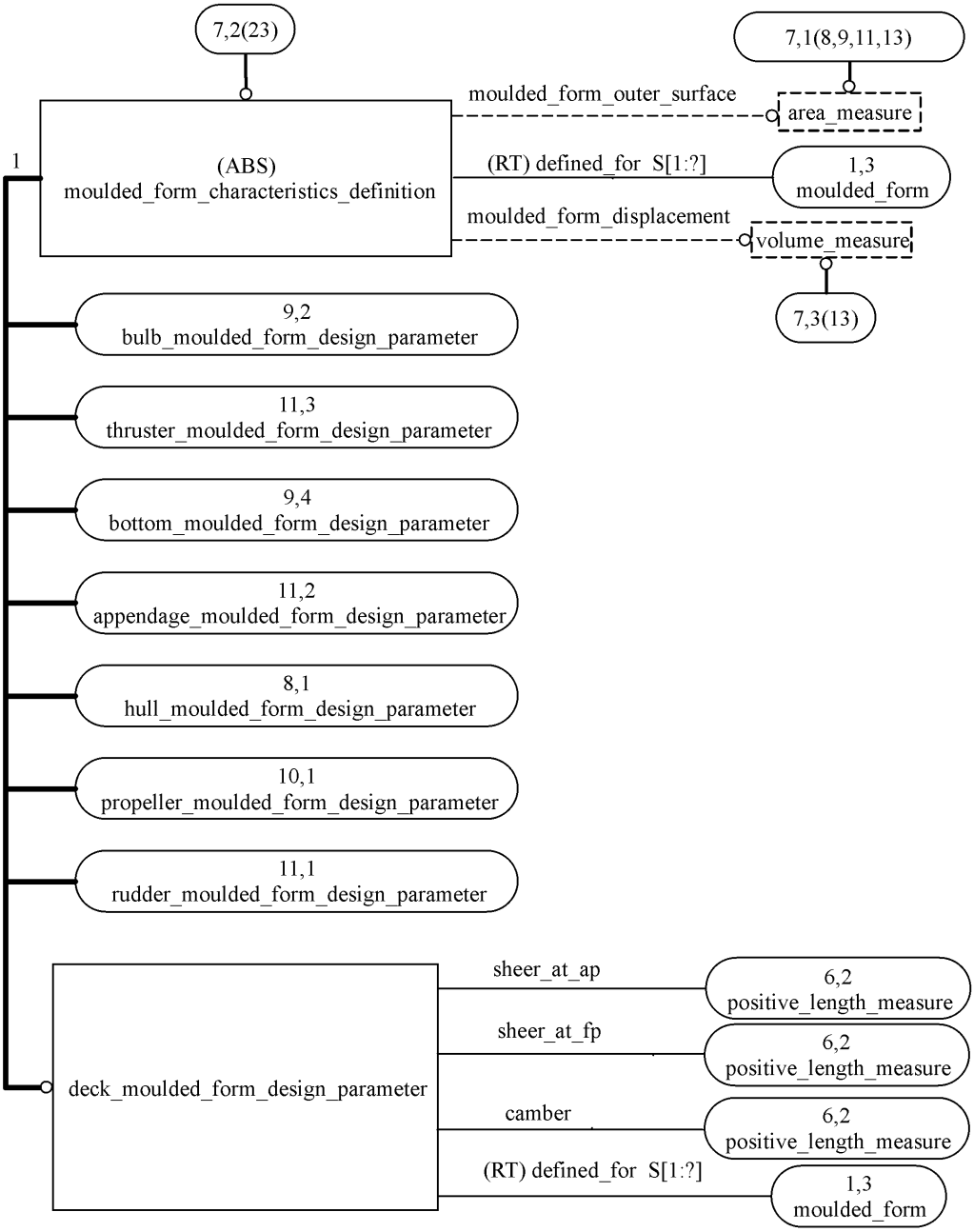


Figure G.7 — ARM EXPRESS-G diagram 7 of 25

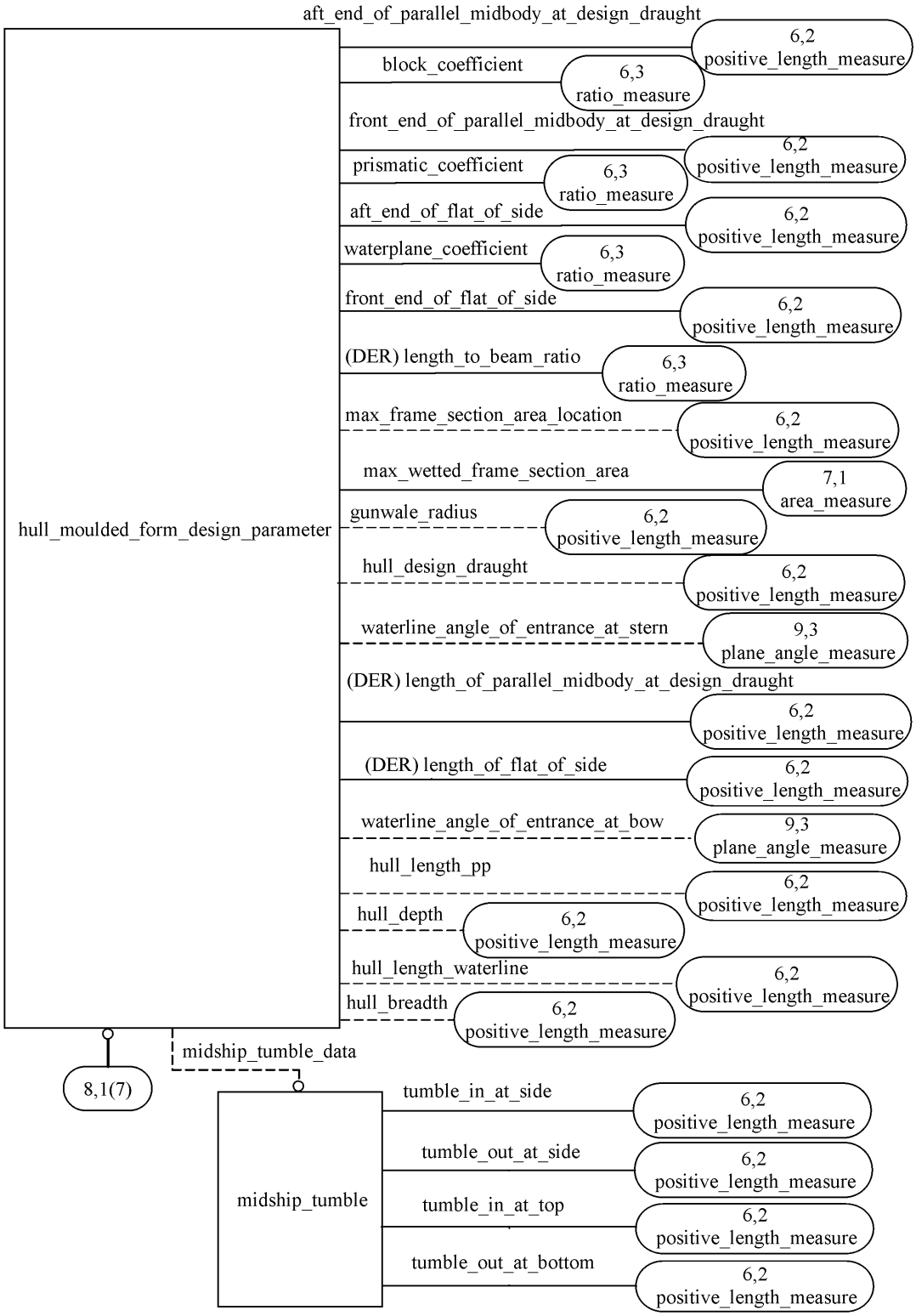


Figure G.8 — ARM EXPRESS-G diagram 8 of 25

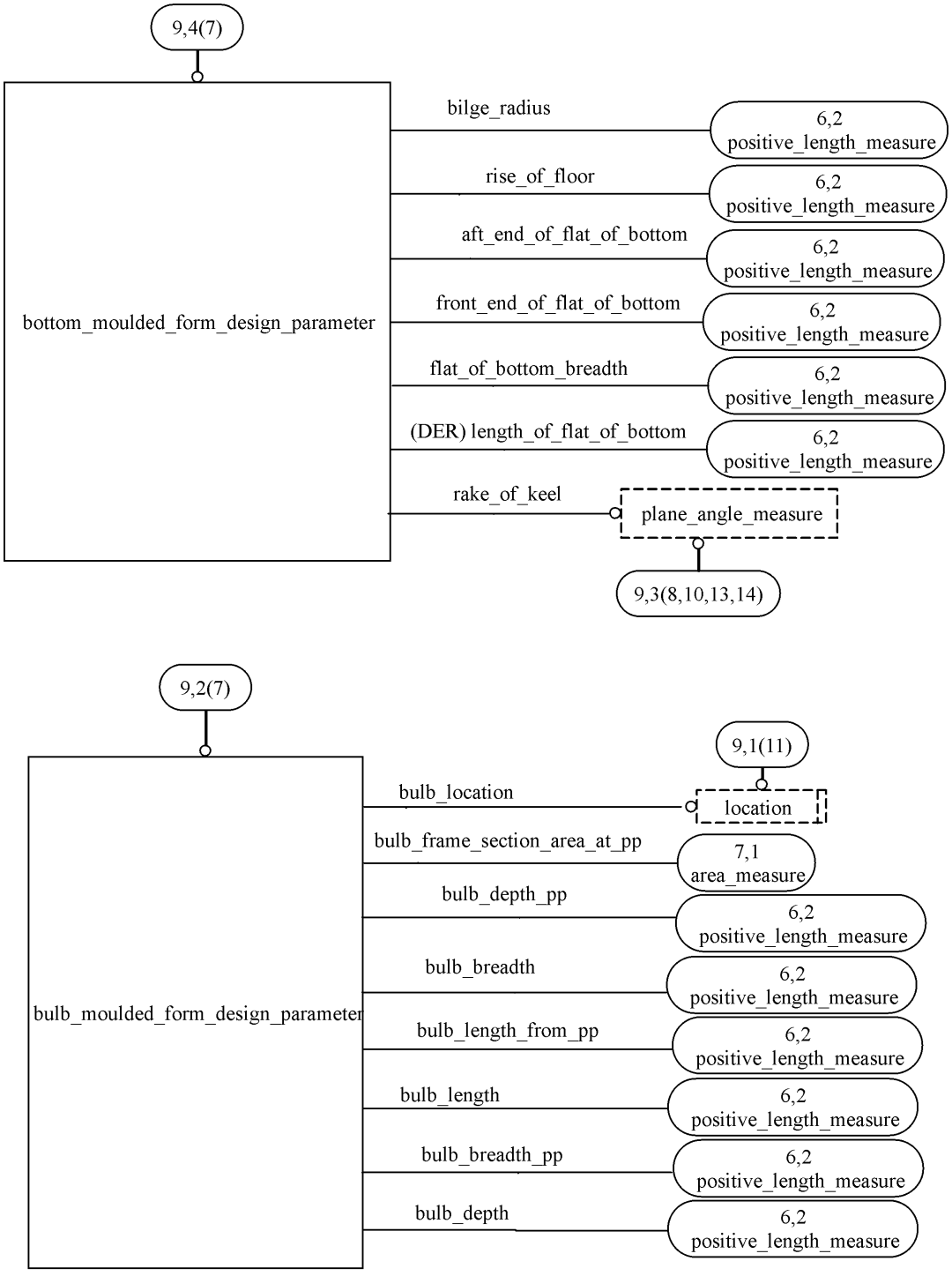


Figure G.9 — ARM EXPRESS-G diagram 9 of 25

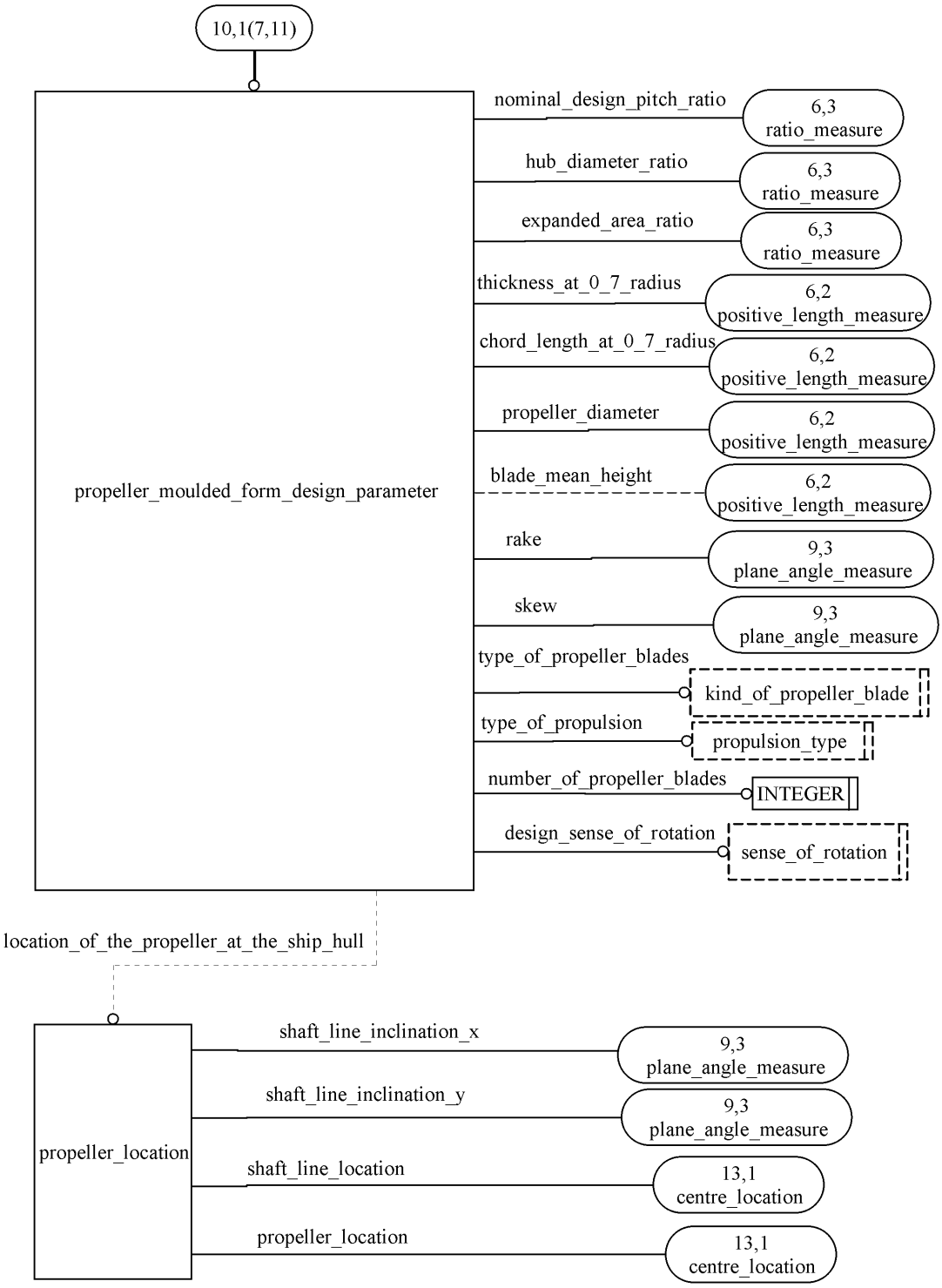


Figure G.10 — ARM EXPRESS-G diagram 10 of 25

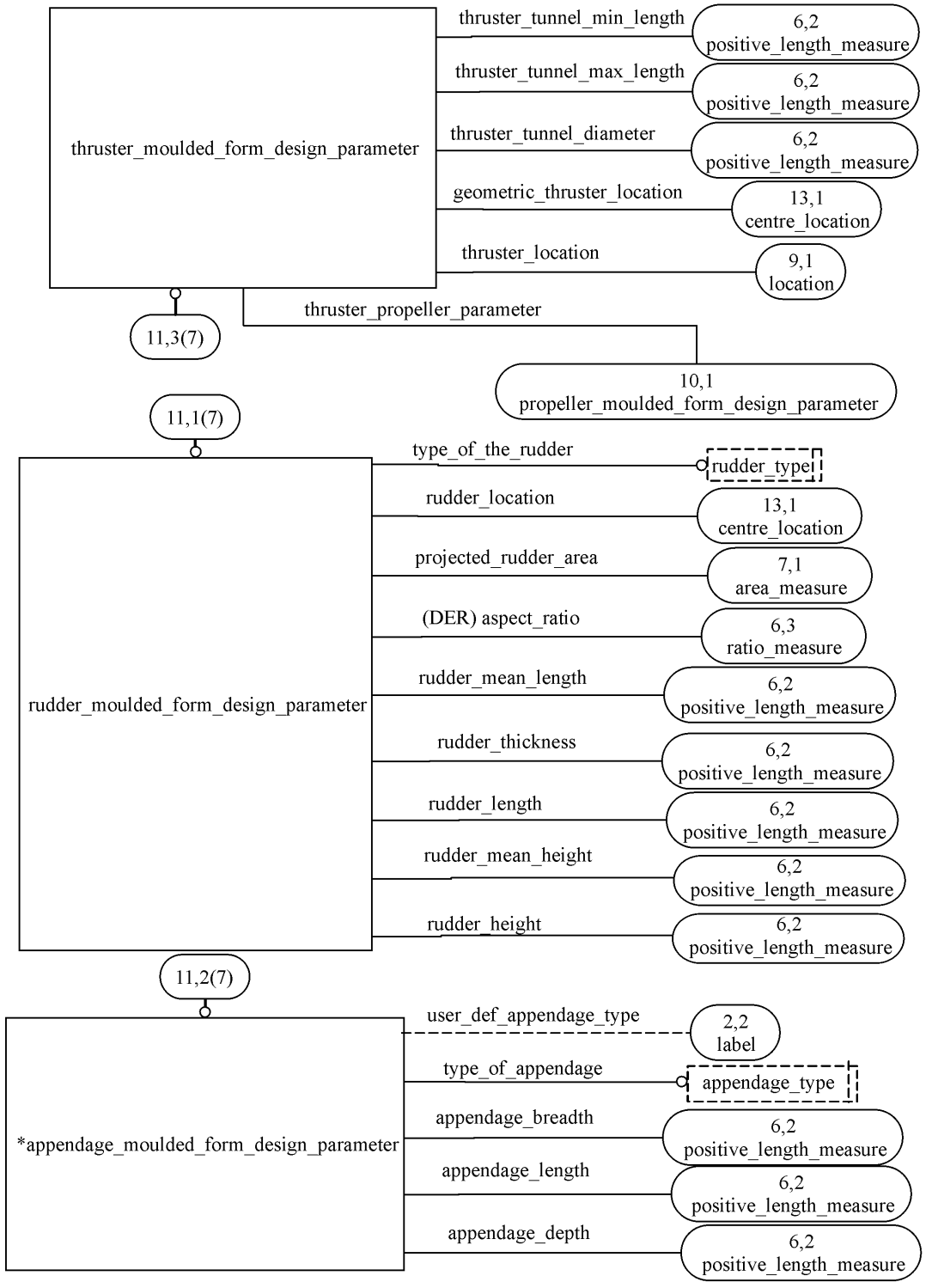


Figure G.11 — ARM EXPRESS-G diagram 11 of 25

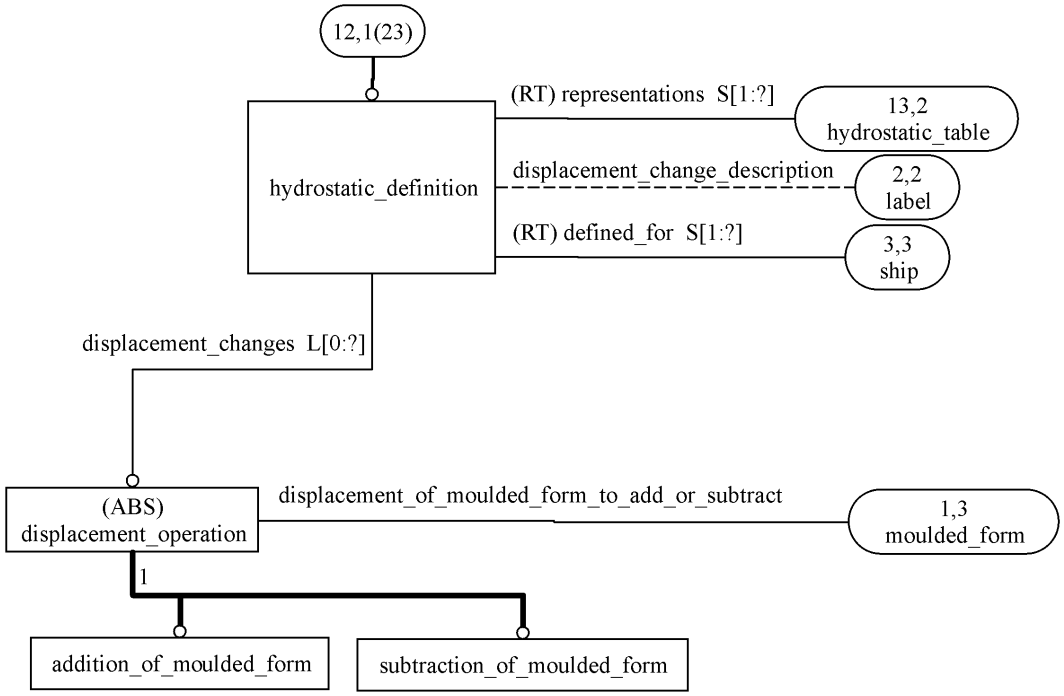


Figure G.12 — ARM EXPRESS-G diagram 12 of 25

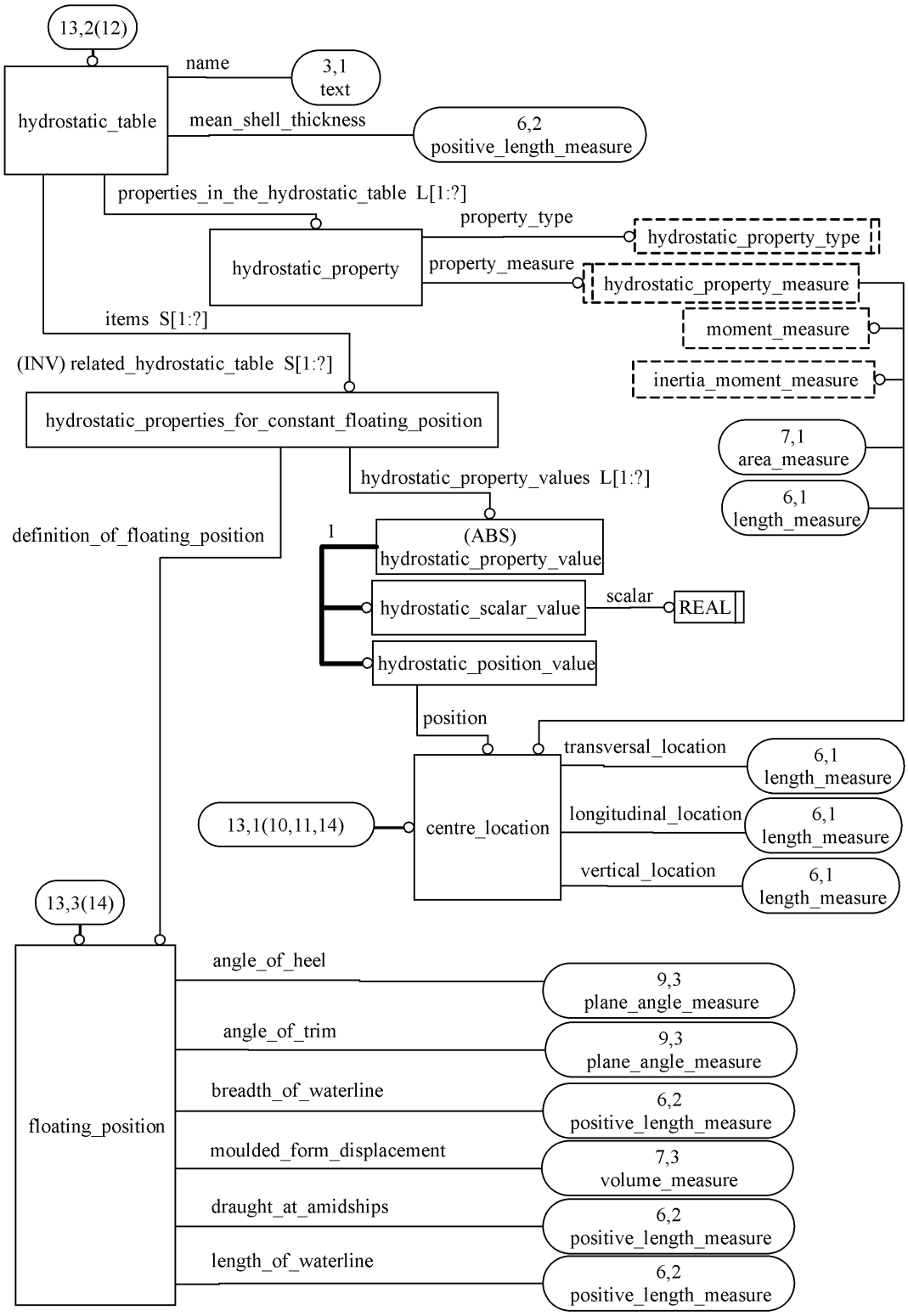


Figure G.13 — ARM EXPRESS-G diagram 13 of 25

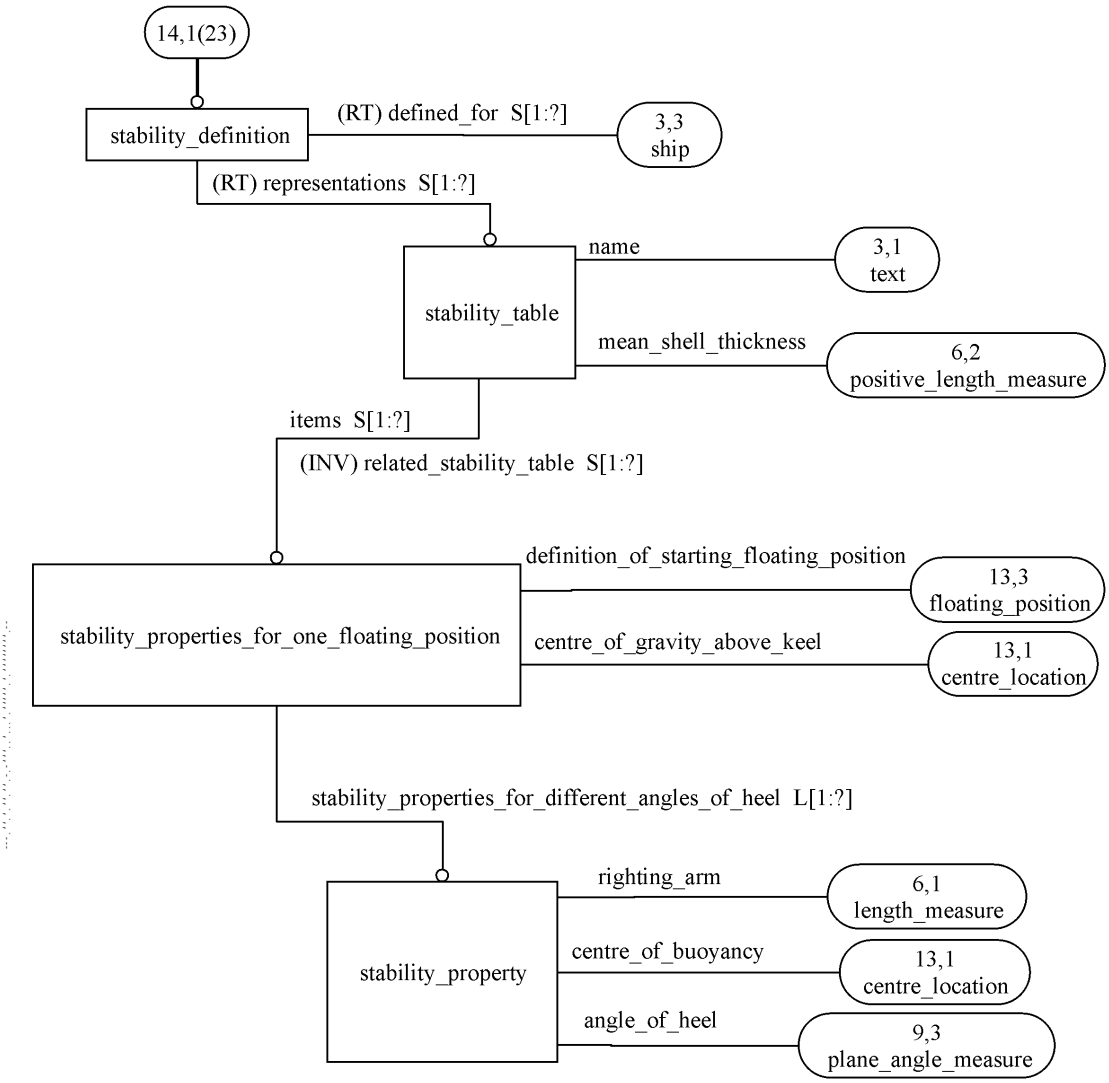


Figure G.14 — ARM EXPRESS-G diagram 14 of 25

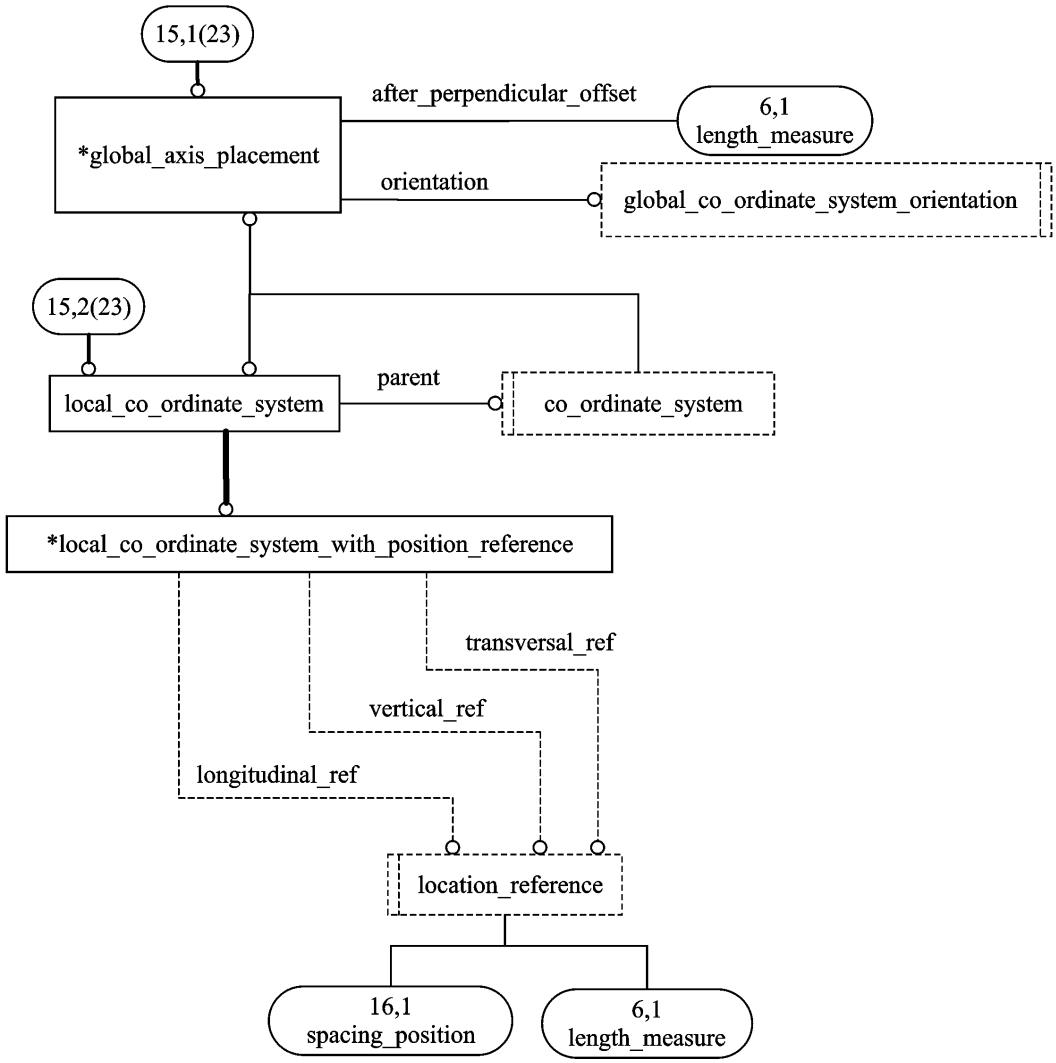


Figure G.15 — ARM EXPRESS-G diagram 15 of 25

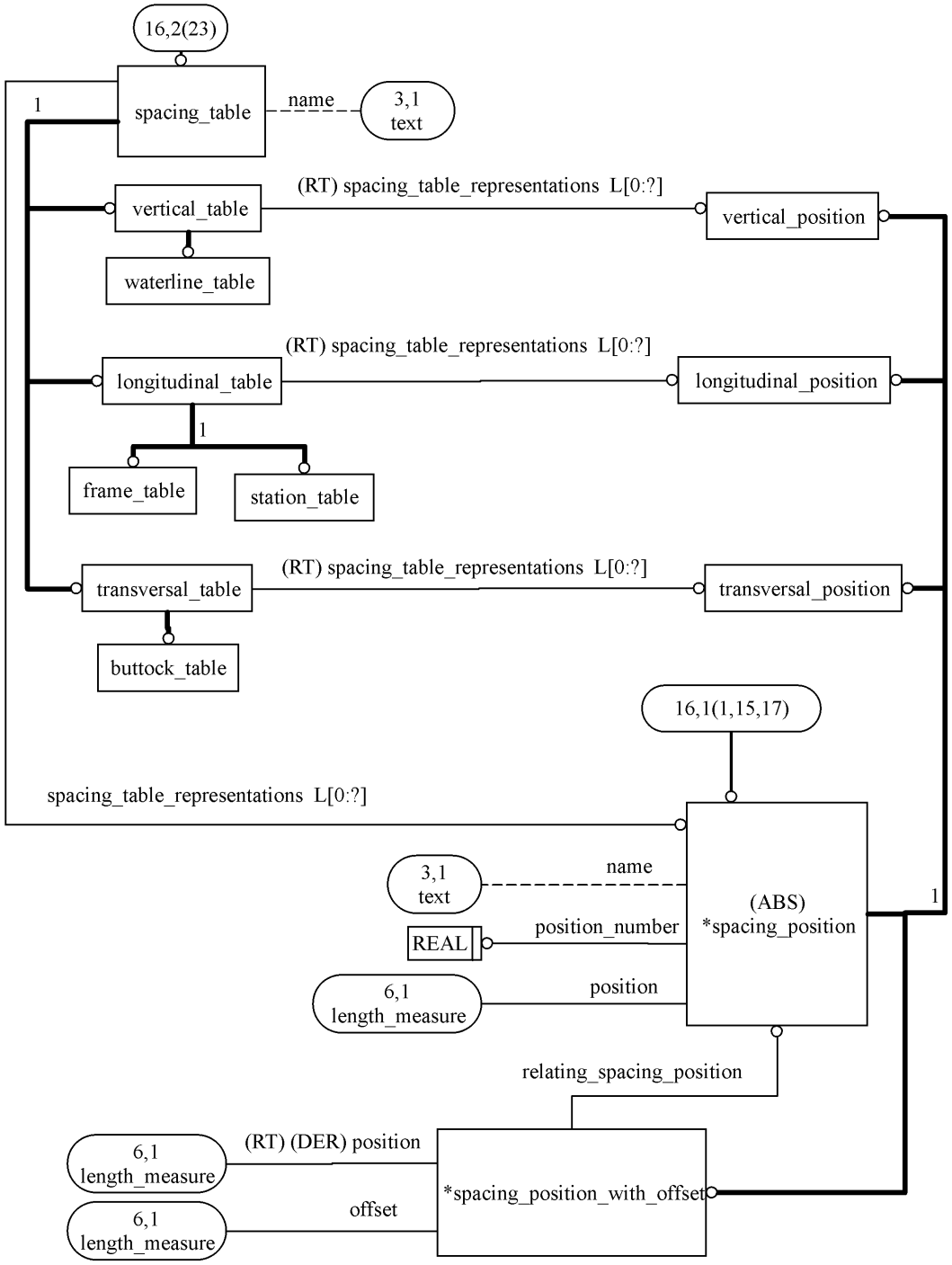


Figure G.16 — ARM EXPRESS-G diagram 16 of 25

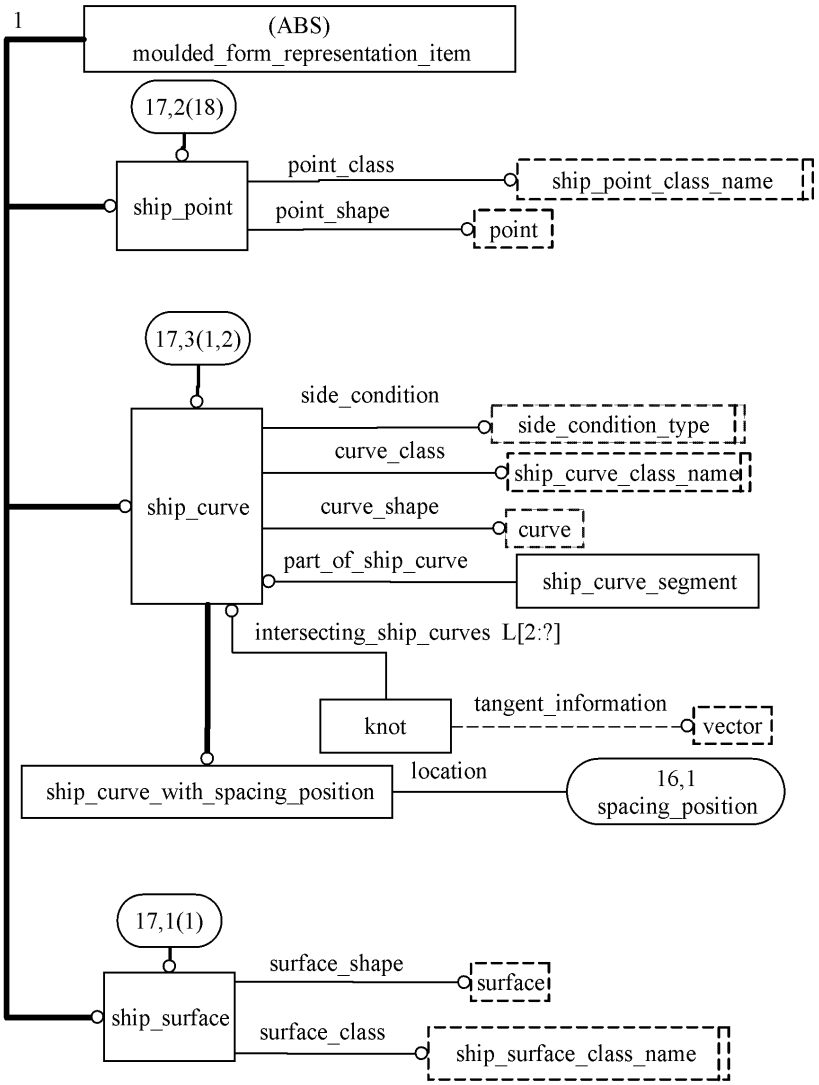


Figure G.17 — ARM EXPRESS-G diagram 17 of 25

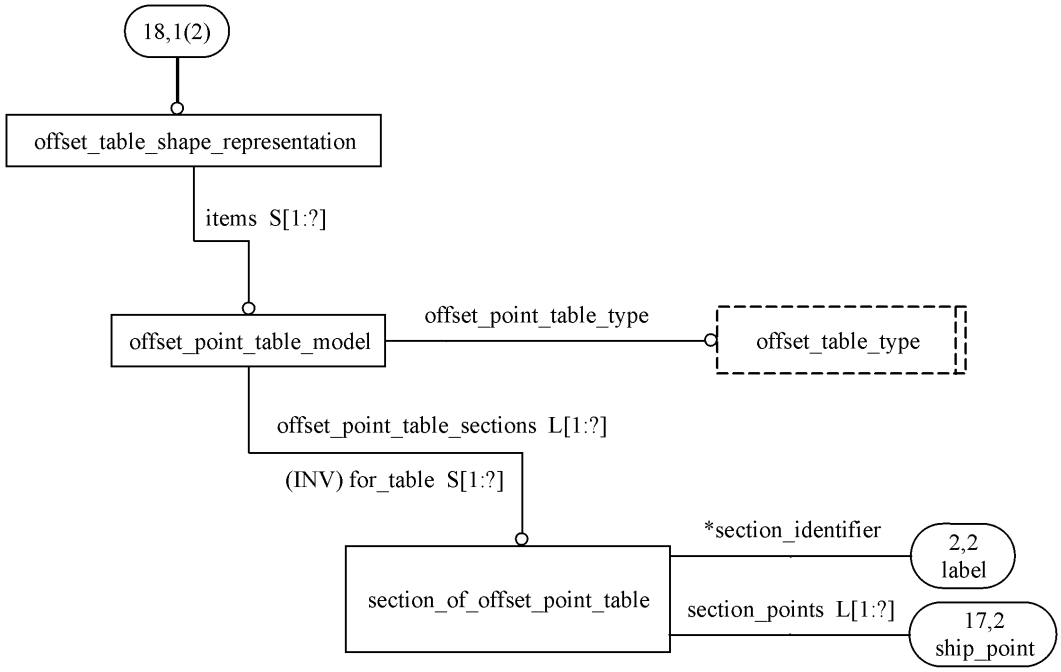


Figure G.18 — ARM EXPRESS-G diagram 18 of 25

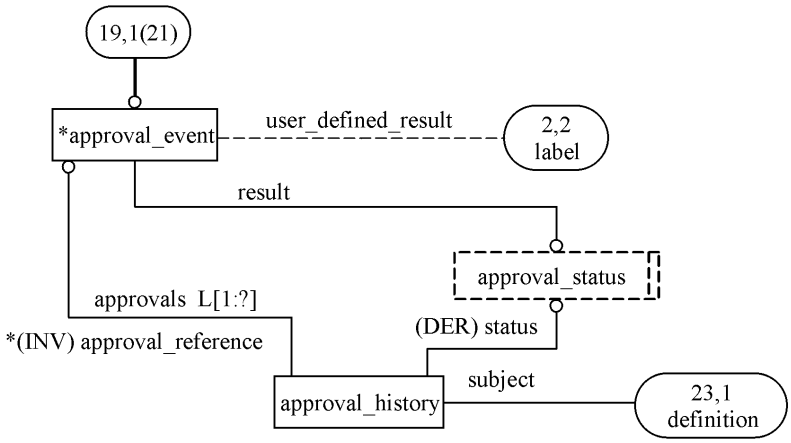


Figure G.19 — ARM EXPRESS-G diagram 19 of 25

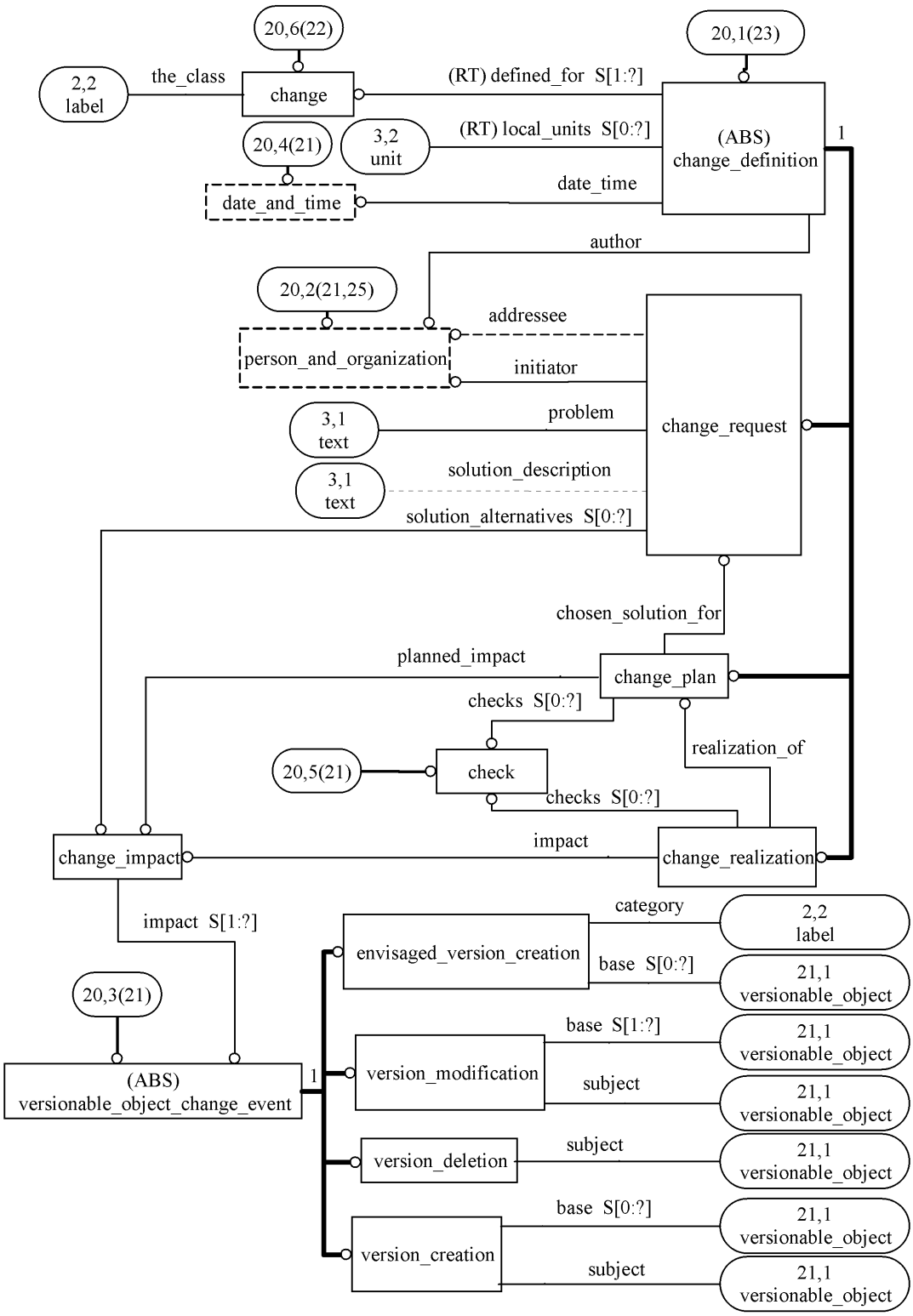


Figure G.20 — ARM EXPRESS-G diagram 20 of 25

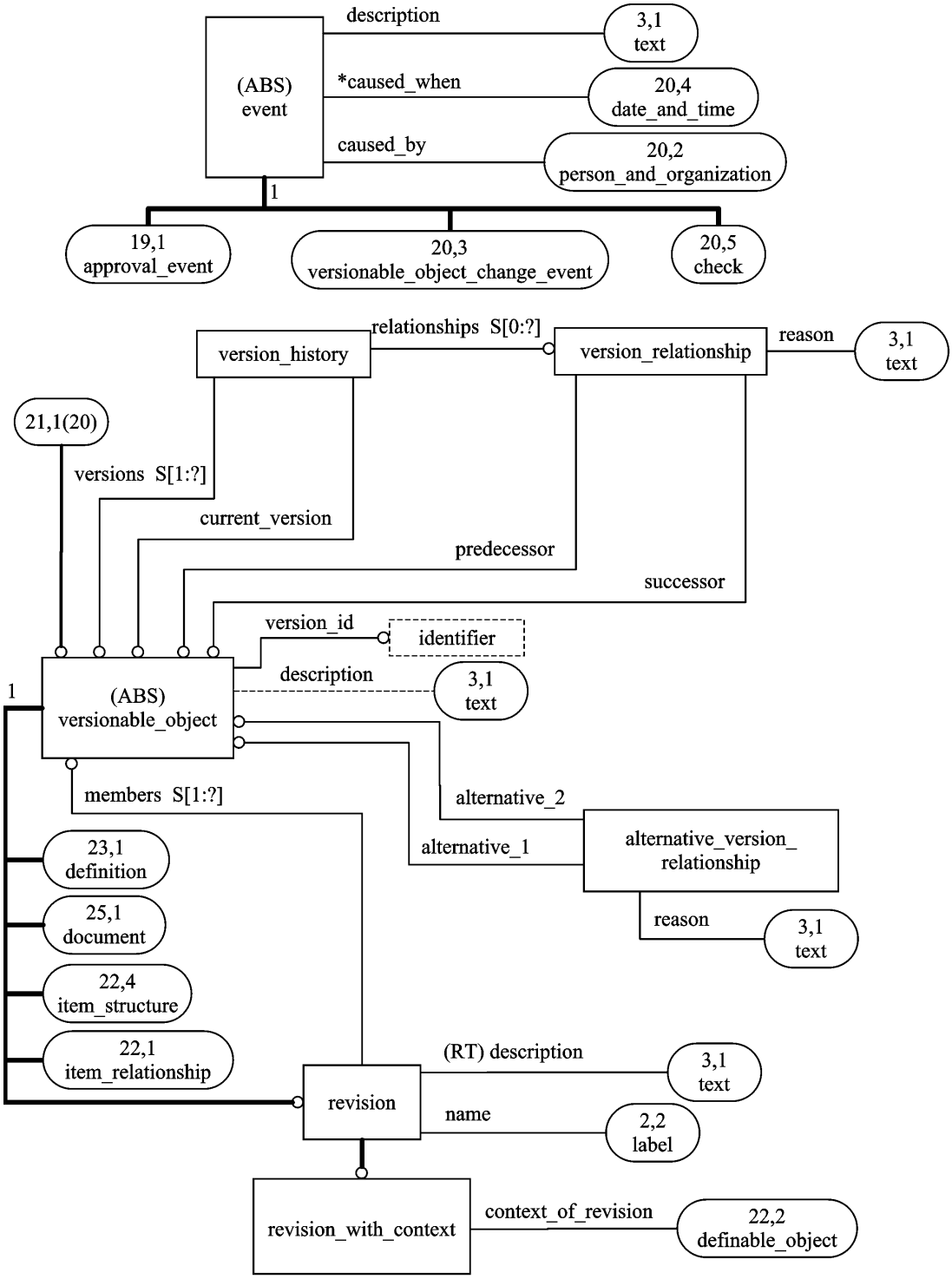


Figure G.21 — ARM EXPRESS-G diagram 21 of 25

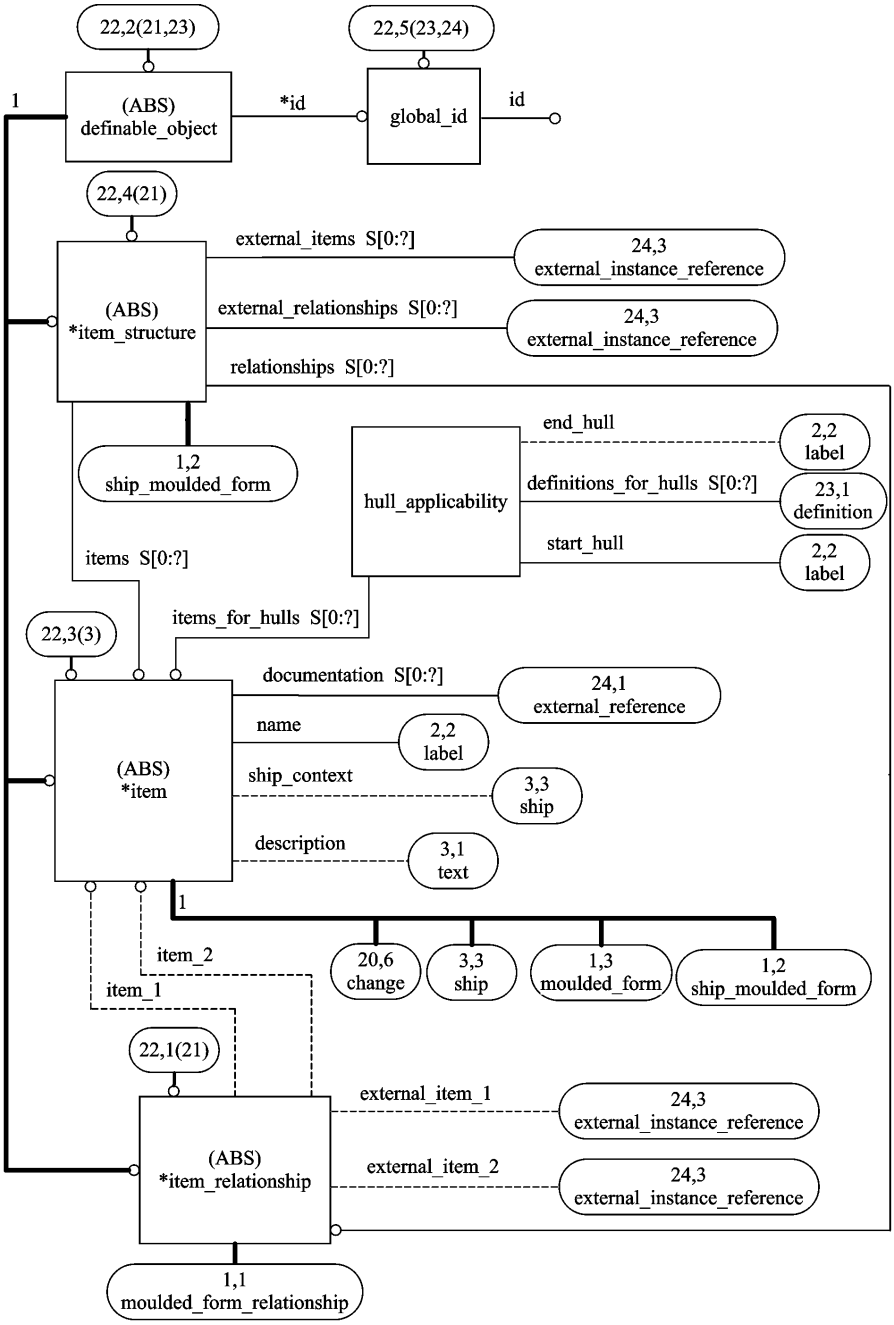


Figure G.22 — ARM EXPRESS-G diagram 22 of 25

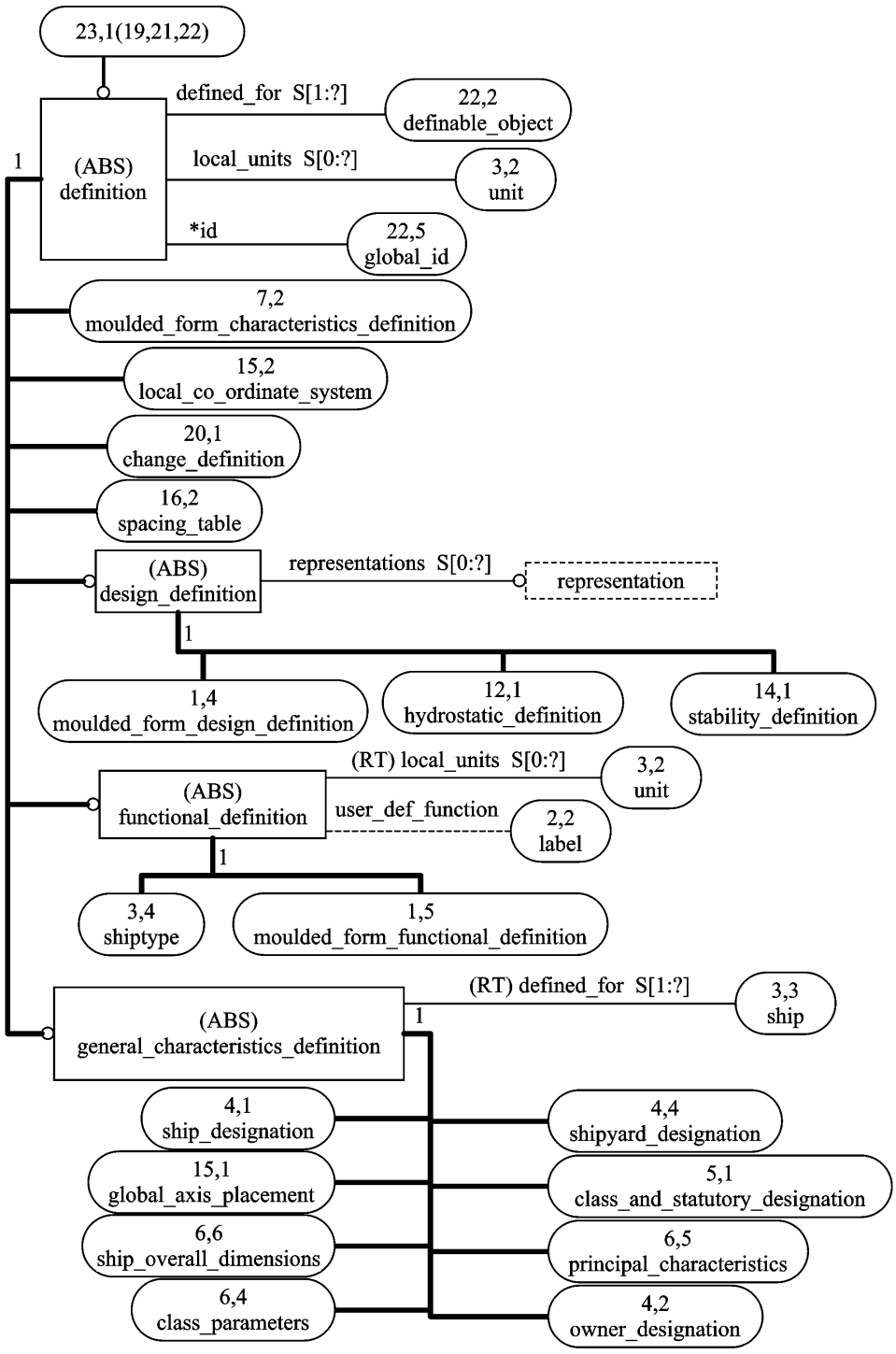
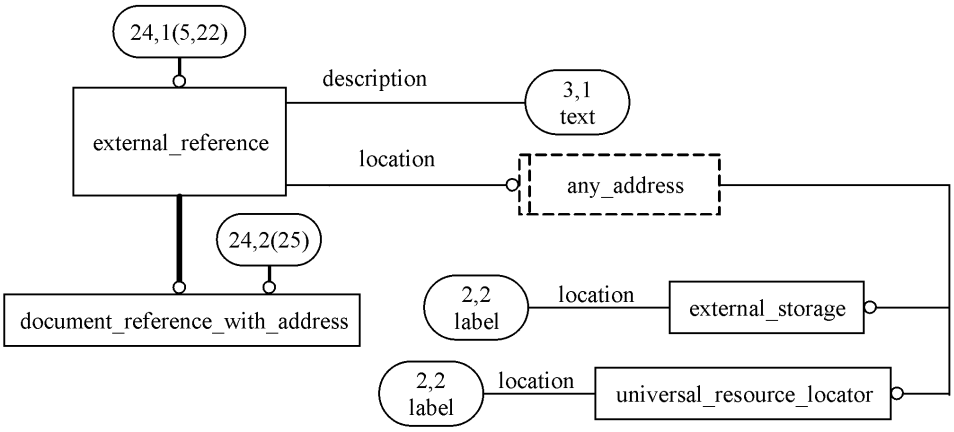


Figure G.23 — ARM EXPRESS-G diagram 23 of 25



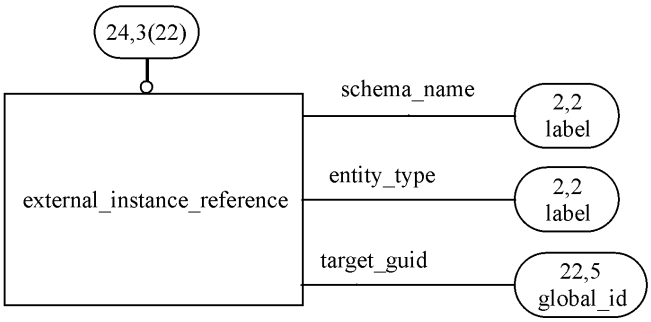


Figure G.24 — ARM EXPRESS-G diagram 24 of 25

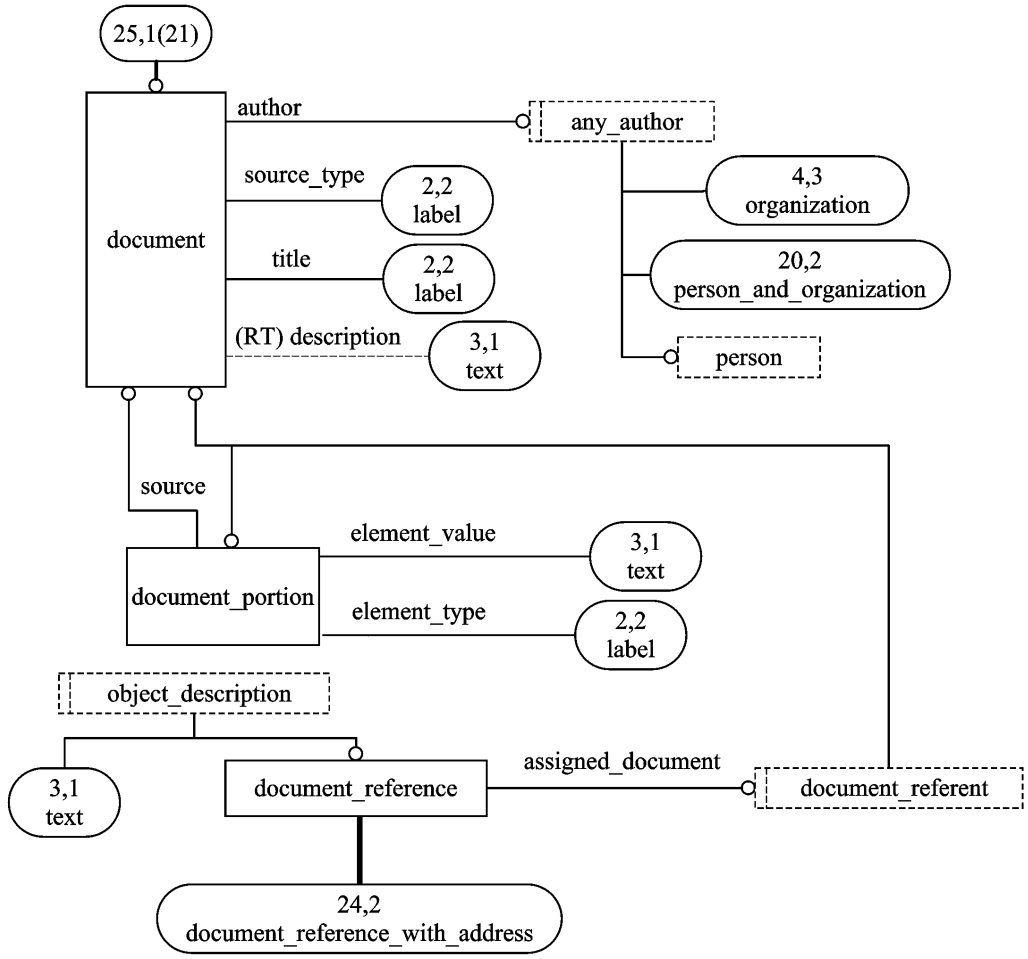


Figure G.25 — ARM EXPRESS-G diagram 25 of 25

Annex H
(informative)

AIM EXPRESS-G

Figure H.1 through H.26 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 an annex A of ISO 10303-11.

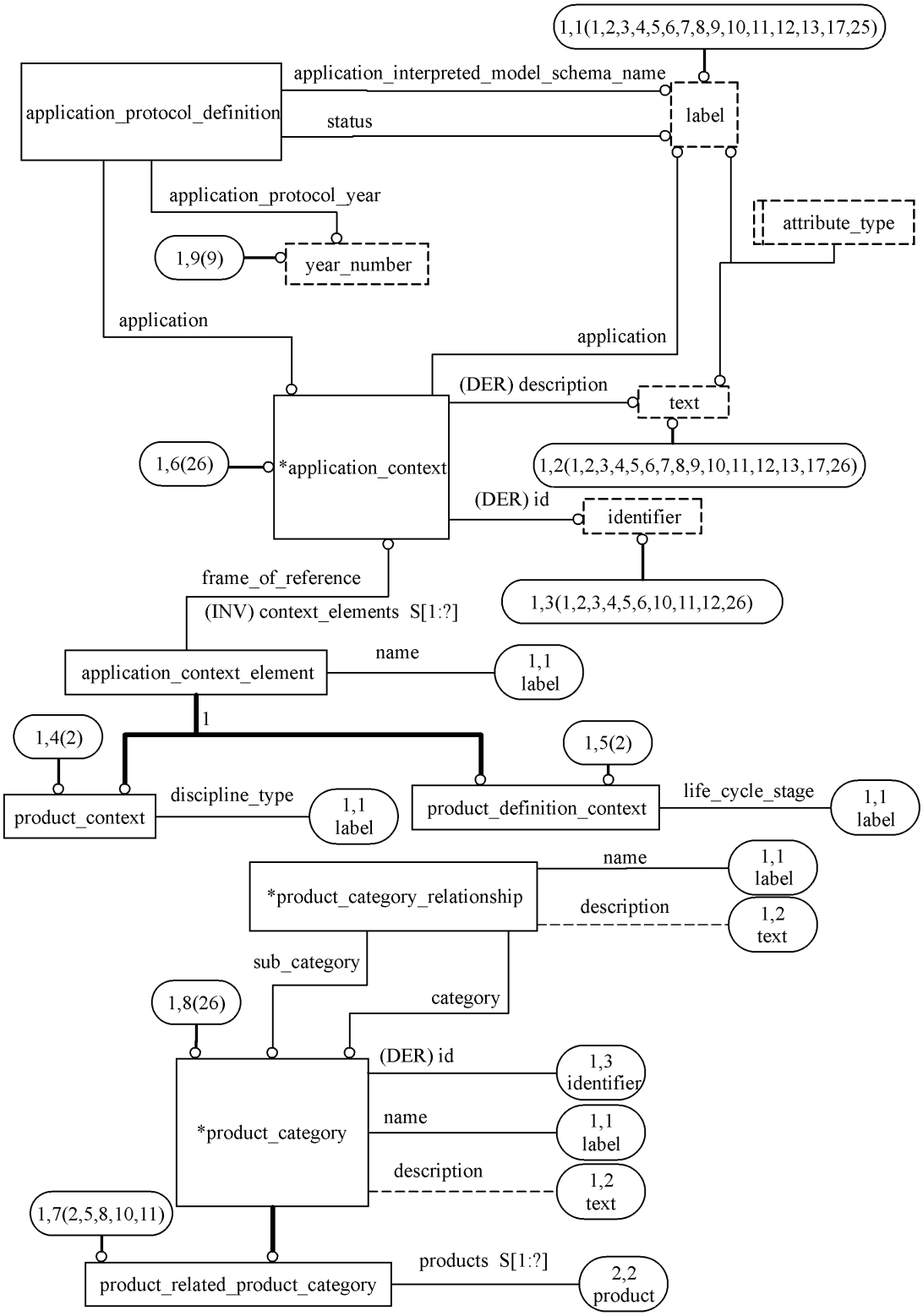


Figure H.1 — application context - AIM diagram 1 of 26 in EXPRESS-G

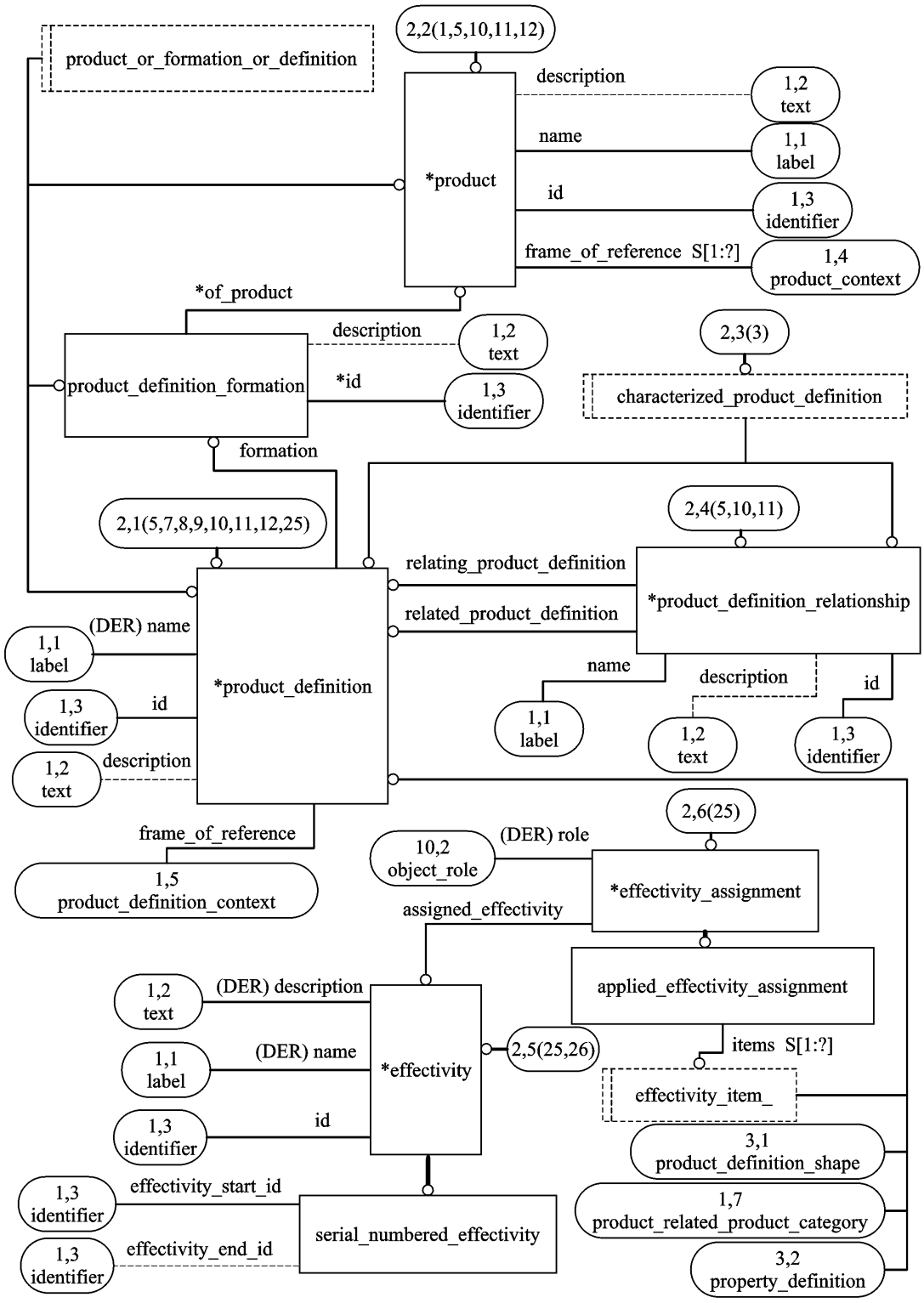


Figure H.2 — product definition - AIM diagram 2 of 26 in EXPRESS-G

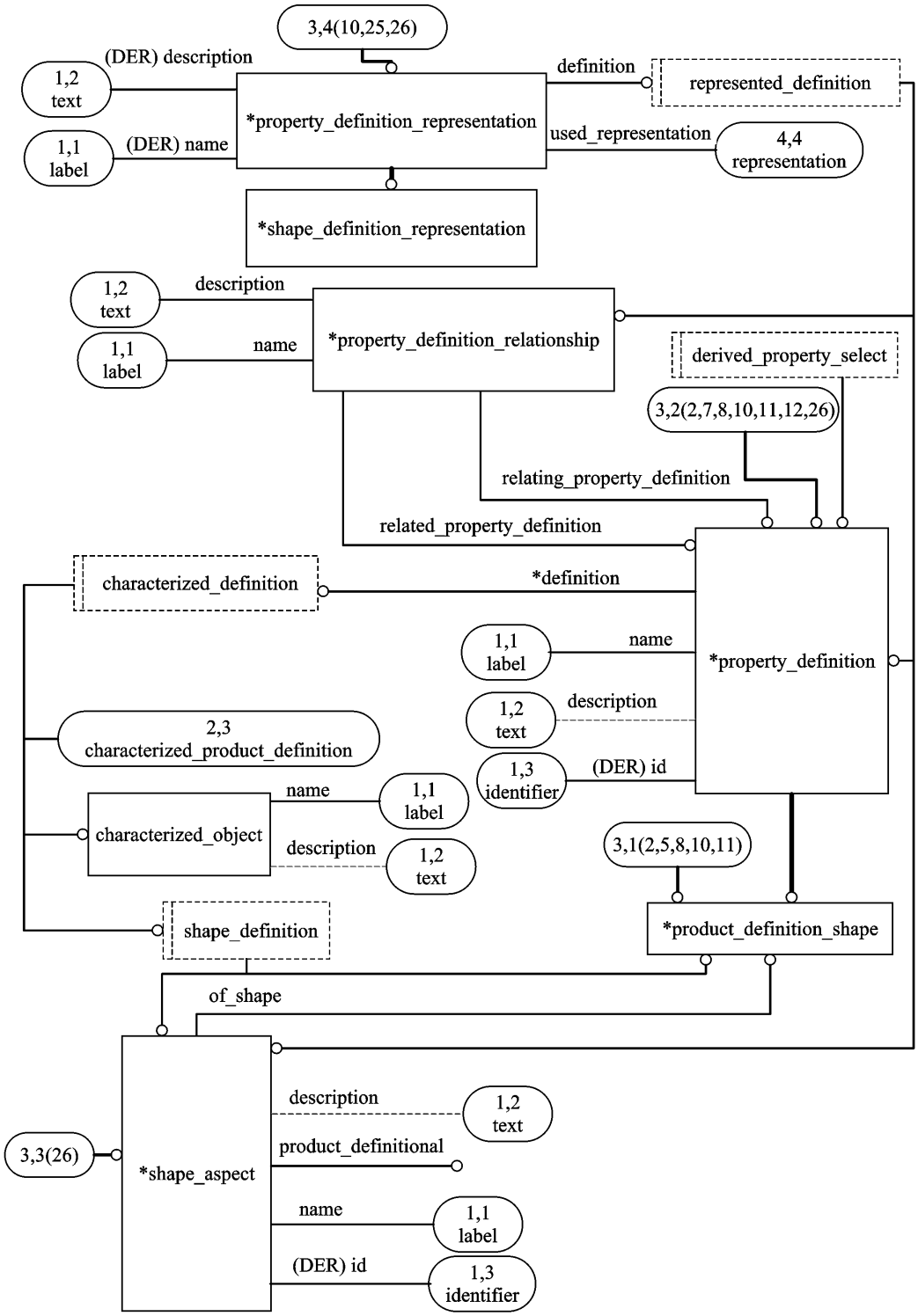


Figure H.3 — property definition - AIM diagram 3 of 26 in EXPRESS-G

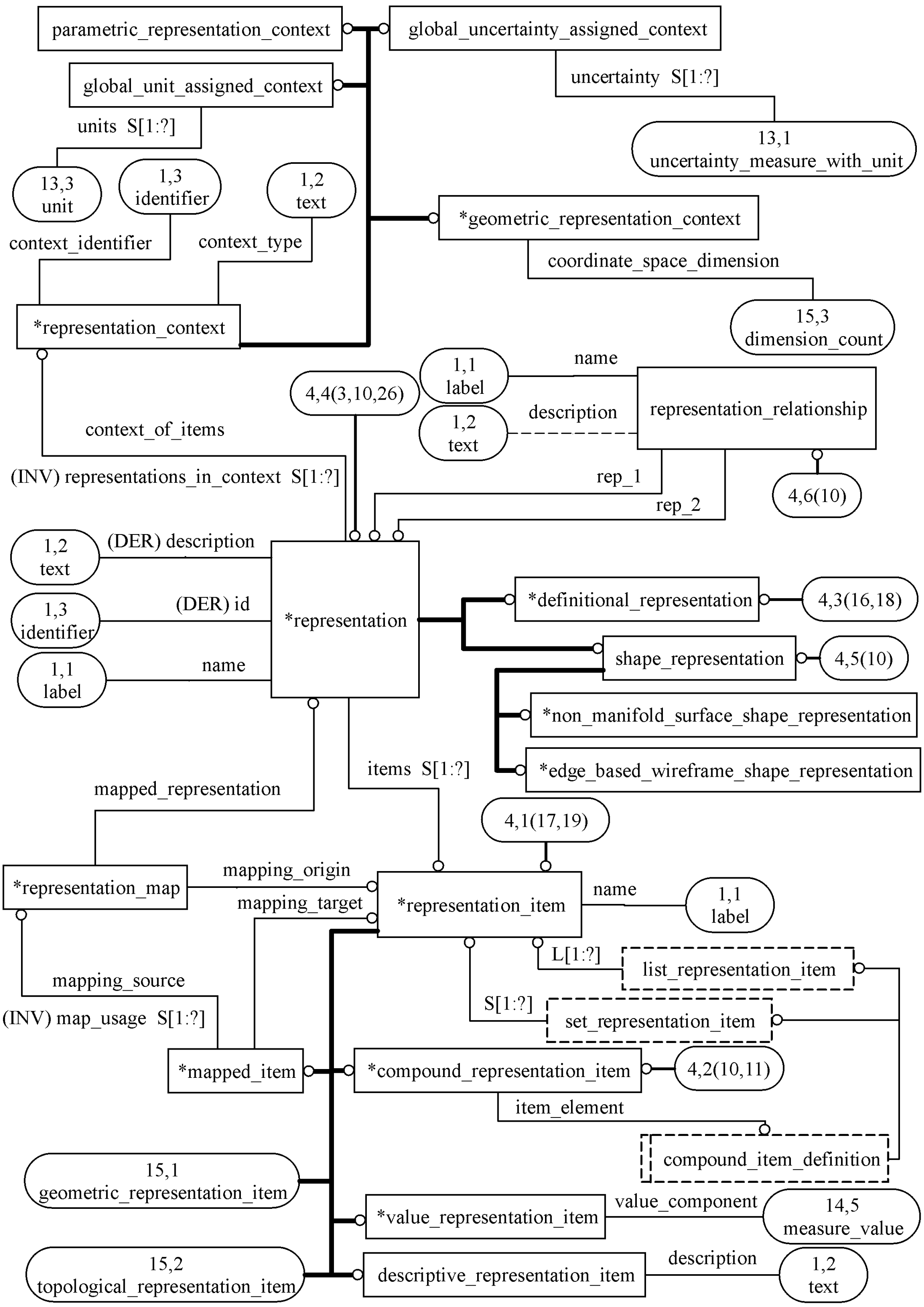


Figure H.4 — representation - AIM diagram 4 of 26 in EXPRESS-G

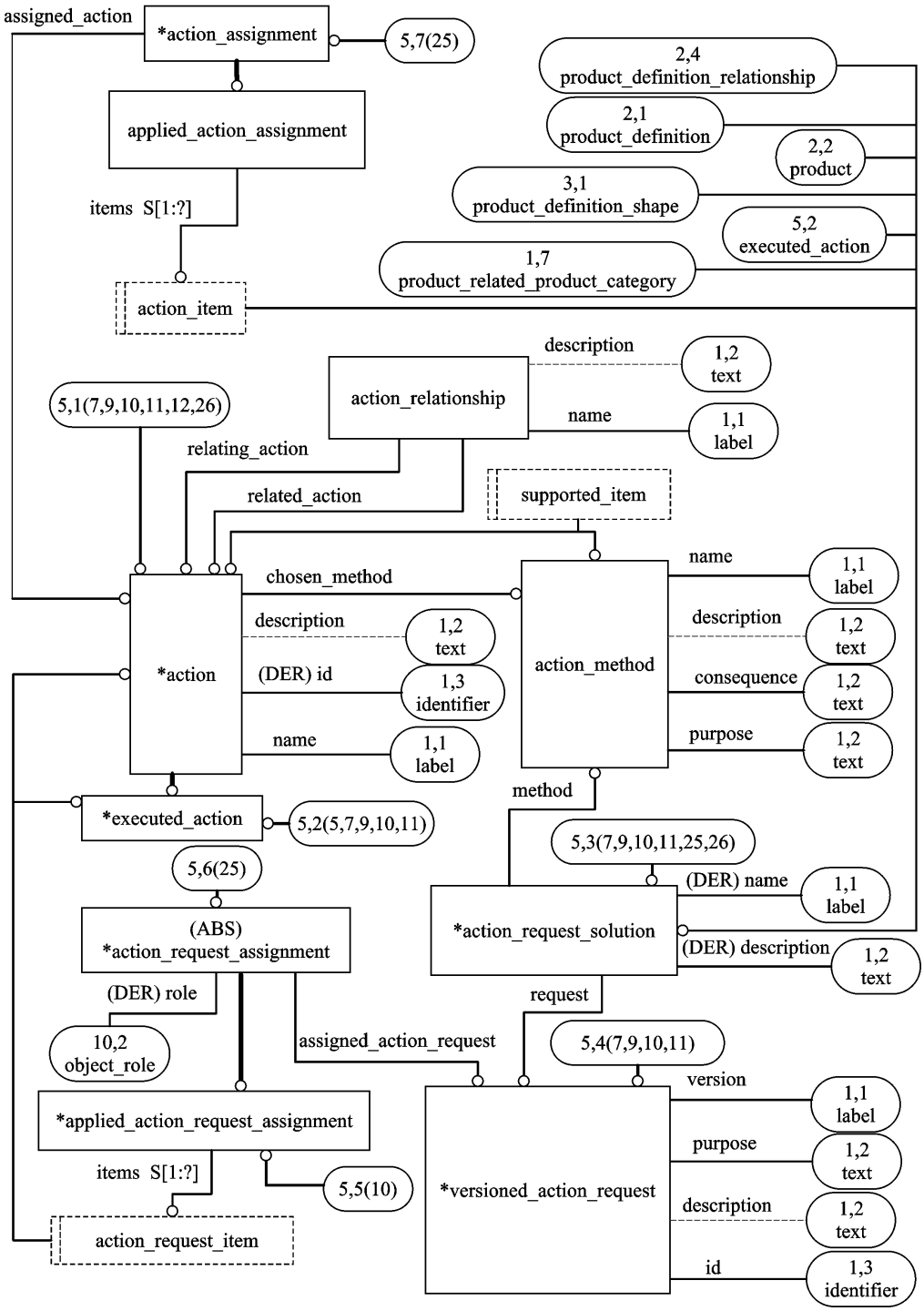


Figure H.5 — action - AIM diagram 5 of 26 in EXPRESS-G

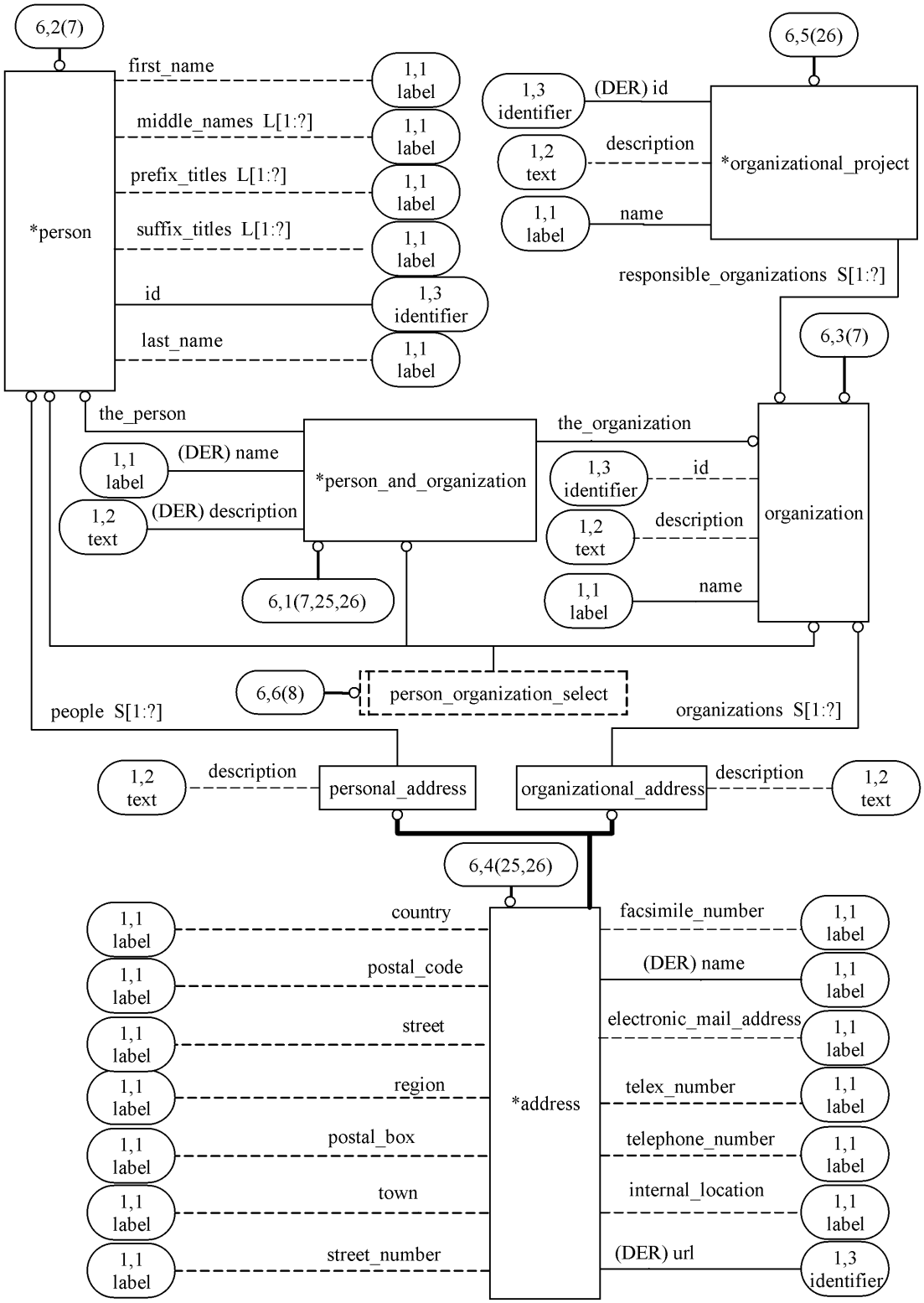


Figure H.6 — person and organization - AIM diagram 6 of 26 in EXPRESS-G

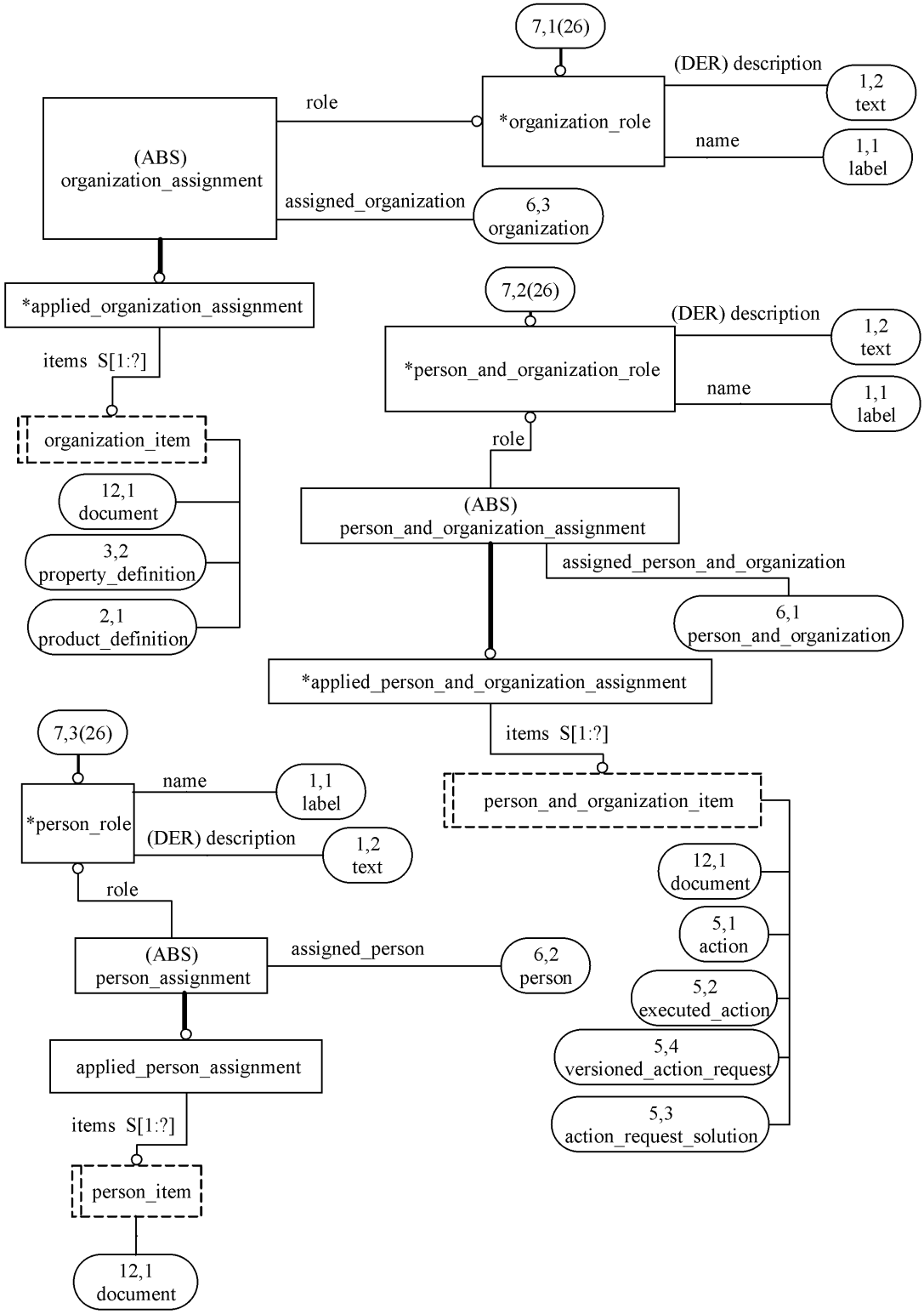


Figure H.7 — person and organization assignment - AIM diagram 7 of 26 in EXPRESS-G

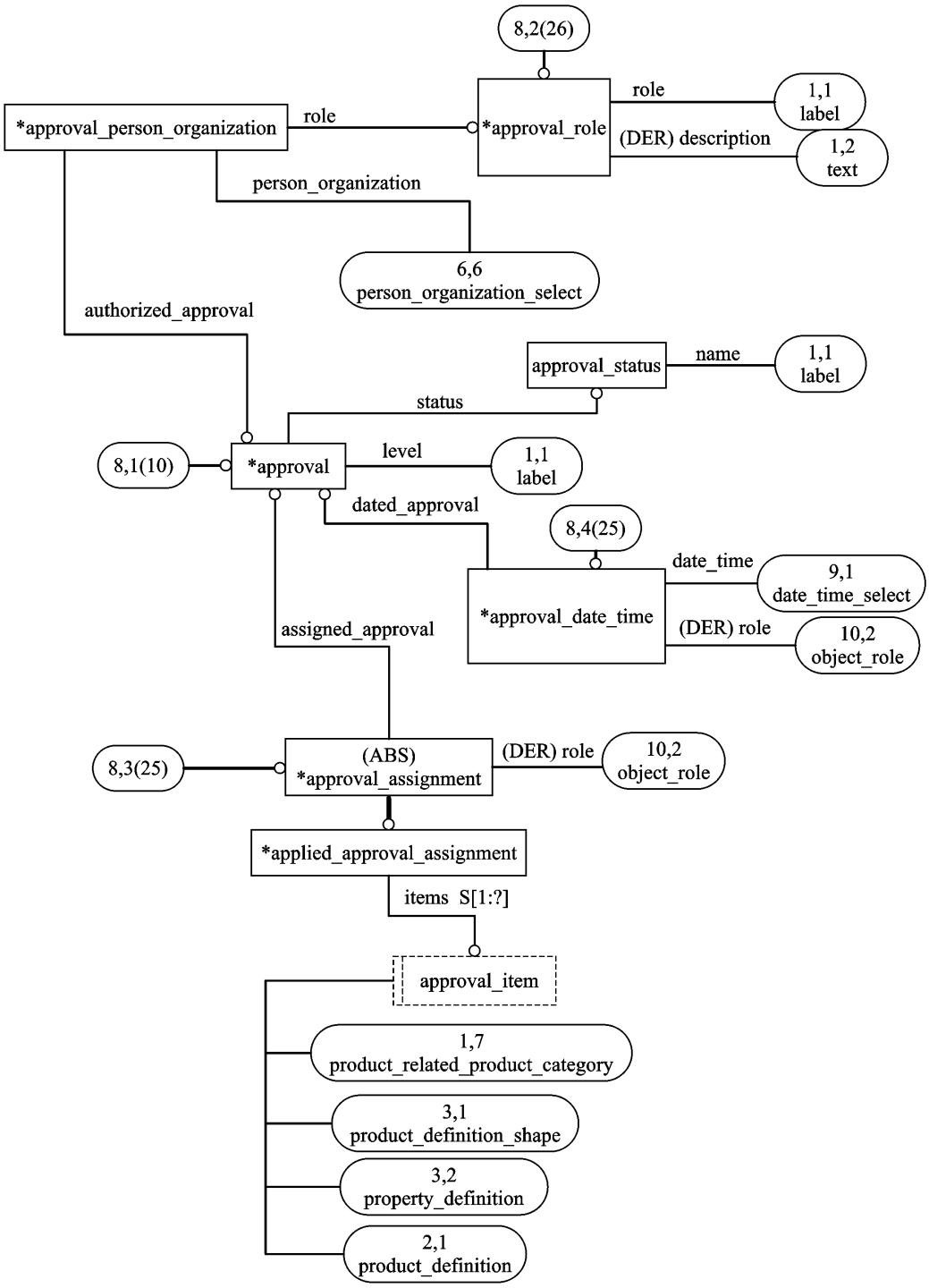


Figure H.8 — approval - AIM diagram 8 of 26 in EXPRESS-G

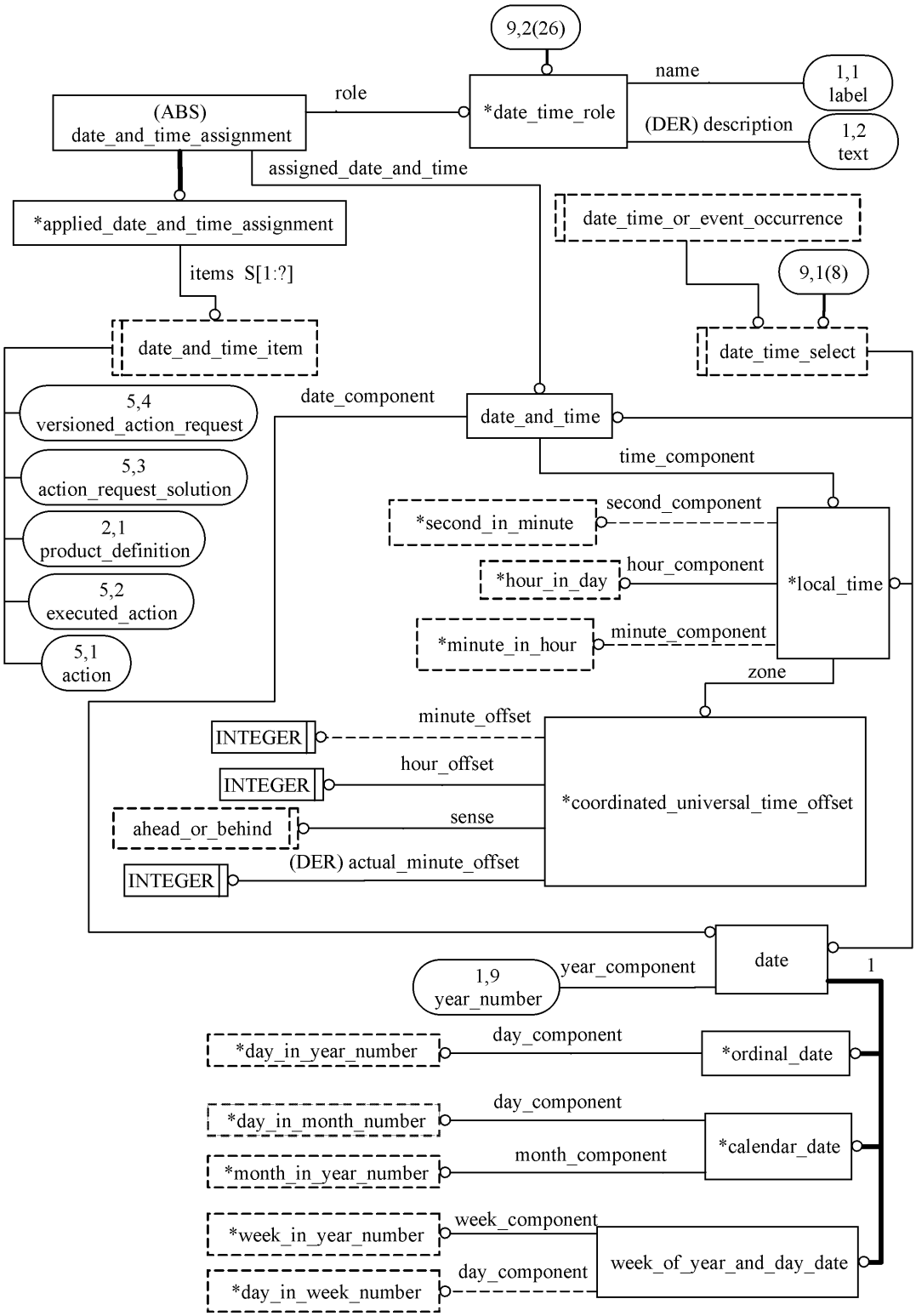


Figure H.9 — date and time - AIM diagram 9 of 26 in EXPRESS-G

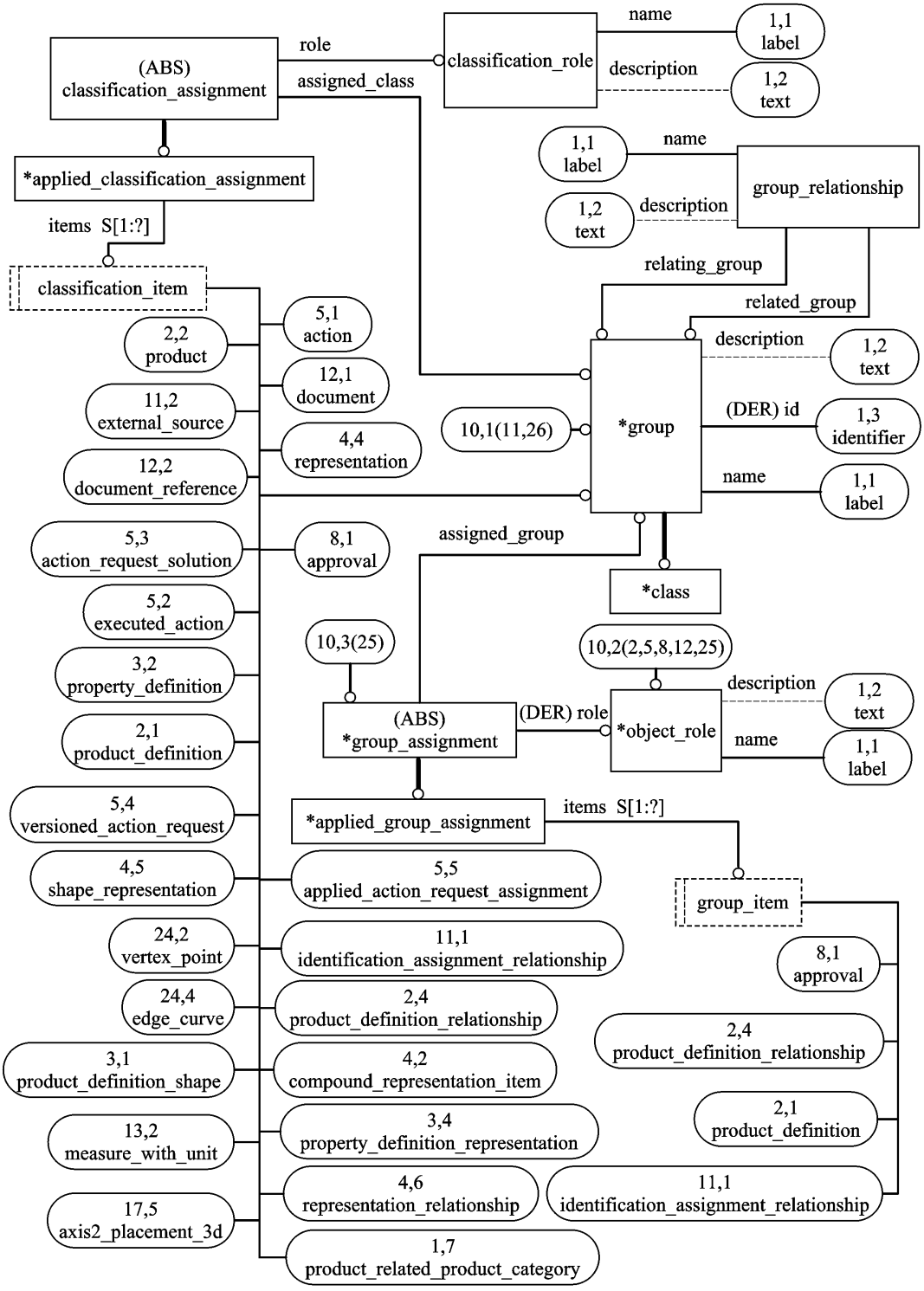


Figure H.10 — classification assignment and group - AIM diagram 10 of 26 in EXPRESS-G

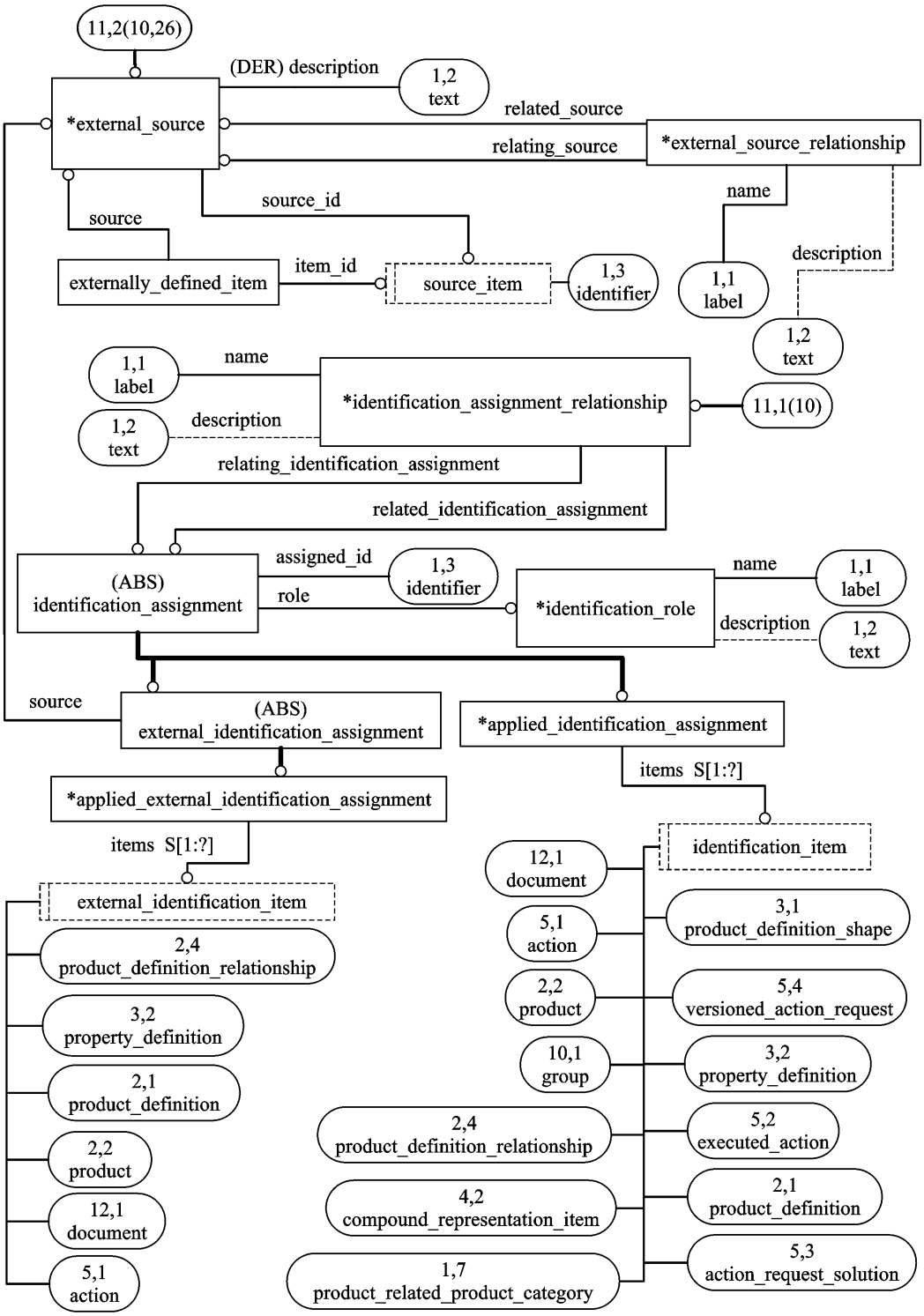


Figure H.11 — identification assignment external source - AIM diagram 11 of 26 in EXPRESS-G

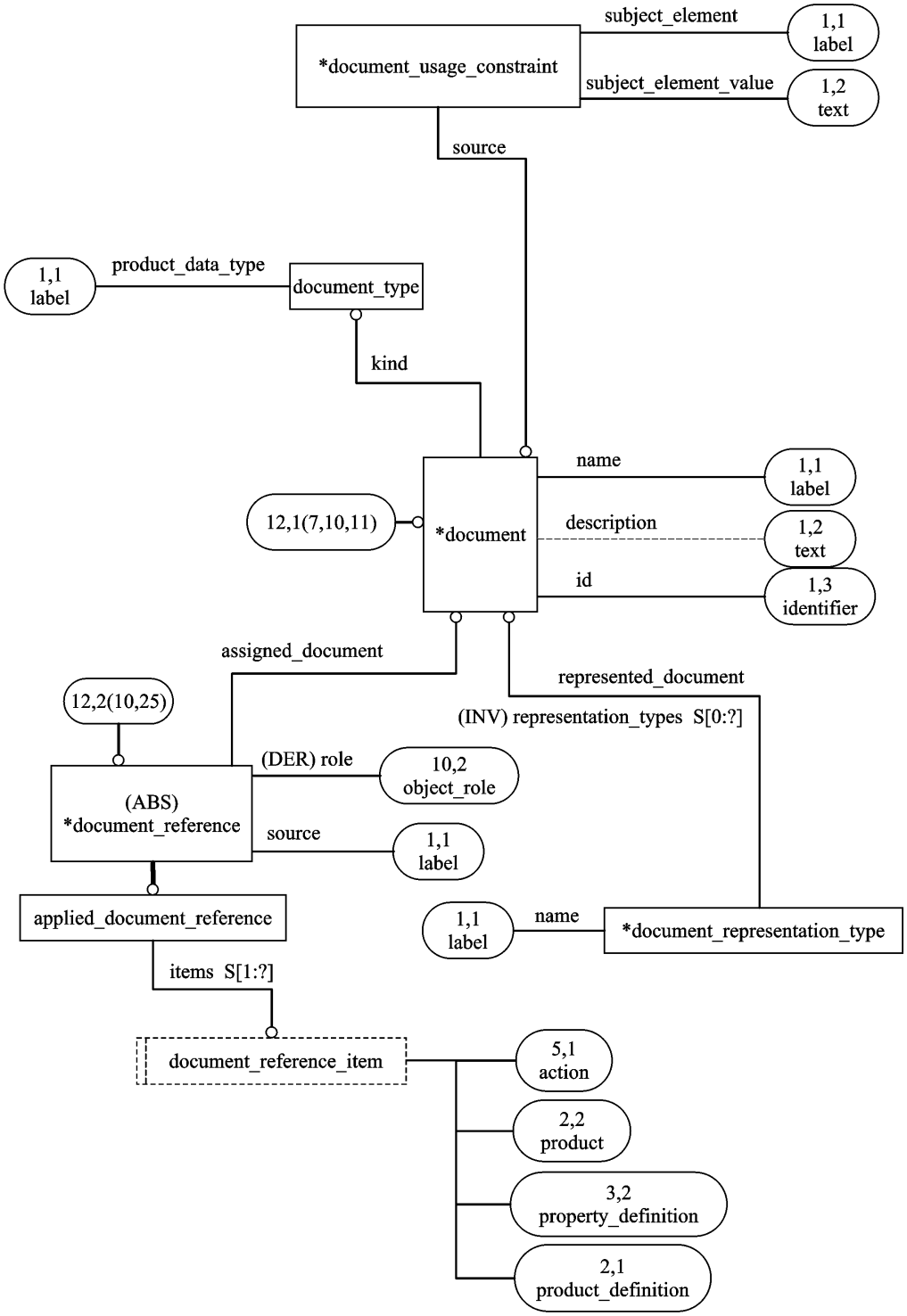


Figure H.12 — document - AIM diagram 12 of 26 in EXPRESS-G

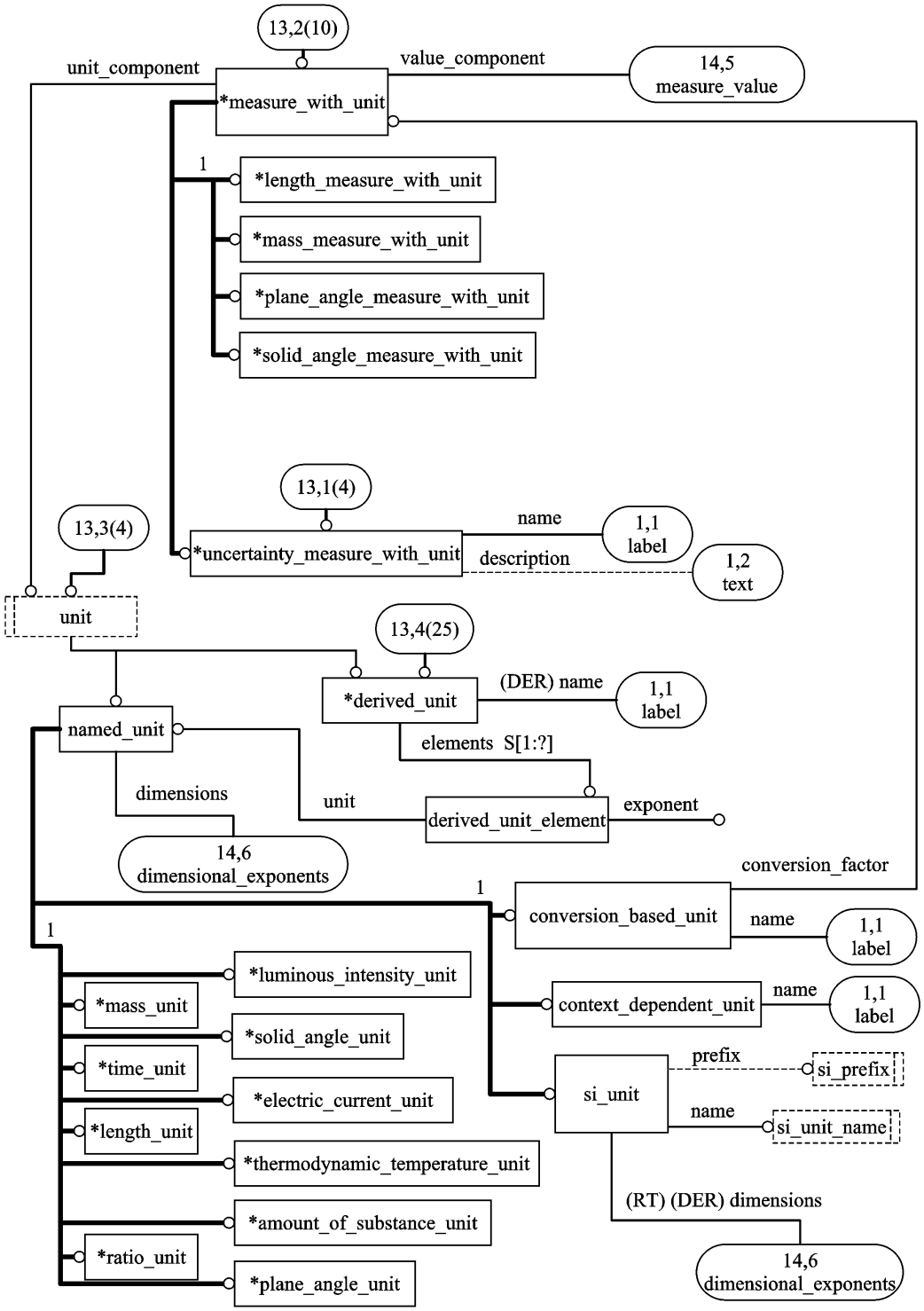
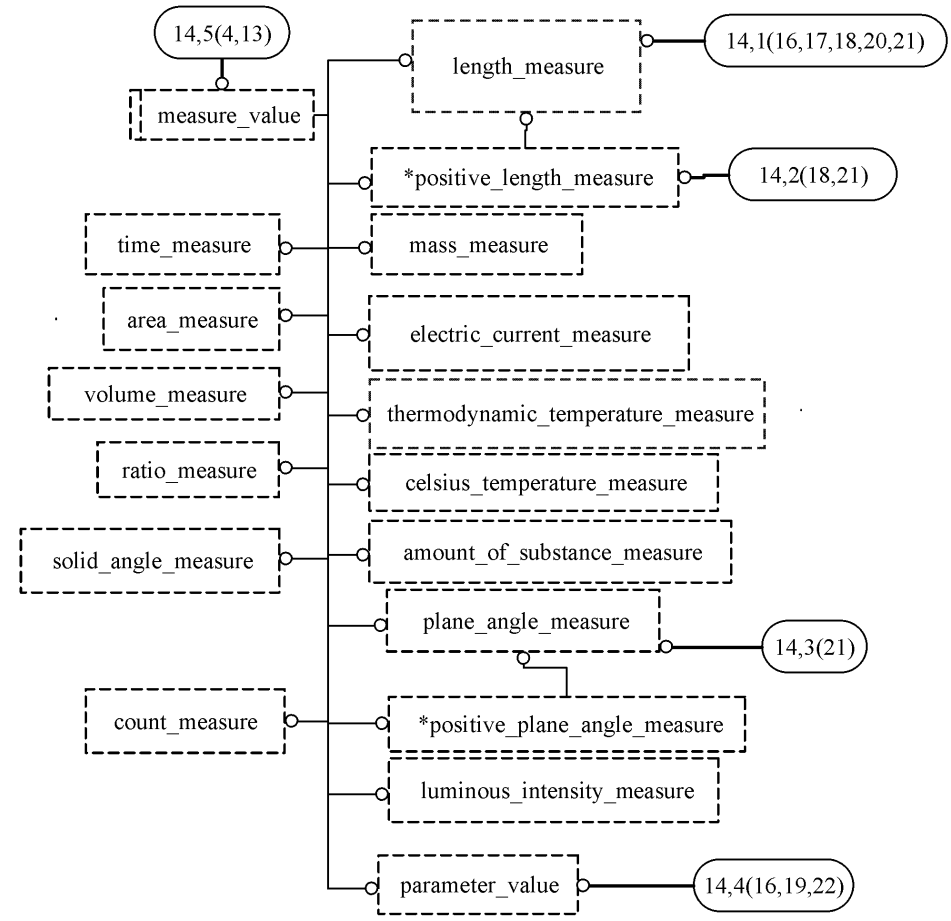


Figure H.13 — measure with unit - AIM diagram 13 of 26 in EXPRESS-G



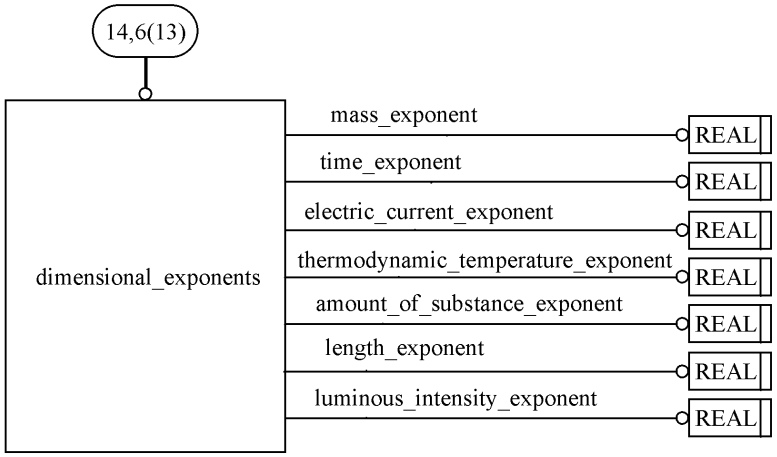


Figure H.14 — measure value - AIM diagram 14 of 26 in EXPRESS-G

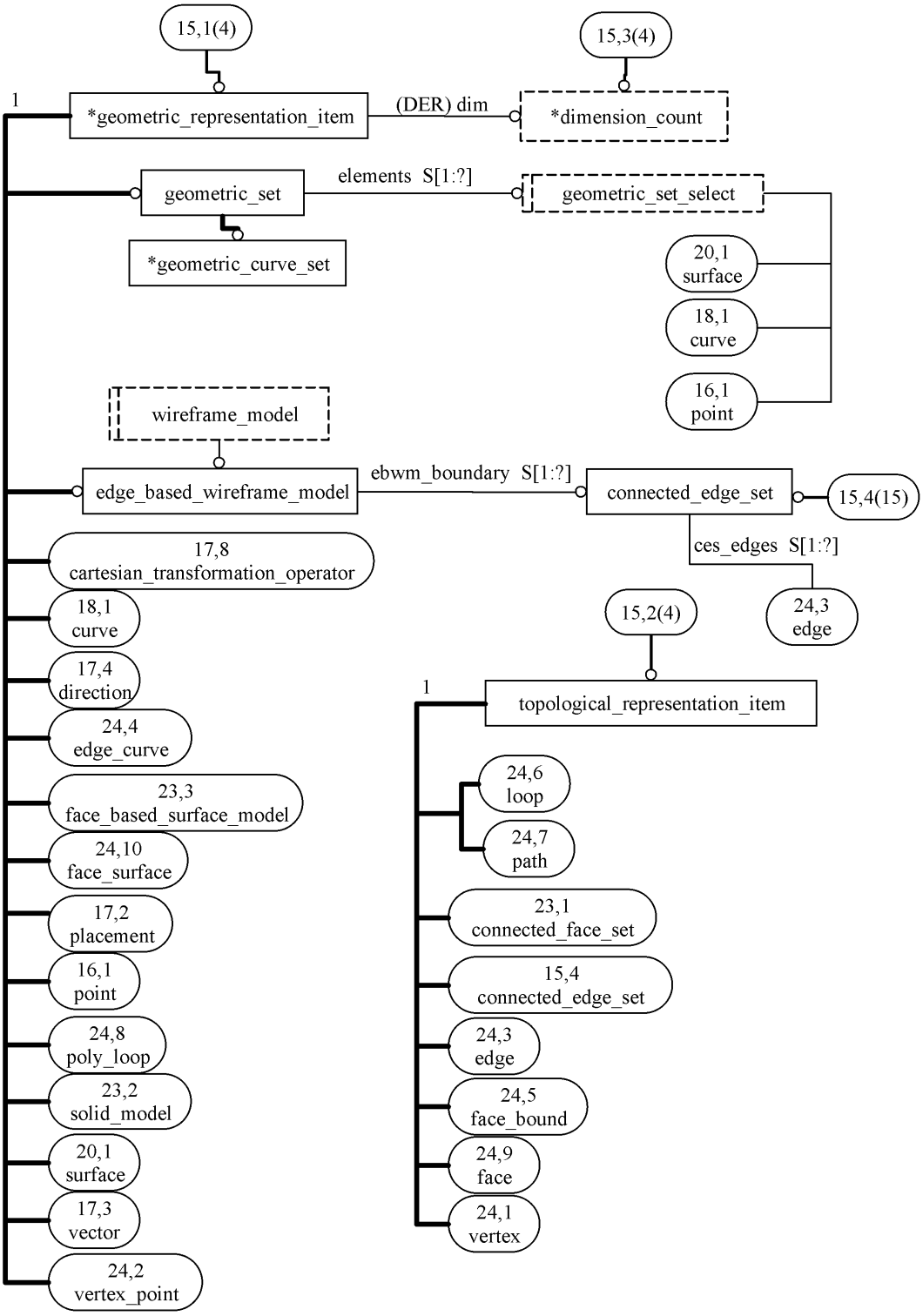


Figure H.15 — geometric and topological representation - AIM diagram 15 of 26 in EXPRESS-G

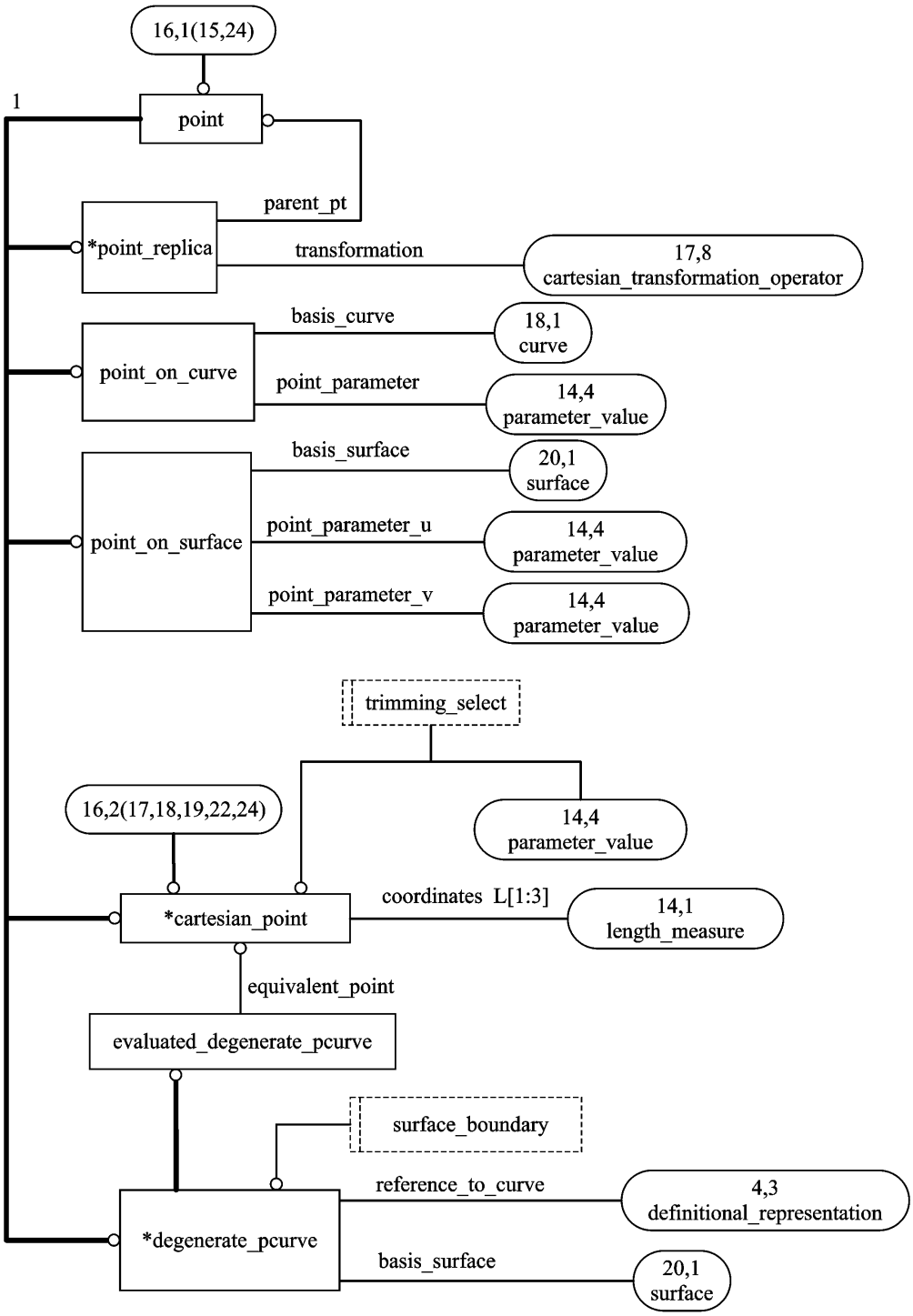


Figure H.16 — point - AIM diagram 16 of 26 in EXPRESS-G

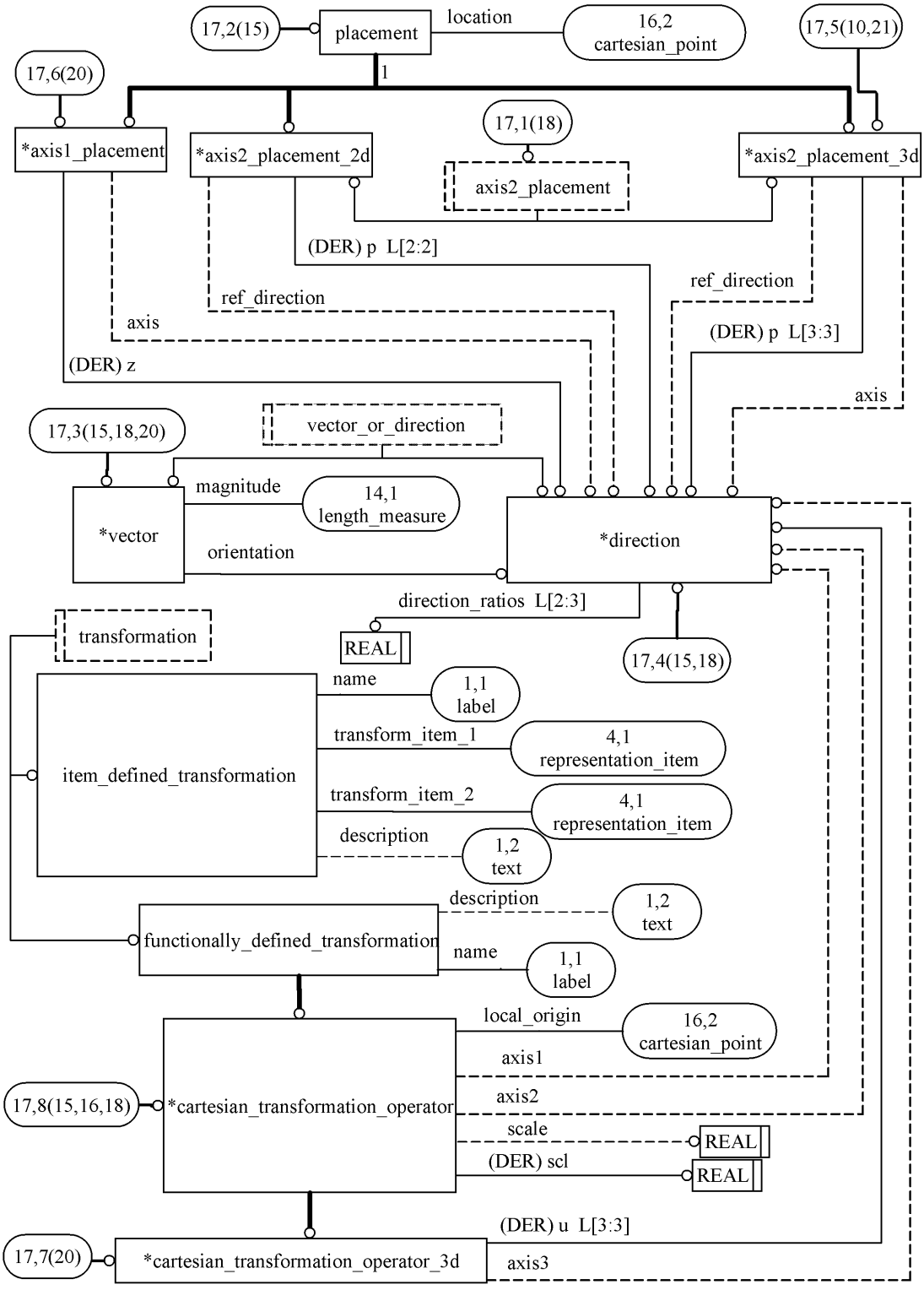


Figure H.17 — placement - AIM diagram 17 of 26 in EXPRESS-G

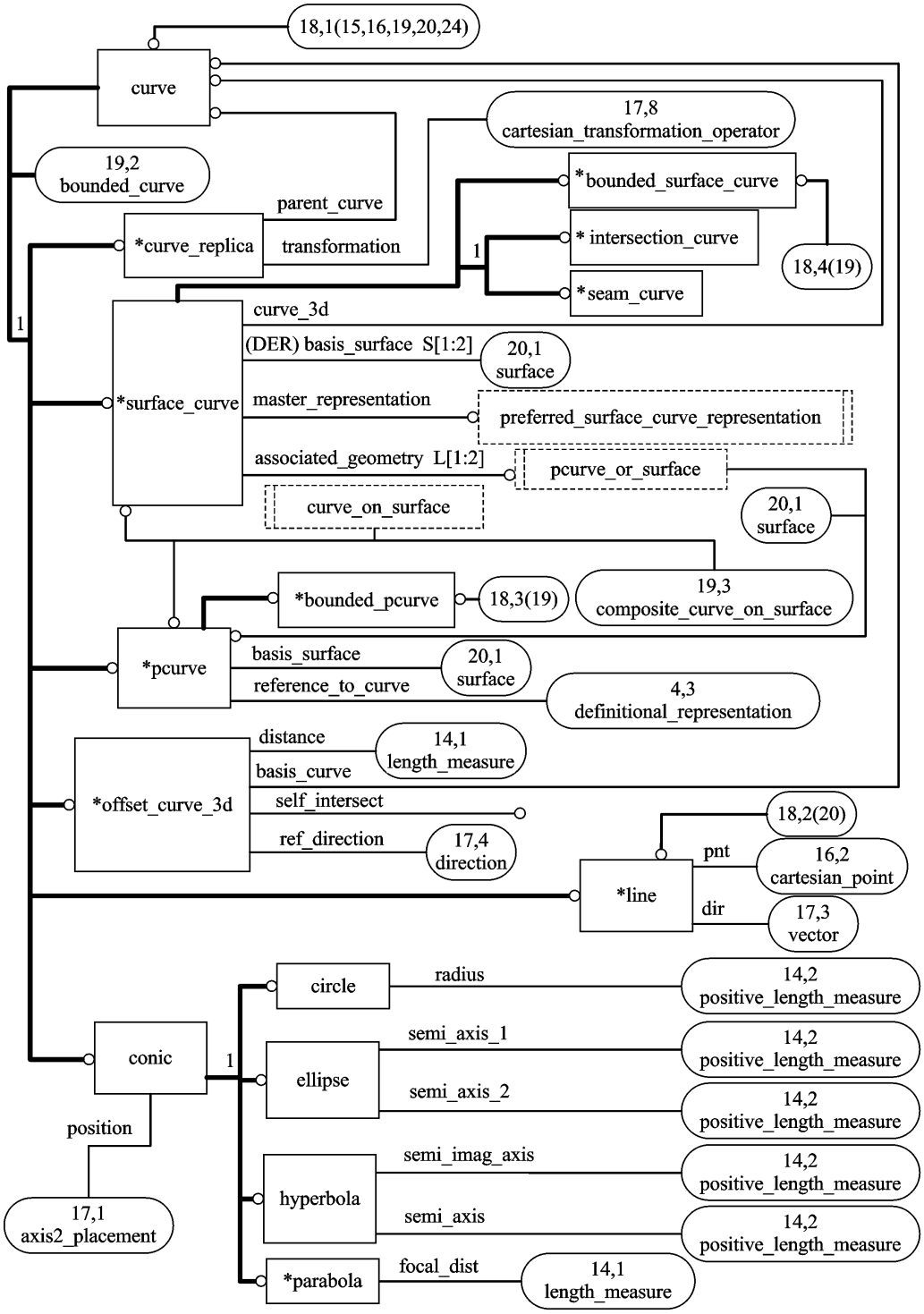


Figure H.18 — curve - AIM diagram 18 of 26 in EXPRESS-G

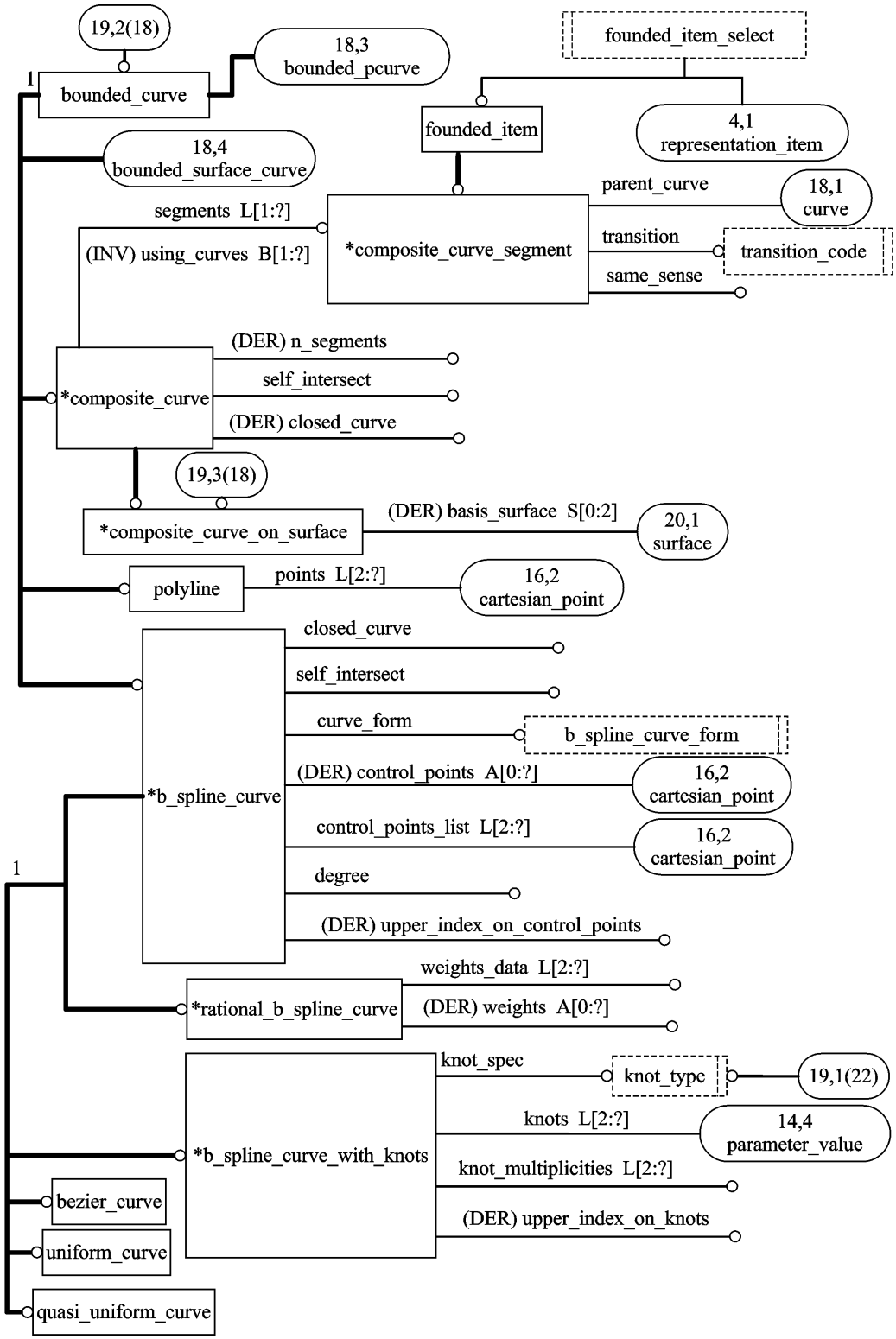


Figure H.19 — bounded curve - AIM diagram 19 of 26 in EXPRESS-G

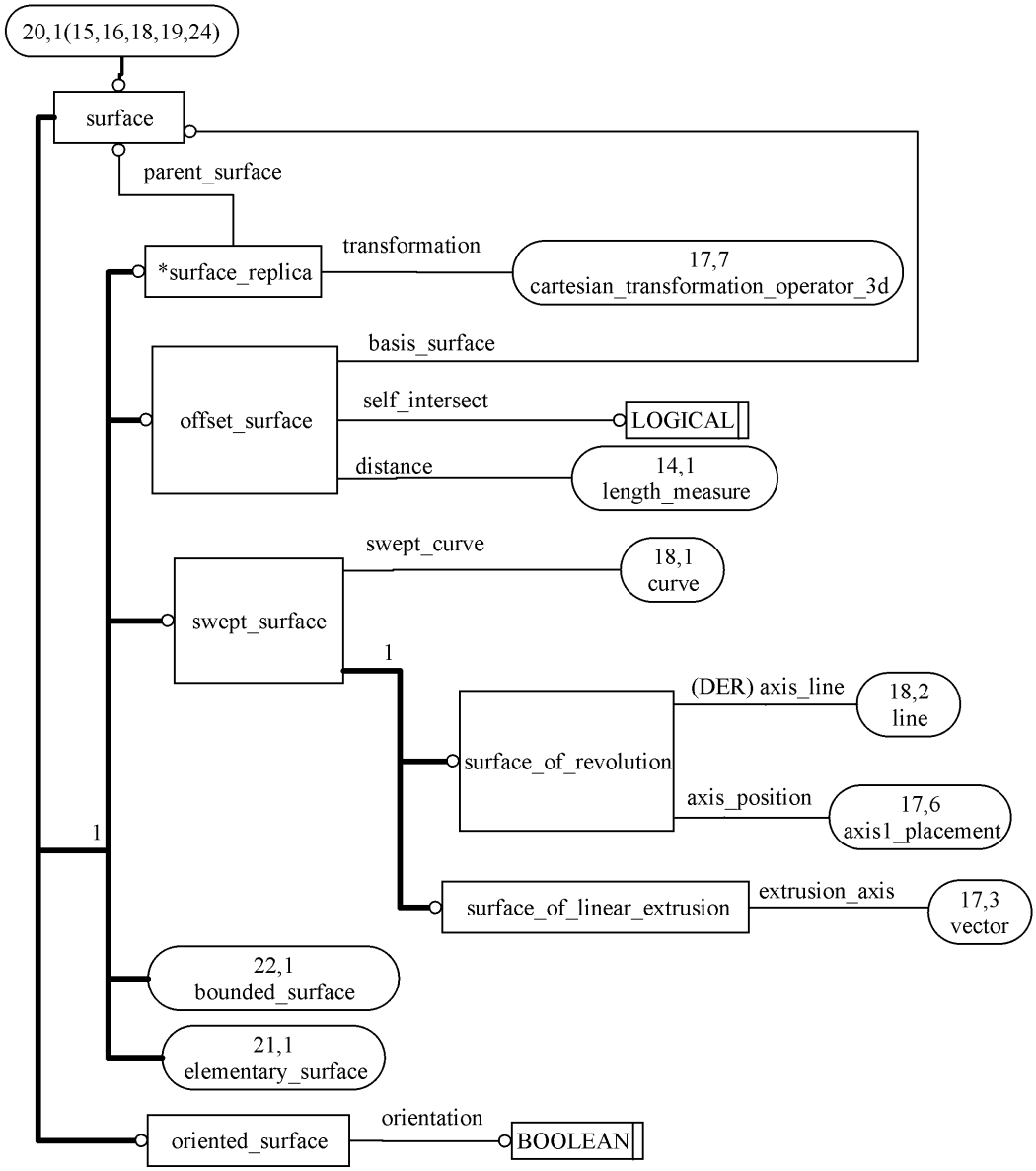


Figure H.20 — surface - AIM diagram 20 of 26 in EXPRESS-G

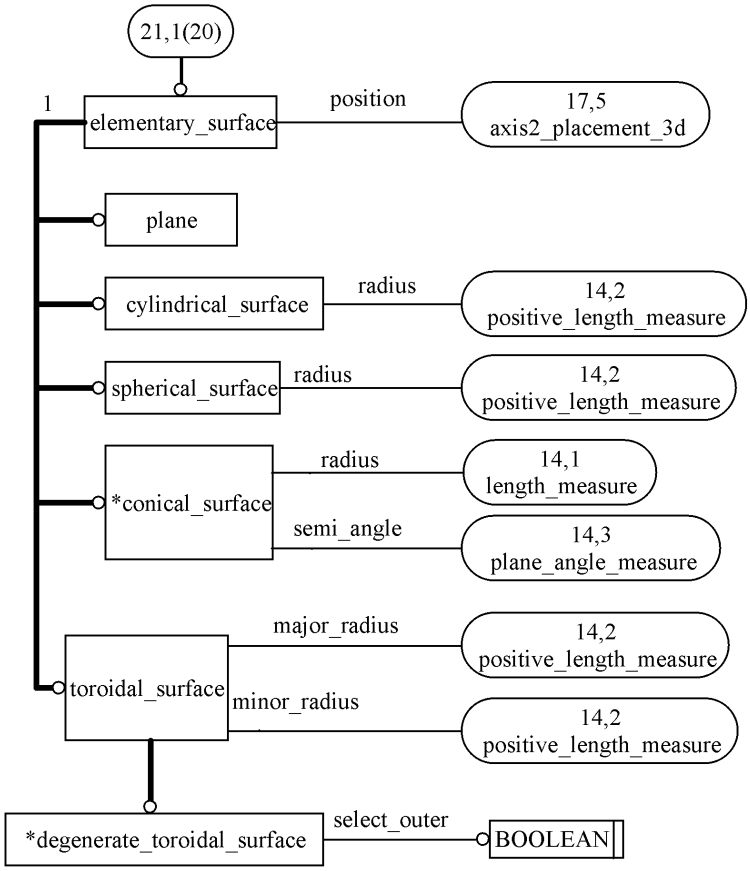


Figure H.21 — elementary surfaceAIM diagram 21 of 26 in EXPRESS-G

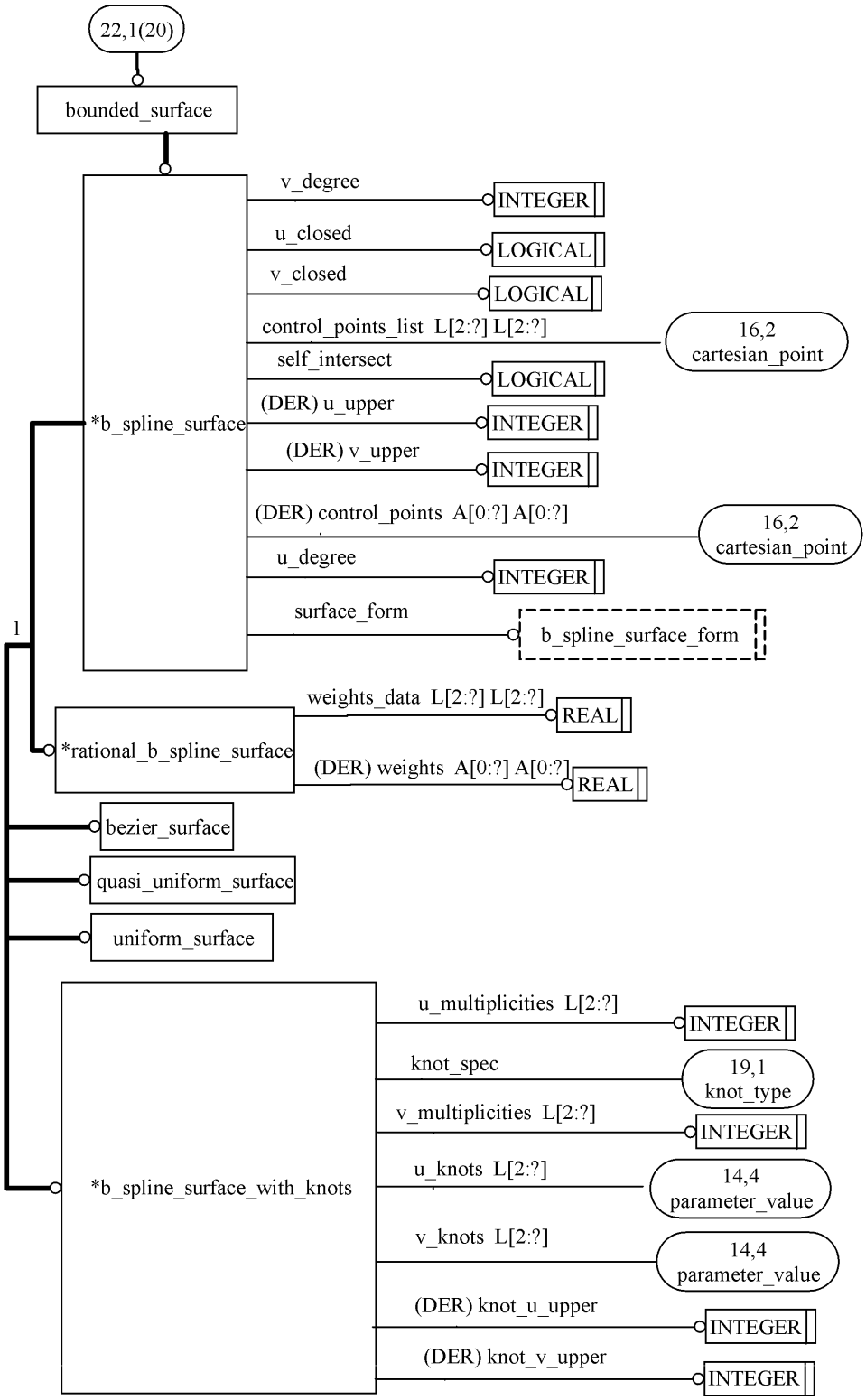


Figure H.22 — bounded surface - AIM diagram 22 of 26 in EXPRESS-G

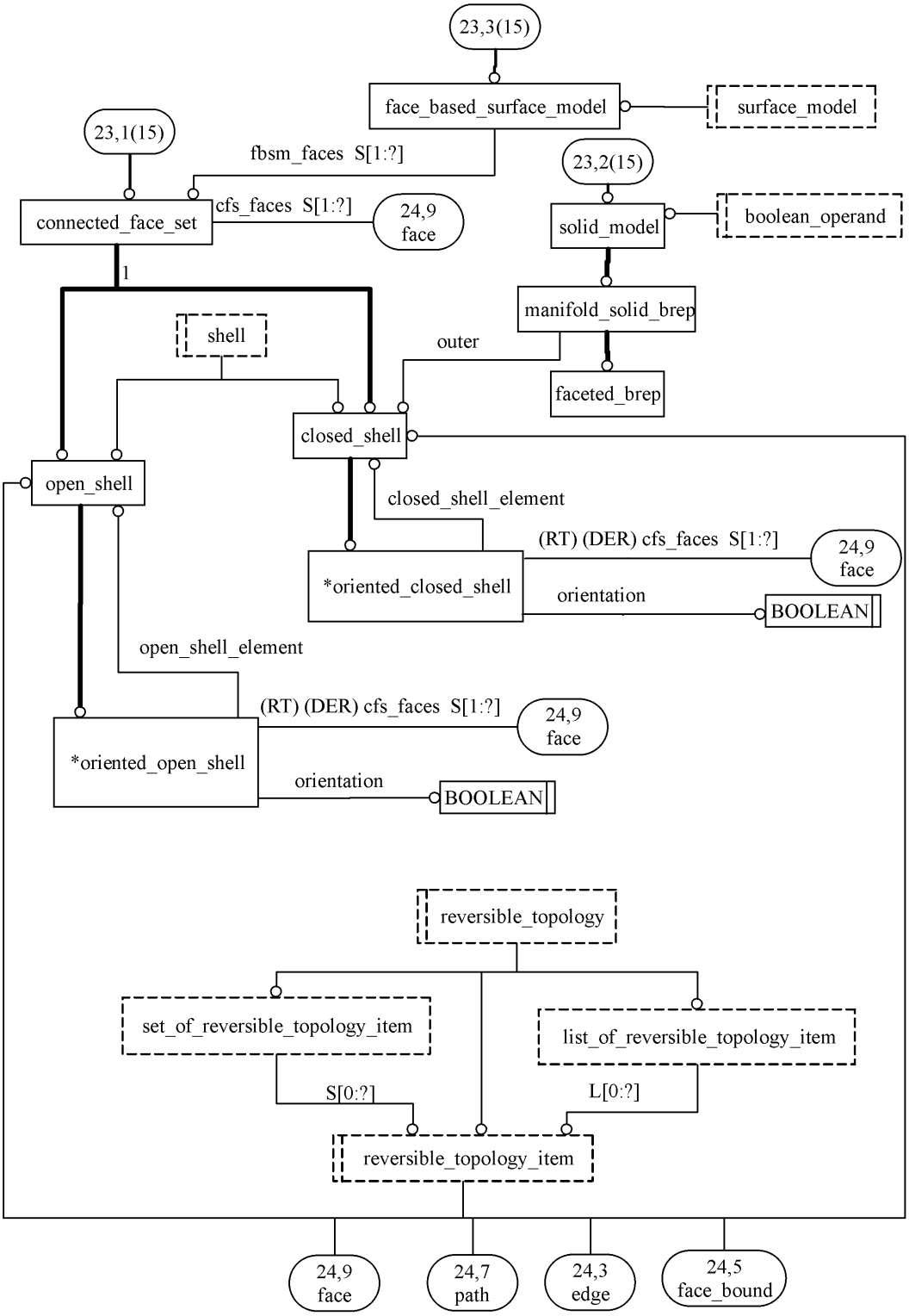


Figure H.23 — solid model and shell - AIM diagram 23 of 26 in EXPRESS-G

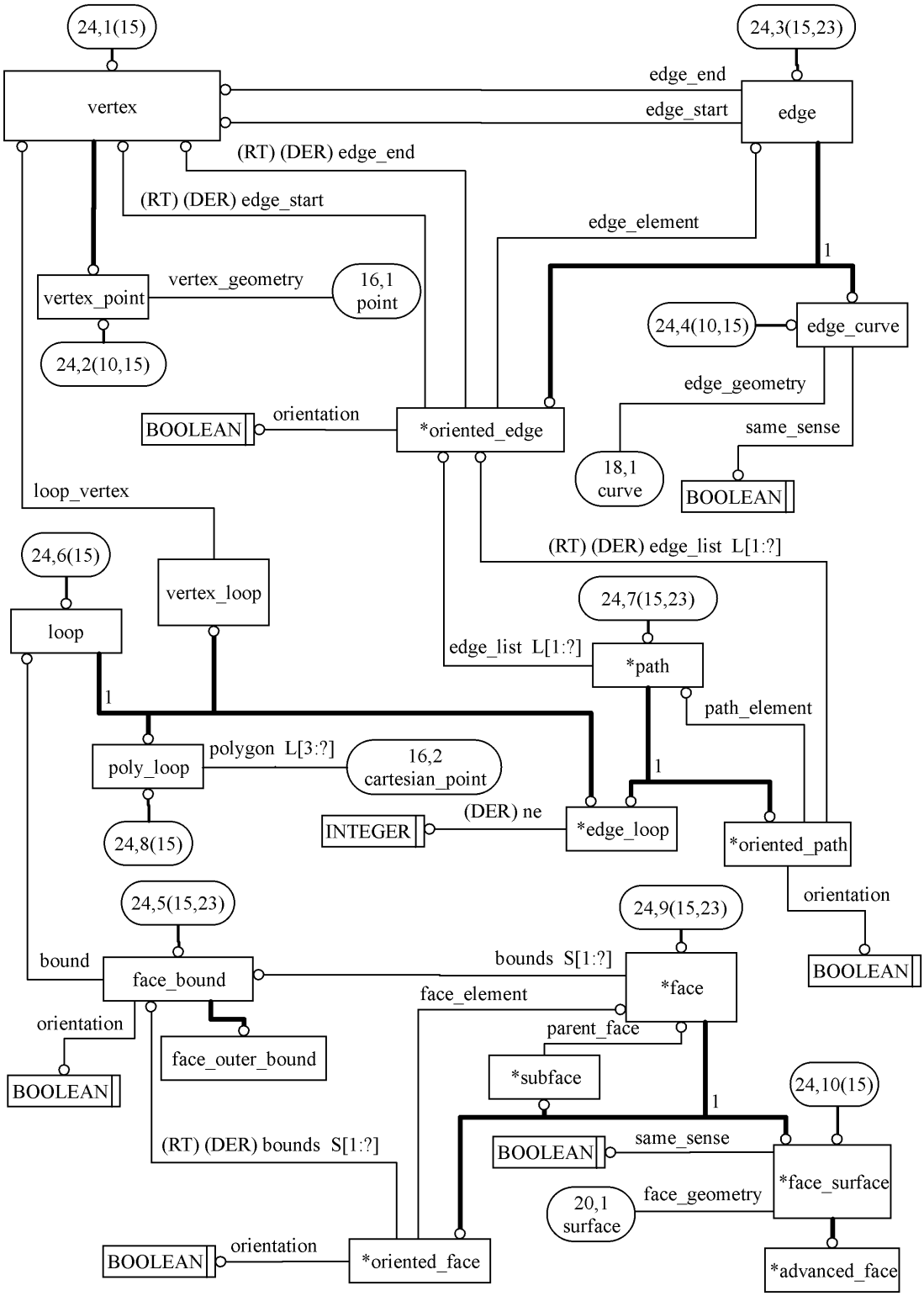
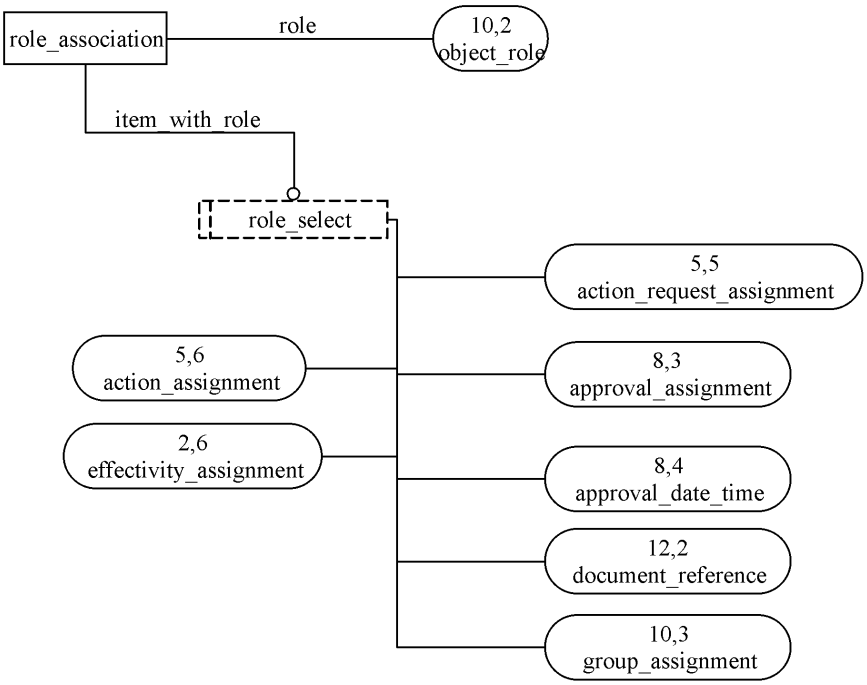


Figure H.24 — topology - AIM diagram 24 of 26 in EXPRESS-G



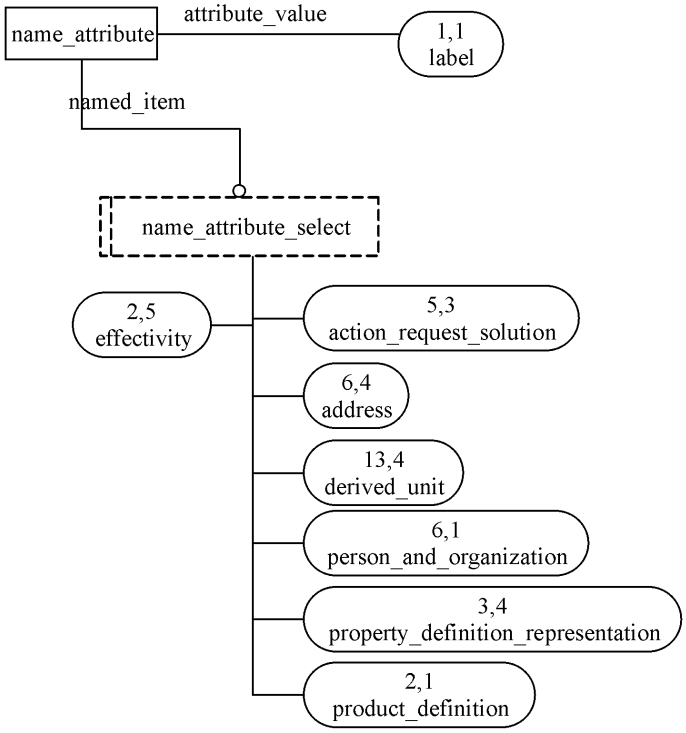


Figure H.25 — name attribute and role association - AIM diagram 25 of 26 in EXPRESS-G

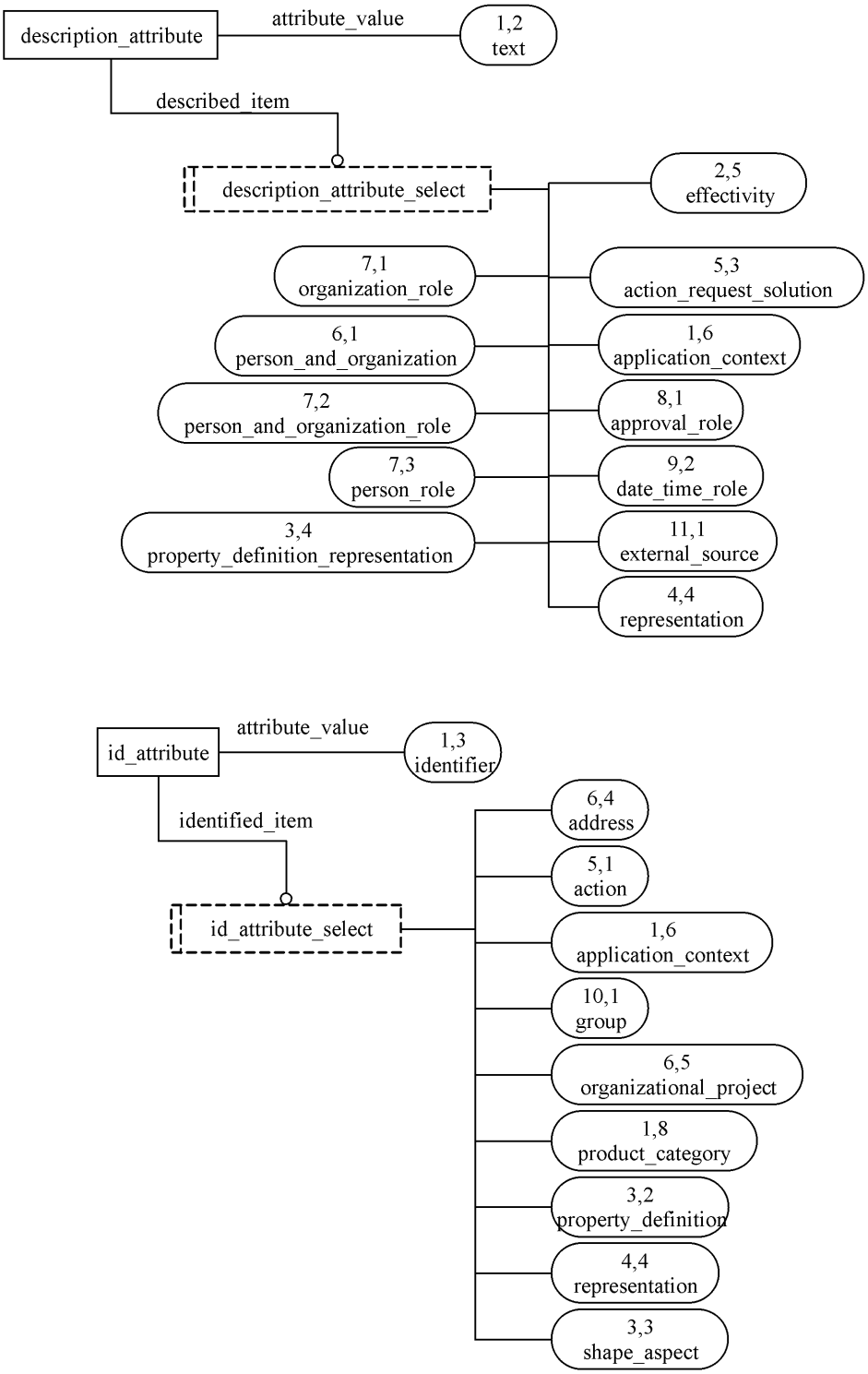


Figure H.26 — id and description attribute - AIM diagram 26 of 26 in EXPRESS-G

Annex J
(informative)

Computer interpretable listings

It also provides a listing of each EXPRESS schema specified in this part of ISO 10303 without comments or other explanatory text. These listings are 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 TC 184/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)

Application protocol usage guide

This chapter will answer some questions to the ARM of this part of ISO 10303.

K.1 How can a spacing position be related to a moulded form?

Use a local_co_ordinate_system_with_position_reference pointing to this moulded_form.

K.2 How can an externally defined moulded form be referenced?

The list of moulded forms in the ship moulded form includes an externally defined mouled form, identified by the external reference and in particular by the global id.

K.3 What happens with the coordinate systems when the ship is in water at a specific floating position ?

The ship has a fixed global coordinate system. For hydrostatic calculations a local coordinate system is defined which is fixed to the water surface. The parent coordinate system is the global coordinate system.

K.4 Usage guidlines and test case definitions

The usage guide is documented in the ESTEP AP 216 Test Case Definitions [12]. These guidelines describe the usage scenarios, and example test cases.

Annex L
(informative)

Technical discussions

L.1 The ship product model

The aim of STEP is the support of methods and functionality to develop product models. A product model is the unambiguous representation of a product, such as a ship, automobile, airplane or production plant, in a computer interpretable neutral format, throughout the lifetime of the product. The goal is to represent a product from the requirement definition and conceptual design stages through to production, maintenance and finally decommissioning and dismantling. The representation is intended to include all geometric and non-geometric data associated with the product including topology, geometry, functionality, strength, parameter and characteristics, condition status and service history.

STEP is based on the idea that the master representation of a product is related to its product structure. That means, to define a ship within STEP, basic identification information and ship structure information has to be provided. As part of this approach, geometry is taken as a particular type of representation of the ship and its structure. The consequence of this is that when the moulded form geometry is exchanged using STEP additional information on how each element represented by the geometry relates to a particular part of a particular ship is exchanged as well. This is different from existing and previous exchange standards where only the pure geometry is exchanged. This is the reason why STEP is product-based and not drawing or geometry-based. It is also the reason for using STEP as a set of standard descriptions for reusable product-based data, the total collection of which are referred to as the product data model. STEP has been developed in conjunction with its own methods, so that it can be implemented in file exchange or databases alike.

In view of the complexity of the ship it is unavoidable to subdivide the STEP product model for the ship into recognizable and usable parts; the consequence of this is an agreed upon Application Protocol planning model. This model divides the shipbuilding domain 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 L.1.

The key elements of the ship product model are:

— arrangements;

— moulded forms;

— mechanical systems (machinery, propulsion, cargo handling);

— structures;

— distribution systems (piping, HVAC, electrical, hydraulics/pneumatics);

— outfitting & furnishings;

— communication;

— combat systems;

— navigation;

— operation.

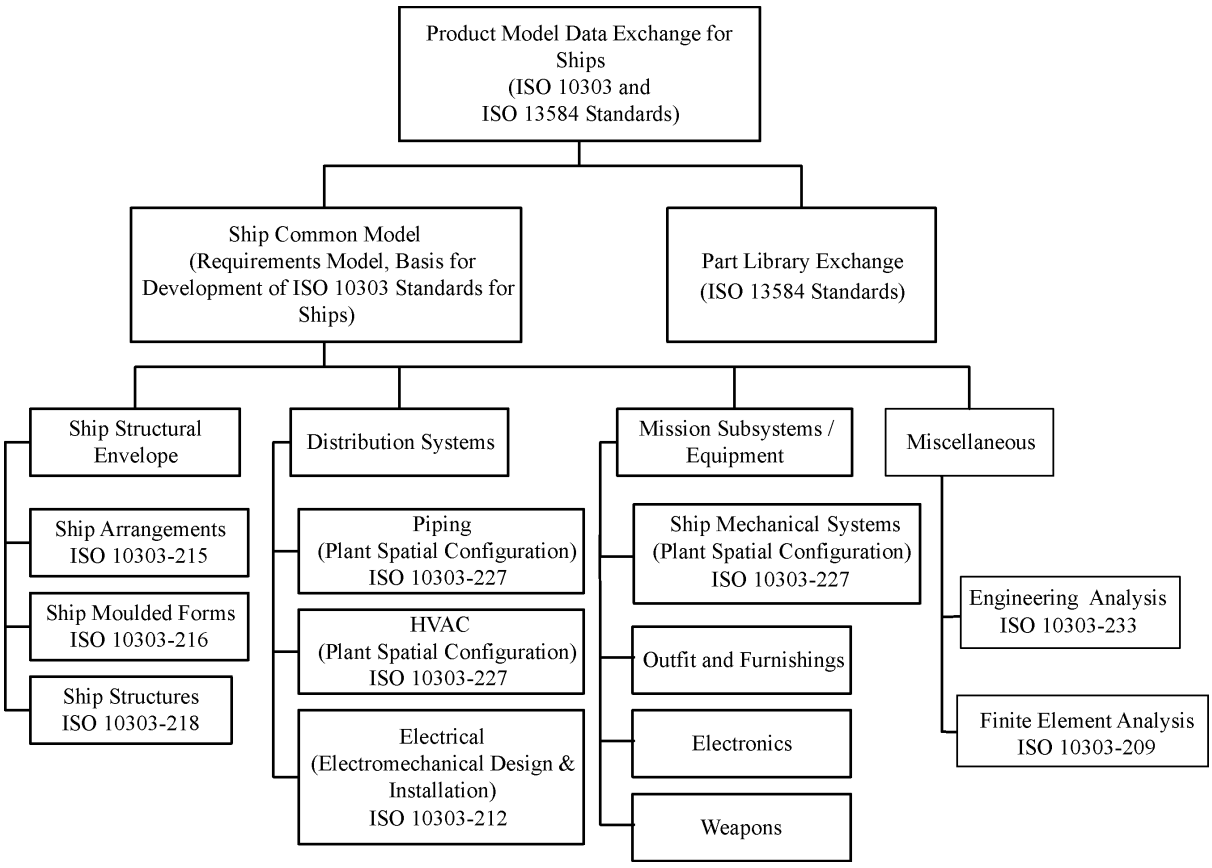


Figure L.1 — Ship Product Model

Each functional area of the ship product model is described by one or more different application protocols. The shipbuilding application protocols are:

— ISO 10303-215: Ship Arrangements;

— ISO 10303-216: Ship Moulded Forms;

— ISO 10303-218: Ship Structures.

The subdivision of the entire ship product model allows a distributed modeling approach. It is also possible then to start the modeling work with the functional areas reflecting the early stages of the life cycle of the ship, and validate these models before starting the modeling work at areas in the later life cycle stages. The life cycle of this part of ISO 10303 and the units of functionality that represent this life cycle is represented in Figure L.2.

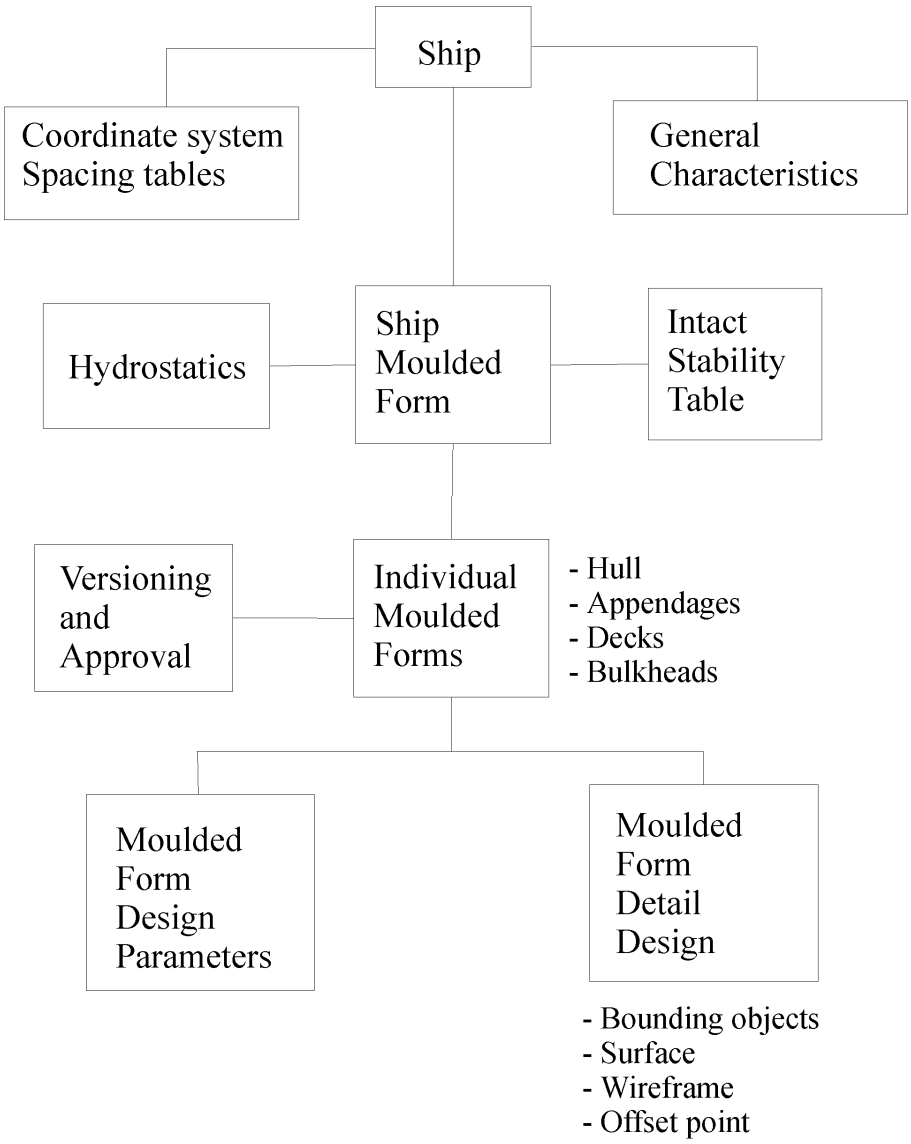


Figure L.2 — Structure of this part of ISO 10303

Application protocols are the only parts of STEP that are intended to be used for a data exchange, but

nevertheless 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 can the product model 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 (SCM), which is described in the following chapter.

L.2 The Ship Common Model

The Ship Common Model (SCM) defines a common framework and modeling 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 modeling framework, a set of domain (independent and re-usable) 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 (ARM) of the different ship APs.

The SCM is documented in the AP Development Guidelines for Shipbuilding [11]. These guidelines describe the:

— building block approach;

— modeling guidelines;

— Ship Common Model.

L.3 Modeling guidelines

The modeling guidelines can be seen as extensions to the EXPRESS language defined in Part 11 of STEP. They specify how to use the EXPRESS constructs for data modeling in the shipbuilding area. These modeling guidelines are also used by the ship common model.

All modeling work for the shipbuilding APs is done in the form of building blocks. A building block is an EXPRESS-based specification that is used for the definition of Units of Functionality UoF for an AP. A UoF may include one or several building blocks. A building block consists of three schemas:

— *import* schema providing an interface for those elements of other building blocks to be used by the model schema of this building block,

— *export* schema, making available those elements of the model schema intended to be used by other building blocks and

— *model* schema defining all the new elements for this building block.

In addition to the schemas, each building block has a building block header with some administrative information to allow automatic processing by the building block e-mail server. The modeling

guidelines are related to the building block concept. The current guidelines cover the following areas:

— building block name;

— size of building block;

— existing building blocks;

— commenting building blocks;

— restrictions on usage of EXPRESS;

— referencing STEP resources;

— cardinality constraints across building blocks;

— reference functions;

— reference instances across Application Protocols;

— importing building blocks into Application Protocols;

— compiling building blocks.

The guidelines for commenting building blocks are important for the application protocol, etc. These comments are used by tools to produce the clauses 4.2 and 4.3 of an Application Protocol (semi-) automatically.

L.4 Modeling framework

The modeling 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. The following three EXPRESS based building blocks define the framework:

— definitions;

— generic_product_structures;

— representation_resources.

The framework is a high level approach which splits the product model across the main constructs, namely

— items;

— definitions;

— representations.

whilst being linked via a number of generic relationships (see Figure L.3).

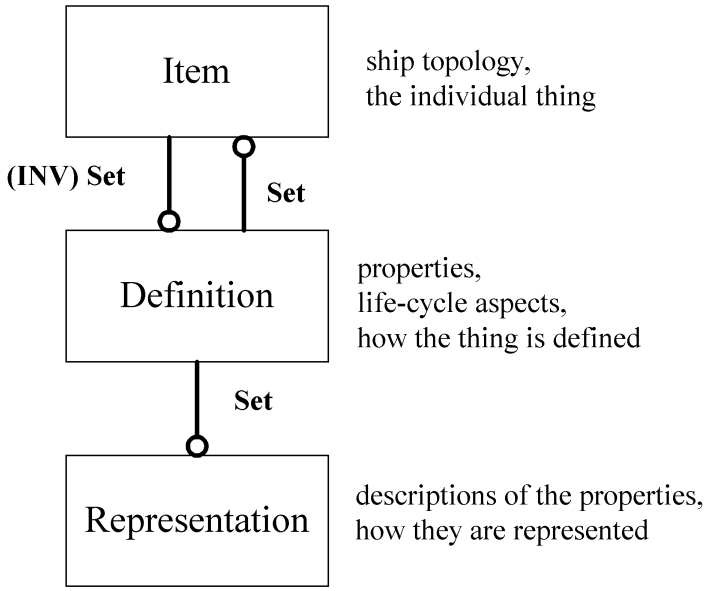


Figure L.3 — Modeling framework

Items represent a concept and can be seen as a placeholder. An item is a constant flag during the entire life cycle for a thing. New items in the shipbuilding APs are introduced by subtyping them from item.

Typical items in the shipbuilding APs are

— components of the ship (hull, superstructure, deck, rudder, propeller, etc...);

— equipment (pump, generator, main engine, pipe, etc...);

— steel structure elements (double bottom, frame, bulkhead, stiffener, etc...).

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. The definition can be changed for an item during the life cycle. These life cycle aspects are considered by the life cycle concept (see Figure L.4) where different definitions for different life cycle stages are introduced for an item. New definitions in the shipbuilding APs can be created by subtyping them from definition.

Every property of a concept and therefore, every definition of an item may be described in many different ways. A definition of a moulded form can be represented by a

— offset table representation;

— wireframe representation;

— surface representation;

— spacing position.

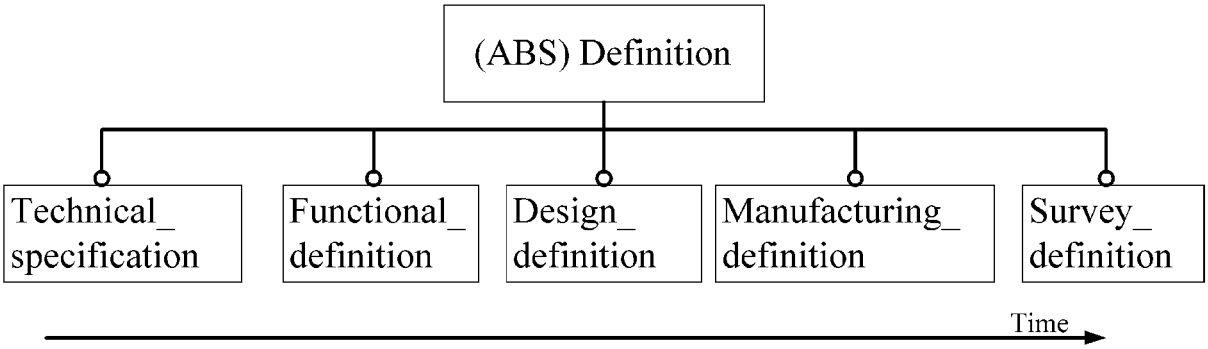


Figure L.4 — Life cycle concept

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 modeling framework can be restricted in the subtypes of each AP. This is done by redeclarations of the attributes. For example a moulded form design definition, a subtype of definition is defined for a moulded form, which is a subtype of item (see Figure L.5).

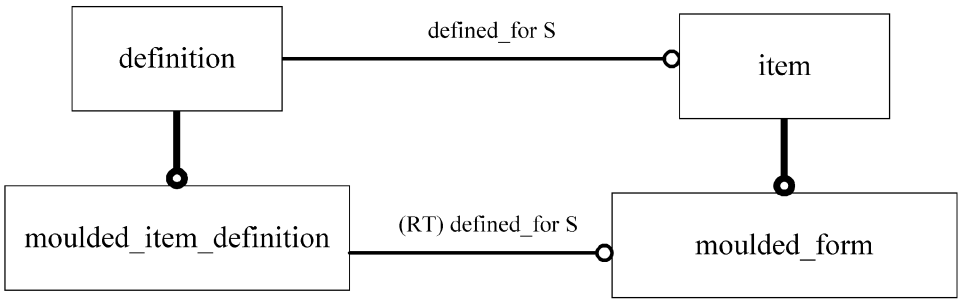


Figure L.5 — Redeclaration of attributes

L.5 Domain models

The intention of domain models is to create a new layer of generic elements underneath the main constructs of the modeling framework, with the possibility to restrict the attribute relations using redeclarations. These new elements introduced by the domain models are subtypes of the main constructs of the modeling framework.

The following domain models are part of the Ship Common Model:

— product structure;

— part;

— product structure by system;

— product structure by assembly;

— product structure by space;

— connectivity.

L.6 Common utilities

Common utilities are EXPRESS based building blocks which are specific for the shipbuilding Aps, but also general enough so that they can be used in different Aps.

Common utilities are currently available for

— ship general characteristics;

— configuration management;

— location concepts (global local co-ordinate system, spacing grid);

— basic geometry and topology;

— moulded form points;

— moulded form lines;

— moulded form surfaces;

— ships;

— materials;

— features;

— units;

— external reference/external instance reference.

For more information about the SCM and the modeling guidelines see the AP Development Guidelines for Shipbuilding.

L.7 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 4.2.88), or locally in the local_units attribute of the Definition object (see 4.2.23). Units shall be instantiable in either SI units or other systems of units.

There are many ARM objects that can define their own local units. This local unit mapping requires a property_defintion_representation (with name = "local units") to reference a representation with a global_unit_assigned_context. However many of these same ARM objects already have attributes that map to another representation that contains the complex entity geometric_representation_context AND global_unit_assigned_context. This complex instance is required by rule 5.2.4.114 - representation_has_global_unit_assigned_context. Also if the measure has a tolerance value associated with it, the complex instance will have the additional entity type global_uncertainty_assigned_context. Therefore if the ARM object requires local units, it is recommended that the units associated with this complex instance be the same as the units referenced via the local units attribute reference path.

Table L.7 lists the units that are used in the application interpreted model of this part of ISO 10303, corresponding to each ARM measure type.

Table L.7 — ARM measures and corresponding AIM measures and units

Application Object attribute reference	Application Interpreted Model measure and unit
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
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

Application Object attribute reference	Application Interpreted Model measure and unit
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
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
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.

Application Object attribute reference	Application Interpreted Model measure and unit
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. 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.
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.

Application Object attribute reference	Application Interpreted Model measure and unit
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.
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.

L.8 Globally unambiguous identifier usage

Globally unambiguous identifiers (GUID) have been incorporated as part of the Ship Common Model and are used by all the Shipbuilding Application Protocols. They are used to support the following incremental exchange scenarios:

— Separate exchanges of items, definitions, and representations within an AP while preserving the relationships between these application objects. For example, the first exchange may only identify the moulded_forms, by sending only Moulded_form entities, while the second exchange may contain the specific properties and representations, by sending Moulded_form and Moulded_form_design_definition entities.

— Separate exchanges of a single ship model using AP 216 while preserving relationships between selected AP 216 application objects. For example, the first exchange may include the hull Moulded_form entity and the second include the main deck Moulded_form entity whose boundary is defined by an External_instance_reference entity containing the globally unambiguous identifier of the hull.

— Separate exchanges of a single model using several shipbuilding AP's while preserving relationships between application objects in different AP's. For example, the first exchange may include a deck Moulded_form entity in AP 216 and the second include the main deck Plates in AP 218 whose underlying surface is defined by an External_instance_reference entity containing the globally unambiguous identifier of the deck moulded_form in AP216.

Globally unambiguous identifiers and incremental exchanges are important for shipbuilding for the following reasons:

— Volume of data in a ship design. It is simply not practical to exchange an entire ship design in one exchange. Typical data exchanges focus on specific areas of the ship or specific systems. It is important that the result of these multiple exchanges results in a ship model which preserves the basic relationships between shipbuilding elements

— Long design and production time. A ship design is developed over an extended period, as long as several years for military ships. Similarly, the production information is also developed over a similar period. It is important to exchange data at any point in the design or production cycle while preserving the basic relationships between shipbuilding elements.

— Instance based design. A ship design is composed of individual occurrences of parts at specific locations on the ship. It is important to uniquely identify and track each individual instance on the ship through design, production, and eventually to operation.

— Hull applicability. A class of ships may be designed similar to a single ship design, with hull-specific exceptions called out. It is composed of individual occurrences of parts at specific locations on the ships. It is important to uniquely identify and track each individual instance in the ship class through design, production, and eventually to operation.

— Concurrent design and production. Typically, production of a ship begins before the entire design is complete. Both the design and production information is assembled incrementally. Further, in order to manage changes, it is important to uniquely identify each individual instance on the ship through design and production.

— Co-design. Several design agents may collaborate on the design of a ship or ship class. This is feasible only if each company can identify the incremental changes made by the other partner.

— Co-production. Several shipbuilders may collaborate on the production of a ship or ship class. This is feasible only if each company can identify the incremental changes made by the other partner.

— Collaboration. A number of companies may collaborate in the design and production of a ship or ship class. This is feasible only if each company can identify the incremental changes made by each company.

L.8.1 Functional requirements

Since GUID's are used to support incremental exchanges, it is not possible to fully specify the functional requirements for GUID's in the Shipbuilding Application Protocols. The functional requirements for GUID's are as follows:

— The GUID must be globally unique and persistent, within a ship design, across ship designs and across companies. It must point to the instance that it designates or to nothing at all (as when the instance has been deleted).

— Each software system must assign persistent GUID's to each application object created or updated.

— Each software system must support access to the application object via GUID.

— Each software system must store the GUID of all application objects.

— Each software system must return same GUID as it received if application object was not changed.

L.8.2 Implementation constraints

A number of different implementation schemes and technologies can be used to support this notion of GUID. However, there are several implementation constraints, which should be considered:

— Must support interoperability across companies, applications/systems, and technologies. (The identifier should not be opaque but must expose the essential data elements needed for persistent identification. Opaque identifiers were specified in CORBA v1 and proved an inhibitor to interoperability across systems).

— Must support systems that use value-based keys as well as systems that use system-generated object identifiers. (Should not impose the requirement on value-based systems to store an additional attribute for system-generated identifiers.)

— Instance identifiers must contain globally unique type identifiers but also have additional requirements. Consequently, instance identifiers carry more information than type identifiers (such as PLIB BSU's).

— Identifier should be character-based and the syntax should be specified so that industry standard tools and algorithms can be used.

— Identifier should support existing Part 21 and SDAI usage as well as anticipated Part 28 XML usage.

— Identifiers should be based on unique identification of single data objects in the software application. The Ship Common Model typically uses four or more objects to describe a single data object: item, design_definition, functional_definition, and manufacturing_defintion. While it is important to uniquely identify each STEP application object, it is also important to the software application to uniquely identify the data object, which generates these STEP application objects.

L.8.3 ISE recommended GUID format

The Integrated Shipbuilding Environment (ISE) project has adopted and is implementing translators based upon the following GUID format.

The ARM attribute Global_id.id (AIM attribute applied_classification_assignment.id) is written as the string in the following format:

‘<XX_Key x_href=YY Id=WW x-owner=ZZ x-rev=AA />’

Argument definition:

XX_Key - XX is name of related ARM entity, e.g. Moulded_form_function_definition. This argument is required. The first letter of application object is upper case, rest lower case. Suffix = "_Key".

x-href=YY - YY is the URL of SOAP server and/or database This argument is required.

Id=WW - WW is a string which identifies object. Id is required only for definable_objects. The Id is one of two types:

Value (transparent usage- a unique value which identifies this instance
GUID (opaque usage) - a system generated identifier. Must be prefixed by "guid:"

x-owner=ZZ - ZZ is a string which identifies the object. This argument is required only for Definition objects. Same string as Id of related definable_object.

x-rev=AA - AA is revision number. This argument is required only for Definition objects. Same string as Definition.version_id attribute.

The combination of (XX_key, x-href, Id) is unique for definable_objects

The combination of (XX_key, x-href, x-owner, x-rev) is unique for Definitions

EXAMPLE 1 The following examples illustrate this usage for the related moulded_form entities in AP 216:

```
'<Moulded_form_Key
x-href="//joshua.ingr.com"
Id="guid:123-123-123" />'

'<Moulded_form_functional_definition_Key
x-href="//joshua.ingr.com"
x-owner="guid:123-123-123"
x-rev="A"/>'

'<Moulded_form_design_definition_Key
x-href="//joshua.ingr.com"
x-owner="guid:123-123-123"
x-rev="A"/>'
```

EXAMPLE 2 A Part entity is defined in EXPRESS and is instantiated in a repository whose locator is "http://www.acme.com/stepdataserver". The following example shows how its persistent identifier would be encoded and, then, represented in a Part 21 file:

```
SCHEMA part_schema;

ENTITY Part
  Name : STRING;
  Description: STRING;
END_ENTITY;

END_SCHEMA;
```

The globally-unique persistent identifier for an instance of this type would be encoded as:

```
<Part_Key  href = "http://www.acme.com/stepdataserver"  Id = "P1"  />
```

This example illustrates the three essential data components for globally-unique persistent instance identification: a globally unique type name, a globally-unique instance locator (repository), the locally unique identifier (key) within that locator (repository).

The globally unique type name is the combination of the EXPRESS schema and type names mapped according to ISO10303-28. The EXPRESS SCHEMA name is associated with the type's namespace ("urn:iso10303:osb/part_schema"); and the EXPRESS ENTITY name is captured as the qualified element name with the string '_Key' appended (p:Part_Key). The xlink:href attribute captures the locator of the repository which manages the instance. This includes the authority (company) and establishes the global uniqueness of the identifier. The remaining elements capture the local identification attribute names and values (Id = "P1") that are guaranteed to be unique within the repository. The recommended name if there is a single local identifier is Id.

A special attribute, x-rev, may be included to expose version information about the referenced instance. The rule is that if the x-rev is present it signifies a specific relationship of this instance to other instances whose keys have the same values for all fields besides x-rev. Each such instance represents different versions of the same object; the order of the versions is application-dependent.

A system that uses system-generated object identifiers (such as guid's) would use the guid string as the value of the first key attribute.

<p:Part_Key xmlns:p = "urn:iso10303:osb/part_schema" x-id = "_10"
xlink:href = "http://www.acme.com/stepdataserver"
Id = "guid: BDA4A1BA-110C-11d0-8CC3-0080F394BA32" x-rev = "A"/>

The globally-unique persistent identifier would appear in a STEP file as:

#10 = Part('P1','Description for this part');
#20 = applied_identification_assignment(
'<p:Part_Key xmlns:p = "urn:iso10303:osb/part_schema"
xlink:href = "http://www.acme.com/stepdataserver" Id = "P1" x-re v= "A"/>',
#21,#10);
#21 = identification_role('xml_guid',' ');

Bibliography

[1] ISO 10303-209:2001, *Industrial automation systems and integration — Product data representation and exchange — Part 209: Application protocol: Composite and metallic structural analysis and related design*

[2] ISO 10303-212:2000, *Industrial automation systems and integration — Product data reresentation and exchange — Part 212: Application protocol: Electrotechnical design and installation*

[3] ISO 10303-215:—[2)], *Industrial automation systems and integration — Product data representation and exchange — Part 215: Application protocol: Ship arrangement.*

[4]ISO 10303-218:—[2)], *Industrial automation systems and integration — Product data representation and exchange — Part 218: Application protocol: Ship structures.*

[5] ISO 10303-227: —[3)], *Industrial automation systems and integration — Product data representation and exchange —* Part 227: *Application protocol: Plant spatial configuration.*

[6] ISO 10303-233:—[2)], *Industrial automation systems and integration — Product data representation and exchange —* Part 233: *Application protocol: Systems engineering data representation.*

[7] ISO 10303-234:—[2)], *Industrial automation systems and integration — Product data representation and exchange —* Part 234: *Application protocol: Ship Operational logs, records, and messages.*

[8] ISO 13584 – 42:1998, *Industrial automation systems and integration – Parts library – Part 42: Description methodology: Methodology for structuring part families.*

[9] *IDEF0 Federal Information Processing Standards Publication 183, Integration Definition for Function Modeling (IDEF0),* FIPS PUB 183, National Institute of Standards and Technology, December 1993.

[10] Lewis, Edward J., Editor, *"Principles of Naval Architecture, Second Revision"*, The Society of Naval Architects and Marine Engineeers, 1988.

[11] *AP Development Guidelines for Shipbuilding*, ISO TC184/SC4/WG3 N701, 19th October, 1997

[12] *ESTEP AP 216 Test Case Definition, Rev. C,* ISO TC184/SC4/WG3 N1143, 18th December, 2001

2) To be published.

3) To be published. (Revision of ISO 10303-227:2001).

Index

ICS 25.040.40

Price based on 946 pages