INTERNATIONAL STANDARD

ISO 10303-209

First edition
2001-10-01

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

Part 209:
Application protocol: Composite and metallic structural analysis and related design

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

Partie 209: Protocole d'application: Analyse structurelle composite et métallique et conception associée

Reference number
ISO 10303-209:2001(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.ch
Web www.iso.ch

Printed 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 3.

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

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

International Standard ISO 10303-209 was prepared by Technical Committee ISO/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, integration application resources, application protocols, abstract tests suites, application interpreted constructs, and application modules.

This part is a member of the application protocol series.

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

<http://www.nist.gov/sc4/editing/step/titles/>.

Annexes A, B, C, D, and E form a normative part of this part of ISO 10303. Annexes F, G, H, J, and K are for information only.

Introduction

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

This part of ISO 10303 is a member of the application protocol series. This part of ISO 10303 specifies an application protocol (AP) for the exchange of computer-interpretable composite and metallic structural product definitions, including product shape, associated finite element analysis (FEA) models and analysis results, and the material properties of these products.

The shape of a composite or metallic product definition includes the part and its constituents, including any ply shapes necessary for FEA node and mesh generation of boundary definitions. This information is suitable for the automated generation of composite material properties and geometric properties for finite elements. The shape definitions for design and analysis are each independently configuration controlled.

The finite element model idealizes a product or aspects of a product so that it may be analyzed to validate the structural performance and structural integrity of a product.

Finite elements of homogenous (isotropic) metallic and adhesive material properties are treated in this part of ISO 10303 as a subset of anisotropic composite material response. The differences between these two material response idealizations are: 1) a simplified material response, and 2) having no associated composite constituent information.

Assembly information provides the relationships necessary to identify analysis boundary conditions, and when combined with part geometry, topology, and finite element analysis output, provides the input necessary for detail analyses such as those for fastened structural joints.

This part of ISO 10303 satisfies the need for exchange of information between the iterative design and analysis stages of the product life cycle. Product configuration information provides the audit trail necessary to control the designed shape, its associated finite element model, and any related analysis shape information during these iterative stages of the product life cycle.

This application protocol defines the context, scope, and information requirements for the exchange of the information necessary to perform the design through analysis stages of the life cycle of composite and metallic structural parts, 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. Note that the definitions and EXPRESS provided in the integrated resources for constructs used in the AIM may include select list items and subtypes which are not imported into the AIM. The expanded listing given in annex A contains the complete EXPRESS for the AIM without annotation. A graphical representation of the AIM is given in annex H. Additional requirements for specific implementation methods are given in annex C.

Additionally, this application protocol enumerates the conformance requirements which identify the implementation options for the abstract test suite. This application protocol may be implemented as a whole, or as one of the allowed conformance classes. These conformance classes state the implementation options for the representation of finite element analysis models, controls and results, analysis reports, geometric models, composite material constituents and their representations, materials, and configuration control.

A high level planning information model for this application protocol is shown in Figure 1. At this level, the product can be conceptualized as a part that has both design and analysis product definitions. Each definition has one or more shape representations. The analysis product definition has an associated finite element model, analysis controls, and analysis results in addition to its shape representations.

The three possible product shape representations in this application protocol include the nominal design shape, an idealized analysis shape, and a finite element model node shape. The nominal design shape includes geometry and topology for the part and its constituents, such as ply boundaries. The idealized analysis shape includes only the geometry and topology for mesh generation boundaries and associated node geometry. The node shape includes only the node geometry, with no association to design shape or to analysis idealized shape.

The five types of geometric and topological models that may be used to represent part shape in this application protocol are: wireframe and surface without topology, wireframe geometry with topology, manifold surfaces with topology, faceted boundary representation, and advanced boundary representation.

The finite element analysis model consists of nodes, elements, and the associated element properties. The finite element properties include shape aspects and material properties.

The finite element analysis material properties are specified with respect to an environment. The material response matrices of the material properties may have an associated laminate table. The laminate table specifies the constituents, such as plies in a laminate. Each constituent has a boundary, stock material, and specifications. A separate geometric representation is used for composite material constituent representations.

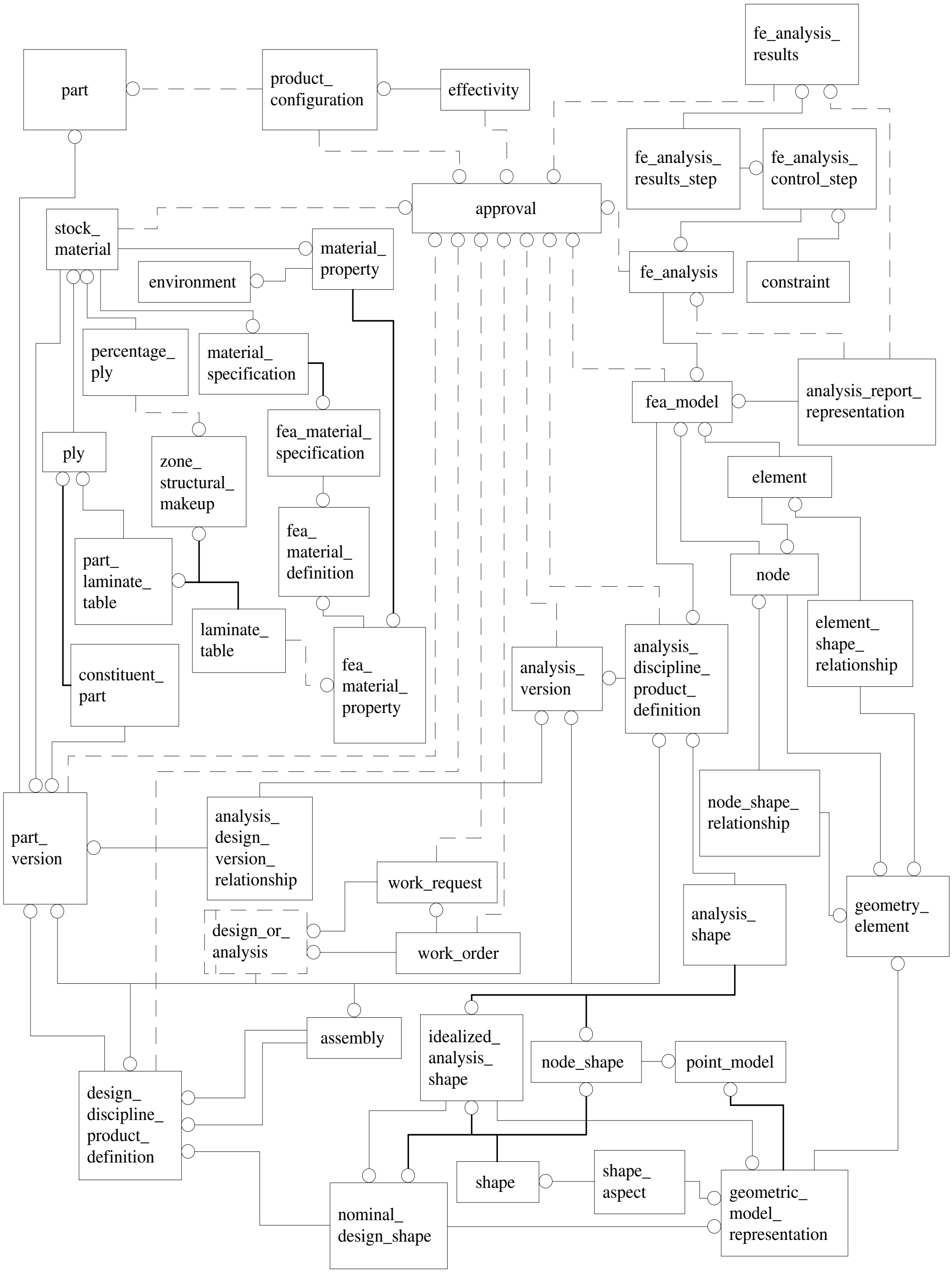

Figure 1 - Data planning model

The finite element analysis controls and results are associated with a finite element model. The analysis controls specify the operations to be carried out upon the model as a series of analysis steps. The analysis results specify the state of the analysis variables at an instant in time. The state information includes nodal solution variables such as deflections, the field variables within the elements, and the values of constraints at a node. An analysis report of the finite analysis results may be presented in tabular and graphical form. The analysis report also documents the detail analyses upon which the finite element analysis results are based.

Industrial automation systems and integration — Product data representation and exchange —
Part 209: Application protocol: Composite and metallic structural analysis and related design

1 Scope

This part of ISO 10303 specifies the use of integrated resources necessary for the scope and information requirements for the analysis and related design of composite and metallic structural parts. This part of ISO 10303 satisfies the need for the exchange of computer-interpretable composite and metallic structural product definitions, including product shape, associated finite element analysis (FEA) models, material properties, and analysis results.

NOTE The application activity model (AAM) 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.

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

— the definition of composite structural parts;

— the definition of metallic structural parts;

— linear static finite element analysis;

— linear modes and frequencies finite element analysis;

— the product definition and configuration control information pertaining to the design through analysis stages of a product's development;

— the information relating the part to the adjoining components in an assembly by either explicit or external reference;

— the 2D and 3D models depicting the product shape;

— the five types of geometric and topologic model representations:

  — wireframe and surface without topology;

  — wireframe geometry with topology;

- manifold surfaces with topology;
- faceted boundary representation;
- advanced boundary representation.

— the representations for design and analysis disciplines and the association of nominal design shape, idealized analysis shape, and finite element node shape representations;

— the association of the constituents of composite and metallic parts with the constituent shape model;

— the depiction of composite laminate tables describing the material, stacking sequence, ply orientation, and constituents of the composite or a portion of the composite with a defined shape;

— the identification of material specifications from internal and external sources and their properties for a specific operating environment;

— the finite element analysis model, analysis controls, and analysis results information;

— the plane stress and simple plane strain types of linear static and linear modes and frequencies finite element structural analyses;

— the graphical presentation of:

- finite element model maps;
- analysis output information displays on top of the finite element model mesh;
- line drawings or images which document the part aspects subjected to detail analysis.

— the tabular presentation of the analysis assumptions, loadings, and critical locations in finite element and detail analysis performed for the assessment of the margin of safety;

— the administrative information necessary to track the approval and configuration control of the design and analysis of a product at a point in the life cycle when approval and configuration control are necessary;

—- the identification of the supplier of a product, design, or analysis and, where required by an organization, the qualification information for the supplier;

— a change to a design and an analysis, including information to identify the change, at a point in the life cycle when tracking a change is necessary;

— the identification, when required, of the contract under which a design is developed and an analysis is performed;

— the identification of the security classification of a part;

— the explicit representation of a bill-of-material.

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

— the business information for the management of a design and analysis project;

— the product definition and configuration control information pertaining to any information other than that necessary for design and analysis;

— alternate representation of the information by disciplines outside of design and analysis such as manufacturing;

— the use of constructive solid geometry for the representation of the shape of the product;

— the other types of finite element analysis beyond linear analysis, such as non-linear static analysis;

— the explicit graphical presentations derivable from design or analysis product representations;

— the composite fabrication process information;

— the product definition of initial or in-process part shapes.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO 10303. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO 10303 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO 31:1992, *Quantities and units*

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

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

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

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

ISO 10303-21:1994, *Industrial automation systems and integration - Product data representation and exchange - Part 21: Clear text encoding of the exchange structure*

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

ISO 10303-41:1994, *Industrial automation systems and integration - Product data representation and exchange - Part 41: Integrated generic resources: Fundamentals of product description and support*

ISO 10303-42:1994, *Industrial automation systems and integration - Product data representation and exchange - Part 42: Integrated generic resources: Geometric and topological representation*

ISO 10303-43:1994, *Industrial automation systems and integration - Product data representation and exchange - Part 43: Integrated generic resources: Representation structures*

ISO 10303-44:1994, *Industrial automation systems and integration - Product data representation and exchange - Part 44: Integrated generic resources: Product structure configuration*

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

ISO 10303-47:1997, *Industrial automation systems and integration - Product data representation and exchange - Part 47: Integrated generic resource: Shape variation tolerances*

ISO 10303-104:2000 *Industrial automation systems and integration - Product data representation and exchange - Part 104: Integrated application resource: Finite element analysis*

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

ISO 10303-502:2000 *Industrial automation systems and integration - Product data representation and exchange - Part 502: Application interpreted construct: Shell-based wireframe*

ISO 10303-507:2001 *Industrial automation systems and integration - Product data representation and exchange - Part 507: Application interpreted construct: Geometrically bounded surface*

ISO 10303-509:2001 *Industrial automation systems and integration - Product data representation and exchange - Part 509: Application interpreted construct: Manifold surface*

ISO 10303-510:2000 *Industrial automation systems and integration - Product data representation and exchange - Part 510: Application interpreted construct: Geometrically bounded wireframe*

ISO 10303-512:1999 *Industrial automation systems and integration - Product data representation and exchange - Part 512: Application interpreted construct: Faceted boundary representation*

ISO 10303-514:1999 *Industrial automation systems and integration - Product data representation and exchange - Part 514: Application interpreted construct: Advanced boundary representation*

3 Terms, definitions and abbreviations

For the purposes of this part of ISO 10303, the following terms, definitions and abbreviations apply.

3.1 Terms defined in ISO 10303-1

For the purposes of this part of ISO 10303, the following terms and definitions given in ISO 10303-1 apply:

— abstract test suite;

— application;

— application activity model;

— application context;

— application interpreted model;

— application object;

— application protocol;

— application reference model;

— application resource;

— assembly;

— component;

— conformance class;

— conformance requirement;

— data;

— data exchange;

— exchange structure;

— generic resource;

— implementation method;

— information;

— information model;

— integrated resource;

— interpretation;

— PICS proforma;

— presentation;

— product;

— product data;

— product_information;

— protocol implementation conformance statement;

— resource construct;

— structure;

— unit of functionality.

3.2 Terms defined in ISO 10303-11

For the purposes of this part of ISO 10303, the following terms and definitions given in ISO 10303-11 apply:

— constant;

— entity;

— entity (data type) instance;

— instance;

— value.

3.3 Terms defined in ISO 10303-21

For the purposes of this part of ISO 10303, the following terms and definitions given in ISO 10303-21 apply:

— clear text encoding;

— keyword;

— section.

3.4 Terms defined in ISO 10303-31

For the purposes of this part of ISO 10303, the following terms and definitions given in ISO 10303-31 apply:

— conformance testing.

3.5 Terms defined in ISO 10303-42

For the purposes of this part of ISO 10303, the following terms and definitions given in ISO 10303-42 apply:

— axi-symmetric;

— bounds;

— boundary;

— boundary representation solid model;

— connected;

— coordinate space;

— curve;

— cycle;

— extent;

— finite;

— graph;

— handle;

— interior;

— list;

— overlap;

— set;

— surface.

3.6 Terms defined in ISO 10303-44

For the purposes of this part of ISO 10303, the following terms and definitions given in ISO 10303-44 apply:

— bill-of-material structure;

— constituent;

— effectivity;

— link;

— lot;

— node.

3.7 Terms defined in ISO 10303-45

For the purposes of this part of ISO 10303, the following terms and definitions given in ISO 10303-45 apply:

— material;

— material property;

— material designation.

3.8 Terms defined in ISO 10303-104

For the purposes of this part of ISO 10303, the following terms and definitions given in ISO 10303-104 apply:

— 2D model;

— 3D model.

3.9 Terms defined in ISO 10303-203

For the purposes of this part of ISO 10303, the following terms and definitions given in ISO 10303-203 apply:

— mechanical part;

— solid model;

— wireframe model.

3.10 Other terms and definitions

For the purposes of this part of ISO 10303, the following terms and definitions apply:

3.10.1
analysis discipline
the organization or activity whose purpose is to use analytical and numerical methods to predict and verify the intended behavior of a part in service.

3.10.2
design discipline
the organization or activity whose purpose is to create a engineering design and representation of a product. This design and its representation are used as the basis of other activities such as analysis or manufacturing.

3.10.3
fill
yarn running at right angle to the warp in a woven fabric.

3.10.4
honeycomb core
thin sheet materials or ribbons formed and bonded into hexagonal honeycomb-like configurations. By varying the sheet material, sheet thickness, cell size, and cell shape, cores of a wide range of density and structural properties can be produced.

3.10.5
idealized
geometry that has been altered to facilitate use for analytical representations.

EXAMPLE a sheet of material whose brep solid representation is idealized to a trimmed surface such that a shell (surface with thickness) finite element can represent the sheet.

3.10.6
radius filler
a constituent of a composite part that is used to fill the gap between constituent parts with rounded corners.

3.10.7
roving
a material with tows in one direction formed into various cross sections.

3.10.8
structural part
a part that has as its principle design requirement the support of a load. The role of this part in a product is defining the shape and protecting the product.

3.10.9
tow
a continuous group of fibers collected into a loose strand or assemblage without any substantial twist. It may be impregnated with a resin.

3.10.10
warp
the lenthwise direction of the weave in cloth or roving.

3.10.11
yarn
a twisted group of fibers that may be woven into a fabric.

3.11 Abbreviations

For the purposes of this part of ISO 10303, the following abbreviations apply:

AAM	application activity model
AIC	application interpreted construct
AIM	application interpreted model
AP	application protocol
ARM	application reference model
BOM	bill-of-material
CAD	computer aided design
CAE	computer aided engineering
CSP	composite structural part
FE	finite element
FEM	finite element model

FEA	finite element analysis
ICAM	integrated computer-aided manufacturing
ID	identification
IDEF0	ICAM definition language 0
IDEF1X	ICAM definition language 1 - extended
PICS	protocol information and conformance statement
SP	structural part
UoF	Unit of Functionality

4 Information requirements

This clause specifies the information required for the exchange of computer-interpretable composite and metallic structural product definition shape, the associated finite element analysis (FEA) model and analysis results, and material 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 activities identified as being in the scope of this application protocol in annex F.

NOTE 3 The mapping table specified in 5.1 shows how the information requirements are met using the integrated resources of this International Standard. The use of the integrated resources introduces additional requirements that are common to other application protocols.

4.1 Units of functionality

The subclause specifies the units of functionality for the composite and metallic structural analysis and related design application protocol. The part of ISO 10303 specifies the following units of functionality:

— activity_control;

— advanced_boundary_representation;

— analysis_report;

— assembly;

— authorization;

— composite_constituent_representation;

— effectivity;

— end_item_identification;

— faceted_boundary_representation;

— fea_model;

— fe_analysis_control;

— fe_analysis_results;

— manifold_surface_with_topology;

— material;

— non_topological_surface_and_wireframe;

— part_composite_constituents;

— part_identification;

— part_laminate_table;

— part_shape;

— wireframe_with_topology;

— zone_composite_constituents_and_their_representation.

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

4.1.1 activity_control

The activity_control UoF describes the history of part and analysis versions. It identifies initial part and analysis requirements as well as the change requirements for revised part and analysis versions. It provides the directive for work to proceed in the development of these initial or modified part and analysis versions.

The following application objects are used by the activity_control UoF:

— change_order;

— change_request;

— date;

— start_order;

— start_request;

— work_order;

— work_request.

4.1.2 advanced_boundary_representation

The advanced_boundary_representation UoF contains the representation of the part by shapes using advanced boundary representation solid models. The representation allows for the definition of curves and surfaces and the topology that bounds them. Boundaries are explicitly defined only by topology. All of the geometry that defines the part shapes shall be associated with topology.

The following application objects are used by the advanced_boundary_representation UoF:

— advanced_boundary_representation;

— geometric_model_representation;

— geometry_element.

4.1.3 analysis_report

The analysis_report UoF contains the objects necessary to represent the text and graphical information documenting finite element and associated detail analyses.

The following application objects are used by the analysis_report UoF:

— analysis_report_representation;

— graphical_representation;

— tabular_representation.

4.1.4 assembly

The assembly UoF represents either directly or by external reference, the components assembled into the part being analyzed.

The following application objects are used by the assembly UoF:

— assembly;

— component_assembly_position;

— make_from;

— next_higher_assembly;

— promissory_usage;

— substitute_part;

— supplied_part_version;

— supplier.

4.1.5 authorization

The authorization UoF contains information that indicates a part or an analysis has been reviewed for information content and correctness and is found to be acceptable. The acceptance is on a specific date, at a particular status, by people having the roles that allow them to grant the authorization.

The following application objects are used by the authorization UoF:

— approval;

— date;

— person_organization.

4.1.6 composite_constituent_representation

The composite_constituent_representation UoF contains the geometric constructs that compose the shape representation of plies. The information consists of the mathematical definition of all two-dimensional and three-dimensional geometric elements that compose a ply_shape. The elements may be points, curves, surfaces, vertices, edges, loops, faces, or shells.

The following application objects are used by the composite_constituent_representation UoF:

— angle_measure;

— beveled_sheet_representation;

— boundary_curve_representation;

— composite_sheet_representation;

— constituent_shape_representation;

— curve;

— direction;

— face_based_sheet_representation;

— filament_laminate_shape;

— flat_pattern_ply_shape;

— geometric_model_representation;

— geometric_sheet_representation;

— geometry_element;

— laid_ply_shape;

— length_measure;

— location;

— loop;

— ply_shape;

— projected_ply_shape;

— surface;

— surface_ply_shape;

— three_d_geometry_set;

— view_ply_shape.

4.1.7 effectivity

The effectivity UoF represents the planned usage of components in a product model.

The following application objects are used by the effectivity UoF:

— date;

— date_effectivity;

— effectivity;

— lot_effectivity;

— retention_period:

— sequence_effectivity.

4.1.8 end_item_identification

The end_item_identification UoF identifies the different products that are sold by an organization to its customers. It provides the information structures for an organization to control the configuration of these products.

The following application objects are used by the end_item_identification UoF:

— product_configuration;

— product_model.

4.1.9 faceted_boundary_representation

The faceted_boundary_representation UoF contains the representation of the shape of a part where planar surfaces are the bounding surfaces for the solid model. Only points and planar polygons are used in this representation. Much of the topological information is implicit for this representation.

The following application objects are used by the faceted_boundary_representation UoF:

— faceted_boundary_representation;

— geometric_model_representation;

— geometry_element.

4.1.10 fe_analysis_control

The fe_analysis_control UoF describes the analysis operations carried out upon the finite element model. A control specifies an initial state of the model, analysis operations to be carried out upon it and constraints upon the model for each of the analysis steps in sequence. The initial state of a step describes the value of analysis variables of a model at an instance including the nodal solution variables (such as deflections), the field variable values within the elements (such as stresses), and the values of the constraints.

The following application objects are used by the fe_analysis_control UoF:

— analysis_message;

— calculated_state;

— constraint;

— element_field_variable_definition;

— element_nodal_freedom_actions;

— fe_analysis;

— fe_analysis_control_step;

— fe_analysis_state;

— fe_analysis_state_definition;

— linear_constraint_equation;

— linear_constraint_equation_value;

— linearly_superimposed_state;

— modes_and_frequencies_control_step;

— nodal_degree_of_freedom_reduction;

— nodal_freedom_definitions;

— output_request_state;

— single_point_constraint;

— single_point_constraint_values;

— specified_state;

— static_control_step.

4.1.11 fe_analysis_results

The fe_analysis_results UoF describes the resulting state of a finite element model to an analysis control as calculated by a finite element analysis application. For each analysis step there is a single resulting state that describes the model at the end of the step. The resulting state describes the value of the analysis variables of a model at an instant including the nodal solution variables (such as deflections), the field variable values within the elements (such as stresses), and the values of the constraints.

The following application objects are used by the fe_analysis_results UoF:

— analysis_message;

— calculated_state;

— constraint;

— element_field_variable_definition;

— element_nodal_freedom_actions;

— fe_analysis_results;

— fe_analysis_results_step;

— fe_analysis_state;

— fe_analysis_state_definition;

— linear_constraint_equation;

— linear_constraint_equation_value;

— linearly_superimposed_state;

— modes_and_frequencies_results_step;

— nodal_degree_of_freedom_reduction;

— nodal_freedom_definitions;

— output_request_state;

— single_point_constraint;

— single_point_constraint_values;

— specified_state;

— static_results_step.

4.1.12 fea_model

The fea_model UoF contains the objects that are associated with the finite element analysis of a part. The information includes generalized nodes, elements, material response matrices, properties, and groups that are combined to form a discrete mesh model of the part.

The following application objects are used by the fea_model UoF:

— curve_cross_section;

— curve_element;

— curve_property;

— curve_section_properties;

— damping_matrix;

— directionally_explicit_element;

— element;

— element_description;

— element_property_geometric_relationship;

— element_shape_aspect;

— element_shape_relationship;

— environment;

— explicit_element;

— fea_material_definition;

— fea_material_property;

— fea_material_specification;

— fea_model;

— fea_model_description;

— geometric_model_representation;

— geometry_element;

— group;

— group_relationship;

— mass_matrix;

— material_property;

— material_specification;

— matrix;

— nodal_results_coordinate_system;

— node;

— node_description;

— node_shape_relationship;

— point_element;

— point_model;

— specification;

— stiffness_matrix;

— substructure_element;

— substructure_node_relationship;

— surface_element;

— surface_property;

— surface_thickness;

— volume_element.

4.1.13 manifold_surface_with_topology

The manifold_surface_with_topology UoF contains the representation of the shape of a part using manifold surfaces with topology. The outer boundary of the part is defined by 3D curves, surfaces, and topological faces.

The following application objects are used by the manifold_surface_with_topology UoF:

— geometric_model_representation;

— geometry_element;

— manifold_surface_with_topology.

4.1.14 material

The material UoF describes the specifications for a material, and the form in which the material is typically procured.

The following application objects are used by the material UoF:

— additional_design_information;

— anisotropic_material;

— design_specification;

— direction;

— discontinuous_fiber_assembly;

— environment;

— filament_assembly;

— isotropic_material;

— material_direction;

— material_property;

— material_specification;

— measure_value;

— process_specification;

— specification;

— stock_core;

— stock_material;

— surface_finish_specification;

— usage_constraint.

4.1.15 non_topological_surface_and_wireframe

The non_topological_surface_and_wireframe UoF contains the representation of the shape of a part using surface or wireframe geometry without topology. These representations are formed only by the use of points, curves, and surfaces. The boundaries of the curves are defined explicitly by points on the curves and by explicit associations between the points and the curves they bound. The boundaries of the surfaces are defined by curves on the surfaces and by explicit associations between the curves and the surfaces they bound. Surfaces and curves shall be explicitly trimmed unless they are closed.

The following application objects are used by the non_topological_surface_and_wireframe UoF:

— geometric_model_representation;

— geometry_element;

— non_topological_surface_and_wireframe.

4.1.16 part_composite_constituents

The part_composite_constituents UoF represents the objects that make up a composite part.

The following application objects are used by the part_composite_constituents UoF:

— composite_assembly;

— constituent_part;

— curve;

— direction;

— filament_laminate;

— length_measure;

— mass_measure;

— ply;

— ply_laminate;

— ply_orientation_angle;

— ply_piece;

— point;

— point_and_vector;

— point_path;

— processed_core.

4.1.17 part_identification

The part_identification UoF is the structure through which a part, unique versions of a part, and different discipline views of a part can be defined.

The following application objects are used by the part_identification UoF:

— alternate_part;

— analysis;

— analysis_design_version_relationship;

— analysis_discipline_product_definition;

— analysis_version;

— date;

— design_discipline_product_definition;

— design_material;

— file;

— mass_measure;

— part;

— part_version.

4.1.18 part_laminate_table

The part_laminate_table UoF describes the order that the physical constituents of a laminate are assembled. The description is for an entire part.

The following application objects are used by the part_laminate_table UoF:

— composite_assembly_sequence_definition;

— composite_assembly_table;

— direction;

— laminate_table;

— location;

— part_laminate_table;

— ply_laminate_sequence_definition;

— ply_laminate_table;

— reinforcement_orientation_basis;

— surface;

— surface_with_direction.

4.1.19 part_shape

The part_shape UoF contains the shape and shape representation objects for a part.

The following application objects are used by the part_shape UoF:

— analysis_shape;

— idealized_analysis_shape;

— node_shape;

— nominal_design_shape;

— shape;

— shape_aspect.

4.1.20 wireframe_with_topology

The wireframe_with_topology UoF contains the representation of the shape of a part by wireframes that are defined by edge topology objects of vertices and edges. This includes 3D curves and topology.

The following application objects are used by the wireframe_with_topology UoF:

— geometric_model_representation;

— geometry_element;

— wireframe_with_topology.

4.1.21 zone_composite_constituents_and_their_representation

The zone_composite_constituents_and_their_representation UoF describes the constituents of a laminate for an aspect of a part.

The following application objects are used by the zone_composite_constituents_and_their_representation UoF:

— angle_measure;

— boundary_representation;

— curve;

— direction;

— draped_orientation_angle;

— edge_zone_shape;

— laid_orientation_angle;

— laminate_table;

— length_measure;

— location;

— loop;

— percentage;

— percentage_laminate_table;

— percentage_ply;

— ply_orientation_angle;

— point;

— point_and_vector;

— point_path;

— point_zone_shape;

— reinforcement_orientation_basis;

— smeared_material;

— surface;

— surface_with_direction;

— thickness_laminate_table;

— zone_structural_makeup;

— zone_structural_makeup_shape_representation.

4.2 Application objects

The subclause specifies the application objects for the Composite and Metallic Structural Analysis and Related Design 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 attributes are defined below.

4.2.1 Additional_design_information

An Additional_design_information is a collection of specifications that are associated with the design of a part. The data associated with an Additional_design_information are the following:

— additional_information;

— design.

4.2.1.1 additional_information

The additional_information specifies the set of Specification (see 4.2.146) objects that make up the Additional_design_information. See 4.3.2 for the application assertion.

4.2.1.2 design

The design specifies the Design_discipline_product_definition (see 4.2.36) to which this Additional_-design_information applies. See 4.3.1 for the application assertion.

4.2.2 Advanced_boundary_representation

An Advanced_boundary_representation is a type of Geometric_model_representation (see 4.2.72) that represents the shape of a part by an advanced boundary representation solid model. The representation allows for the definition of curves and surfaces and the topology that bounds them. All of the geometry that defines the part shapes shall be associated with topology.

4.2.3 Alternate_part

An Alternate_part is a part that is interchangeable with another part with respect to form, fit and function. The data associated with an Alternate_part are the following:

— alternate.

4.2.3.1 alternate

The alternate specifies the Part (see 4.2.112) that the Alternate_part can be used for. See 4.3.3 for the application assertion.

NOTE The usages of an alternate part are not of interest to an organization that owns it and therefore are not tracked by the organization.

EXAMPLE A design requires a sheet metal screw of a certain size. Sheet metal screws are made by various organizations and are equivalent with respect to form, fit, and function for that size. The various screws have different part numbers assigned to them by the manufacturer. The design organization does not track which screw is used in the actual assembly. The screws are alternate parts.

4.2.4 Analysis

An Analysis is an item that is produced as a result of an engineering analysis process. The data associated with an Analysis are the following:

— analysis_type;

— owner.

4.2.4.1 analysis_type

The analysis_type specifies the type of analysis, such as linear static structural analysis.

4.2.4.2 owner

The owner specifies the Person_organization (see 4.2.118) that owns the Analysis. See 4.3.4 for the application assertion.

4.2.5 Analysis_design_version_relationship

An Analysis_design_version_relationship is an association between an Analysis_version (see 4.2.10) and a Part_version (see 4.2.114). The Analysis_version is with respect to the Part_version. The data associated with an Analysis_design_version_relationship are the following:

— analysis;

— design.

4.2.5.1 analysis

The analysis specifies the Analysis_version in the relationship. See 4.3.5 for the application assertion.

4.2.5.2 design

The design specifies the Part_version in the relationship. See 4.3.6 for the application assertion.

4.2.6 Analysis_discipline_product_definition

An Analysis_discipline_product_definition is a product definition or view of a Analysis_version (see 4.2.10) from the perspective of the analysis organization. The Analysis_discipline_product_definition is controlled by the analysis organization.

NOTE The entity may be used to capture the definition of a particular Analysis_version (see 4.2.10) at any intermediate stage of its development where the definition is not formally released by an organization at the Analysis_version level. It may be used to capture the various stages in the definition cycle of a product. In the case of composite parts the views to capture include a finite element model view, a material properties' view of particular features of a part, and a constituent view of particular features of a part.

The data associated with an Analysis_discipline_product_definition are the following:

— approved_by;

— cae_filename;

— creation_date;

— creator;

— description;

— discipline_identification;

— version.

4.2.6.1 approved_by

The approved_by specifies the approval of the Analysis_discipline_product_definition. The approved_by need not be specified for a particular Analysis_discipline_product_definition. See 4.3.8 for the application assertion.

4.2.6.2 cae_filename

The cae_filename references the file that contains the information describing the Analysis_discipline_-product_definition. The cae_filename need not be specified for a particular Analysis_discipline_product definition.

4.2.6.3 creation_date

The creation_date specifies the date that the Analysis_discipline_product_definition was defined. See 4.3.9 for the application assertion.

4.2.6.4 creator

The creator specifies the Person_organization (see 4.2.118) that created the Analysis_discipline_-product_definition. See 4.3.10, 4.3.11 for the application assertion.

4.2.6.5 description

The description specifies the purpose for a particular definition of the product.

EXAMPLE A laminate_table is a type of Analysis_discipline_product_definition where the description might read: "smeared material properties for a particular point on the laminate" or "percentages of different materials within a particular area on the laminate."

4.2.6.6 discipline_identification

The discipline_identification specifies the identification of the functional unit or group within the organization that the definition of the product pertains.

4.2.6.7 version

The version specifies the iteration identified by the Analysis_version. See 4.3.7 for the application assertion.

4.2.7 Analysis_message

An Analysis_message is a type of Fe_analysis_state_definition (see 4.2.61) that is an information message such as a note, warning, or error message. The Analysis_message may be attached to an element, a group of elements, a node, or an entire model. The data associated with an Analysis_message are the following:

— message_level;

— quality.

4.2.7.1 message_level

The message_level specifies the severity of the Analysis_message.

4.2.7.2 quality

The quality specifies the text of the Analysis_message.

4.2.8 Analysis_report_representation

An Analysis_report_representation is the text and graphical information documenting the finite element and associated detail analyses. An Analysis_report_representation may be one or both of the following: Graphical_representation (see 4.2.75), Tabular_representation (see 4.2.167). The data associated with an Analysis_report_representation are the following:

— control;

— model_ref;

— result.

4.2.8.1 control

The control specifies the Fe_analysis (see 4.2.56) being documented. The control need not be specified for a particular Analysis_report_representation. See 4.3.12 for the application assertion.

4.2.8.2 model_ref

The model_ref specifies the Fea_model (see 4.2.65) being documented. See 4.3.14 for the application assertion.

4.2.8.3 result

The result specifies the Fe_analysis_results (see 4.2.58) being documented. The result need not be specified for a particular Analysis_report_representation. See 4.3.13 for the application assertion.

4.2.9 Analysis_shape

An Analysis_shape is a shape for a Part_version (see 4.2.114) as defined by the analysis discipline. An Analysis_shape may be one or both of the following: Idealized_analysis_shape (see 4.2.78), Node_shape (see 4.2.107). The data associated with an Analysis_shape are the following:

— analysis_view.

4.2.9.1 analysis_view

The analysis_view specifies the Analysis_discipline_product_definition (see 4.2.6) whose shape is represented by the Analysis_shape. See 4.3.15 for the application assertion.

4.2.10 Analysis_version

An Analysis_version is the identification of the analysis for a part before, during, or after its analysis has undergone a formal release or change. Only changes that are formally tracked by the organization that is responsible for the analysis of the part shall be identified by an Analysis_version. Iterations on analysis that are not formally tracked by the organization responsible for the definition of the part shall not be identified by an Analysis_version but should be tracked by an Analysis_discipline_product_definition (see 4.2.6). The data associated with an Analysis_version are the following:

— analysis_number;

— approved_by;

— contract_number;

— creator;

— release_status;

— revision_letter;

— security_code.

4.2.10.1 analysis_number

The analysis_number specifies the unique identification of an analysis for a particular organization. See 4.3.16 for the application assertion.

4.2.10.2 approved_by

The approved_by specifies the approval of the Analysis_version. The approved_by need not be specified for a particular Analysis_version. See 4.3.17 for the application assertion.

4.2.10.3 contract_number

The contract_number identifies the business contract under which the part is analyzed. An analysis may but need not be performed under a contract. If the analysis is performed under a contract, the attribute shall specify the unique identification of that contract.

4.2.10.4 creator

The creator specifies the Person_organization (see 4.2.118) that created the Analysis_version. See 4.3.18 for the application assertion.

4.2.10.5 release_status

The release_status specifies the status of the version of an analysis with respect to the dissemination of analysis information. It shall always be one of two values: released or unreleased. The versions that are released have been reviewed and approved for further use. The versions that are unreleased have not been reviewed and subsequently approved for further use. Since the analysis_version application object is also being used to manage pre-release engineering changes, the release_status attribute should always have the value of "unreleased" during the initial design or analysis phase until engineering officially releases the product information.

4.2.10.6 revision_letter

The revision_letter specifies the unique identification of a particular version of an analysis.

4.2.10.7 security_code

The security_code specifies the security classification of a particular version of an analysis.

4.2.11 Angle_measure

An Angle_measure is the amount of rotation about an axis necessary to bring one line or plane into coincidence with another.

4.2.12 Anisotropic_material

An Anisotropic_material is a type of Stock_material (see 4.2.154) whose properties are different in each material direction.

4.2.13 Approval

An Approval is the indication within an organization of concurrence or nonconcurrence of product data. The data associated with an Approval are the following:

— authorized_by;

— effective_date;

— purpose;

— status.

4.2.13.1 authorized_by

The authorized_by identifies the set of Person_organization (see 4.2.118) objects authorizing the Approval. See 4.3.20 for the application assertion.

4.2.13.2 effective_date

The effective_date specifies when the Approval became or shall become effective. See 4.3.19 for the application assertion.

4.2.13.3 purpose

The purpose specifies the reason for the Approval.

4.2.13.4 status

The status specifies the state of consent applied to product data or a relationship between product data. Valid values for the status of an Approval are: approved, not_yet_approved, disapproved, and withdrawn.

Approved means the necessary authorizations have been established. Not_yet_approved means the necessary authorizations are pending. Disapproved means the necessary authorizations have been denied. Withdrawn means the necessary authorizations have been revoked.

4.2.14 Assembly

An Assembly is the parent-child relationship between an assemblage of parts and a component or subassembly used in the assemblage. An Assembly shall be one of the following: Next_higher_assembly (see 4.2.101), Promissory_usage (see 4.2.137).

NOTE In composite laminates, industry practice may vary. If constituent_parts are to be configuration controlled, then they would appear in an assembly relationship and in a laminate table. If these constituent_parts are not managed in the engineering bill-of-materials, then they would not have an assembly relationship, but would appear in a laminate table.

The data associated with an Assembly are the following:

— assembly_part;

— component_part;

— security_code.

4.2.14.1 assembly_part

The assembly_part specifies the parent product definition in the assembly relationship. The assembly_-part may itself be part of a larger assembly. See 4.3.21 for the application assertion.

4.2.14.2 component_part

The component_part specifies the child product definition in the assembly relationship. The component_-part is an autonomous item that cannot be subdivided further. See 4.3.21 for the application assertion.

4.2.14.3 security_code

The security_code specifies the security classification of the component_part within the context of its use in the Assembly.

NOTE The differs from the attribute security_code associated with a Part_version (see 4.2.114) in that some aspect of the component as it relates to an assembly changes its security classification when that part exists as a component within the assembly to which the security_code is related. The Part_version may have a security_code when it exists alone, that is outside of its use in any particular assembly. The security_code specified here is defined in a very specific context.

4.2.15 Beveled_sheet_representation

A Beveled_sheet_representation is a type of Constituent_shape_representation (see 4.2.26) that has a significant thickness, and whose edge is beveled at a constant angle around its entire perimeter. It is an extension of the concept of a thin sheet to products, such as honeycomb core, that have significant thickness but can be formed to mate with a curved surface.

The data associated with a Beveled_sheet_representation are the following:

— bevel_angle;

— boundary_surface;

— vertical_profile_height.

4.2.15.1 bevel_angle

The bevel_angle specifies the angle from the plane normal to the defining surface of part's profile to the beveled surface. See 4.3.22 for the application assertion.

4.2.15.2 boundary_surface

The boundary_surface specifies the base exterior boundary and defining surface of the part. See 4.3.23 for the application assertion.

4.2.15.3 vertical_profile_height

The vertical_profile_height specifies the vertical height of the part's profile from the base boundary to where the bevel begins. See 4.3.24 for the application assertion.

4.2.16 Boundary_curve_representation

A Boundary_curve_representation is the set of curves that define the sheet edge. These curves define a closed area but are not necessarily restrained topologically. The curves that encompass the other curves are the outer boundary, while curves that define a closed area within the outer boundary are cutouts. The data associated with a Boundary_curve_representation are the following:

— bounds.

4.2.16.1 bounds

The bounds specifies a set of Curve (see 4.2.28) objects that defines the boundary aspect. See 4.3.25 for the application assertion.

4.2.17 Calculated_state

A Calculated_state is a type of Fe_analysis_state (see 4.2.60) representing information that has been calculated by an analysis step.

4.2.18 Change_order

A Change_order is a type of Work_order (see 4.2.174) that authorizes the development of a modified design or analysis of a part. A Change_order results in the establishment of a new product definition, Part_version, or Analysis_version. The data associated with a Change_order are the following:

— adopted_solution;

— change_date.

4.2.18.1 adopted_solution

The adopted_solution specifies the solution selected from the set of recommended solutions.

4.2.18.2 change_date

The change_date specifies the date when the Change_order became effective. See 4.3.26 for the application assertion.

4.2.19 Change_request

A Change_request is a type of Work_request (see 4.2.175) that identifies the proposed extent of work to be done to modify the design or analysis of a part. The data associated with a Change_request are the following:

— consequence;

— recommended_solution;

— version.

4.2.19.1 consequence

The consequence specifies the effects in performance, functionality, or form of a particular recommended solution on the affected part or analysis version.

4.2.19.2 recommended_solution

The recommended_solution specifies a potential resolution that would satisfy the requirements described in the Change_request through the description attribute of Work_request.

4.2.19.3 version

The version specifies a unique identifier for each iteration of the Change_request.

4.2.20 Component_assembly_position

A Component_assembly_position is the identification of the positioning of a component in a Next_-higher_assembly (see 4.2.20). The data associated with a Component_assembly_position are the following:

— assembly_shape;

— component_shape;

— definition;

— transformation.

4.2.20.1 assembly_shape

The assembly_shape specifies the form of the assembly in which the component is positioned. See 4.3.27 for the application assertion.

4.2.20.2 component_shape

The component_shape specifies the form of the component in the assembly. See 4.3.27 for the application assertion.

4.2.20.3 definition

The definition specifies the assembly in which the component is positioned. See 4.3.28 for the application assertion.

4.2.20.4 transformation

The transformation specifies the translation and orientation of the component in the geometric space of the assembly.

4.2.21 Composite_assembly

A Composite_assembly is a type of Constituent_part (see 4.2.25) that is made of multiple materials that are bonded together. Composite_assembly may be made from any combination of materials. A completed Composite_assembly need not be a rigid shape. Composite_assembly may also be used as a composite part. The data associated with a Composite_assembly are the following:

— layup_part;

— shape.

4.2.21.1 layup_part

The layup_part identifies constituent parts that make up a composite_assembly. See 4.3.30 for the application assertion.

4.2.21.2 shape

The shape specifies the geometric model used to represent the shape of the Composite_assembly. This is defined in terms of the part's final shape. The shape shall be one of the following: Advanced_-boundary_representation (see 4.2.2), Faceted_boundary_representation (see 4.2.55), Manifold_surface_-with_topology (see 4.2.91), Non_topological_surface_and_wireframe (see 4.2.110), Three_d_geometry_-set (see 4.2.169), Wireframe_with_topology (see 4.2.173). The shape need not be specified for a particular Composite_assembly. See 4.3.29, 4.3.31, 4.3.32, 4.3.33, 4.3.34, and 4.3.35 for the application assertions.

4.2.22 Composite_assembly_sequence_definition

A Composite_assembly_sequence_definition is a single unique sequence or layer within a Composite_-assembly (see 4.2.21). The sequence or layer contains one or more constituent parts. When more than one constituent part resides in the same sequence or layer, the constituents must not overlap.

NOTE If constituents do overlap they must be put on separate sequence or layers.

The data associated with a Composite_assembly_sequence_definition are the following:

— components_in_sequence;

— properties.

4.2.22.1 components_in_sequence

The components_in_sequence specifies the collection of components in a single unique sequence or layer in a part. The components_in_sequence shall be a collection of one or more of the following: Composite_assembly (see 4.2.21), Filament_laminate (see 4.2.68), Ply (see 4.2.119), Ply_laminate (see 4.2.120), Processed_core (see 4.2.133). See 4.3.36, 4.3.38, 4.3.39, 4.3.40, and 4.3.41 for the application assertions.

4.2.22.2 properties

The properties specifies the Fea_material_property (see 4.2.63) that defines the physical responses of the materials. The properties need not be specified for a particular Composite_assembly_sequence_-definition. See 4.3.36, 4.3.38, 4.3.40 and 4.3.41 for the application assertions.

4.2.23 Composite_assembly_table

A Composite_assembly_table is a type Part_laminate_table (see 4.2.113) that provides an ordered list of constituent parts. The ordered list provides basic placement position within the laminate (see Figure 2).

NOTE 1 The valid components of a Composite_assembly_table are: Ply (see 4.2.119), Ply_laminate (see 4.2.120), Filament_laminate (see 4.2.68), Processed_core (see 4.2.133), and Composite_assembly (see 4.2.21).

NOTE 2 A Composite_assembly_table is only valid to describe a Composite_assembly.

The data associated with a Composite_assembly_table are the following:

— sequence_groups.

4.2.23.1 sequence_groups

The sequence_groups specifies the groups of constituents to be laid in sequence. See 4.3.42 for the application assertion.

4.2.24 Composite_sheet_representation

A Composite_sheet_representation is a type of Constituent_shape_representation (see 4.2.26) that defines a sheet product. A sheet product is a product whose thickness is constant and small in comparison to its length and width. Most sheet products will conform to the surface on which they are placed. The shape of a sheet product can be represented in a 3-dimensional model of the actual shape, as a shape projected onto the layup surface, or as a shape projected into a 2-dimensional plane.

EXAMPLE A typical composite sheet is 0.1 to 0.4 millimeters thick while its length and width are much greater than 25 millimeters.
A Composite_sheet_representation shall be one of the following: Face_based_sheet_representation (see 4.2.54), Geometric_sheet_representation (see 4.2.73).

4.2.25 Constituent_part

A Constituent_part is a composite item that makes up a composite part. Constituent_part may, but need not be, defined in a component-assembly relationship. A Constituent_part shall be one of the following: Composite_assembly (see 4.2.21), Filament_laminate (see 4.2.68), Ply (see 4.2.119), Ply_piece (see 4.2.124), Ply_laminate (see 4.2.120), Processed_core (see 4.2.133). The data associated with a Constituent_part are the following:

— constituent_part_identification;

— of_part;

— weight.

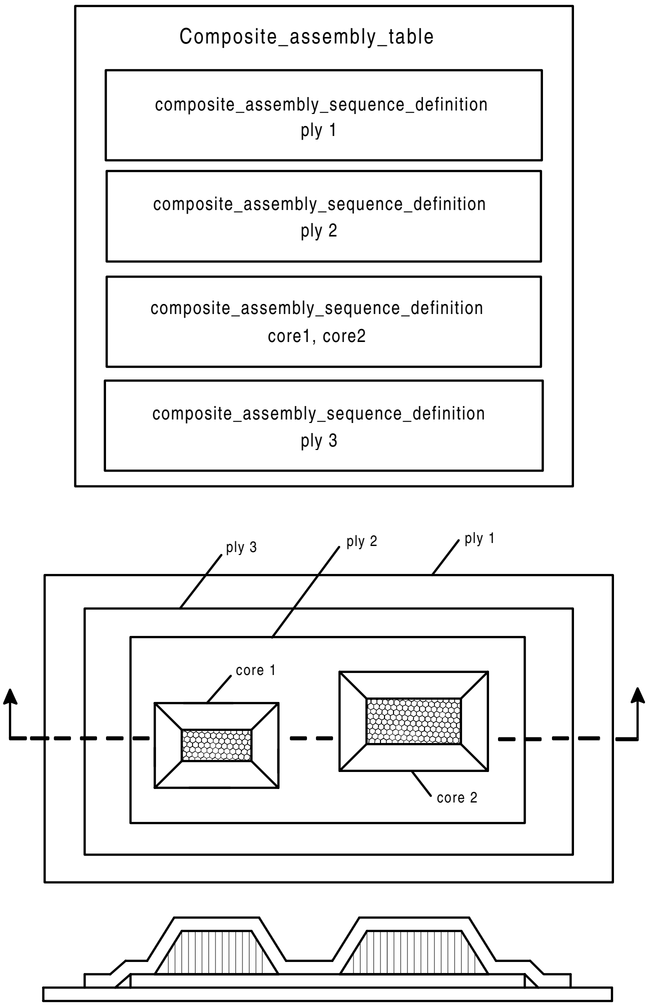

Figure 2 - Composite_assembly_table

4.2.25.1 constituent_part_identification

The constituent_part_identification specifies the composite item.

4.2.25.2 of_part

The of_part specifies the Part_version (see 4.2.114) for the constituent part. See 4.3.44 for the application assertion.

4.2.25.3 weight

The weight quantifies the mass of the Constituent_part. The weight need not be specified for a particular Constituent_part. See 4.3.43 for the application assertion.

4.2.26 Constituent_shape_representation

A Constituent_shape_representation is a type of Geometric_model_representation (see 4.2.72) that defines the shape of composite part constituents. A Constituent_shape_representation shall be one of the following: Beveled_sheet_representation (see 4.2.15), Composite_sheet_representation (see 4.2.24), Three_d_geometry_set (see 4.2.169).

4.2.27 Constraint

A Constraint is a specified value of a degree of freedom at a node or a relationship between freedoms at many nodes for an Fe_analysis_control_step. A Constraint shall be one of the following: Linear_-constraint_equation (see 4.2.84), Nodal_degree_of_freedom_reduction (see 4.2.102), Single_point_-constraint (see 4.2.143). The data associated with a Constraint are the following:

— identification;

— steps.

4.2.27.1 identification

The identification specifies the Identifier (see 4.2.27) for the Constraint.

4.2.27.2 steps

The steps specifies the set of Fea_analysis_control_step (see 4.2.57) objects in which the Constraint is applied. See 4.3.45 for the application assertion.

4.2.28 Curve

A Curve can be envisioned as the path of a point moving in its coordinate space.

4.2.29 Curve_cross_section

A Curve_cross_section is the cross section of a Curve_element (see 4.2.30). The data associated with a Curve_cross_section are the following:

— property;

— section.

4.2.29.1 property

The property specifies the Curve_property (see 4.2.31) for which a cross section is being defined. See 4.3.46 for the application assertion.

4.2.29.2 section

The section specifies the cross-sectional shape of a Curve_element. The section shall be one of the following: Idealized_analysis_shape (see 4.2.78), Nominal_design_shape (see 4.2.109). The section need not be specified for a particular Curve_cross_section. See 4.3.47 and 4.3.48 for the application assertions.

4.2.30 Curve_element

A Curve_element is a type of Element (see 4.2.45) that is an idealization of a portion of a part by a curve segment with a specified cross section.

4.2.31 Curve_property

A Curve_property is the association of material and geometric properties to a set of Curve_element (see 4.2.30) objects. The data associated with a Curve_property are the following:

— defined_elements.

4.2.31.1 defined_elements

The defined_elements specifies the set of Curve_element objects for which properties are being defined. See 4.3.49 for the application assertion.

4.2.32 Curve_section_properties

A Curve_section_properties is the cross-sectional property information for a curve element. Properties represented by a Curve_section_property include:

— cross-sectional area;

— shear area;

— second moment of area;

— torsional constant;

— location of centroid;

— location of shear center;

— location of nonstructural mass;

— polar moment of inertia;

— value of nonstructural mass;

— warping constant.

The data associated with a Curve_section_properties are the following:

— property.

4.2.32.1 property

The property specifies the Curve_property (see 4.2.31) for which a cross section is being defined. See 4.3.50 for the application assertion.

4.2.33 Damping_matrix

A Damping_matrix is a type of Matrix (see 4.2.97). It is an explicit representation of damping at a node, between two nodes in a direction, or for many nodes.

4.2.34 Date

A Date is the identification of a moment in time.

4.2.35 Date_effectivity

A Date_effectivity is a type of Effectivity (see 4.2.44). It is the specification by the design organization of the expected usage of a part in a product configuration. The usage of the part within the product configuration is determined by one or two associated dates. The data associated with a Date_effectivity are the following:

— end_date;

— start_date.

4.2.35.1 end_date

The end_date specifies the last date on which the part is planned to be used or was used in the product_-configuration. The end_date need not be specified for a particular Date_effectivity. See 4.3.51 for the application assertion.

4.2.35.2 start_date

The start_date specifies the first date on which the part is to be used or was used in the product_-configuration. No end_date at the time of exchange signifies that an end_date has not yet been identified. See 4.3.51 for the application assertion.

4.2.36 Design_discipline_product_definition

A Design_discipline_product_definition is one of the design organizational definitions or views of a Part_version (see 4.2.114). A Design_discipline_product_definition is controlled by the design organization.

NOTE The entity may be used to capture the definition of a particular Part_version at any intermediate stage of its development where the definition is not formally released by an organization at the Part_version level. It may be used to capture the various stages in the definition cycle of a product. In the case of composite parts the views to capture include a view of an item's shape, a view of the order constituents are assembled, and a view of the physical constituents that make up a particular feature of the part.

The data associated with a Design_discipline_product_definition are the following:

— approved_by;

— cad_filename;

— creation_date;

— creator;

— description;

— discipline_identification;

— version.

4.2.36.1 approved_by

The approved_by specifies approval of the Design_discipline_product_definition. The approved_by need not be specified for a particular Design_discipline_Product_definition. See 4.3.52 for the application assertion.

4.2.36.2 cad_filename

The cad_filename references the file that contains the geometric description in a computer aided design (CAD) system. The cad_filename need not be specified for a particular Design_discipline_Product_-definition. If it is provided, the reference is interpreted to be to an external definition of the shape of the part that is meaningful only within the organization that designed the part. See 4.3.54 for the application assertion.

4.2.36.3 creation_date

The creation_date specifies the date that the Design_discipline_product_definition was first defined. See 4.3.53 for the application assertion.

4.2.36.4 creator

The creator specifies the Person_organization (see 4.2.118) that created the Design_discipline_product_-definition. See 4.3.56 for the application assertion.

4.2.36.5 description

The description specifies the purpose for a particular definition of the product.

EXAMPLE A laminate_table is a type of design_discipline_product_definition where the description might read: “layup ply table for laminate providing sequence positioning for all plies in the laminate” or “thickness laminate table providing the ply stack at a particular area or point on the laminate.”

4.2.36.6 discipline_identification

The discipline_identification specifies the identification of the functional unit or group within the organization that the definition of the product pertains.

4.2.36.7 version

The version specifies the Part_version (see 4.2.114) being defined. See 4.3.55 for the application assertion.

4.2.37 Design_material

A Design_material is the designation of the part material from a design view. The data associated with a Design_material are the following:

— material_callout;

— Part_defined.

4.2.37.1 material_callout

The material_callout specifies the stock material for the Design_material. The material_callout shall be one of the following: Anisotropic_material (see 4.2.12), Isotropic_material (see 4.2.79), Laminate_table (see 4.2.82). See 4.3.57, 4.3.58 and 4.3.59 for the application assertions.

4.2.37.2 part_defined

The part_defined specifies the part whose material is being defined from a design view. See 4.3.60 for the application assertion.

4.2.38 Design_specification

A Design_specification is a type of Specification (see 4.2.146) that sets the design requirements of parts. These requirements are not set by other design or construction features, or referenced data.

EXAMPLE The requirements set forth in a Design_specification might be weight limits, dimensional limits, reflectivity, color, transparency, feel, and appearance.

4.2.39 Direction

A Direction is a spatial orientation or a line on which something is to be aligned.

4.2.40 Directionally_explicit_element

A Directionally_explicit_element is a type of Element (see 4.2.45). It is a two node element for which the associated matrices are specified directly in a specified direction. The data associated with a Directionally_explicit_element are the following:

— associated_matrix.

4.2.40.1 associated_matrix

The associated_matrix specifies the stiffness, mass, or damping matrix associated with the element. See 4.3.61 for the application assertion.

4.2.41 Discontinuous_fiber_assembly

A Discontinuous_fiber_assembly is a type of Stock_material (see 4.2.154) that is a collection of discontinuous fibers suspended in a homogeneous material (matrix). In general, the length of the fibers are relatively the same within a detail. The orientation of the fibers is usually random.

NOTE 1 When a ply is made from a Discontinuous_fiber_assembly, the Discontinuous_fiber_assembly is in the form of a sheet or is applied as a constant thickness coating to the area defined by the ply.

NOTE 2 Some resin transfer molded (RTM) parts, where the mixture of short fibers and matrix is injected into a mold, can be viewed as a Discontinuous_fiber_assembly.

EXAMPLE Fiberglass chop and mat are types of Discontinuous_fiber_assembly (see Figure 3).

4.2.42 Draped_orientation_angle

A Draped_orientation_angle is a type of Ply_orientation_angle (see 4.2.123) that defines a position and direction at a point within a part.

NOTE The warp direction of a fabric shall be placed in the specified direction prior to draping over the part. Once the fabric has been draped, the orientation angle at any given point must be derived from the draped shape of the fabric. The Draped_orientation_angle will change in the part if the surface of the part has complex curvature.

4.2.43 Edge_zone_shape

An Edge_zone_shape is a type of Zone_structural_makeup_shape_representation (see 4.2.177) that depicts the outline of a zone or area of a part.

NOTE An edge_zone_shape might not be a distinguishable physical edge. It could be where a smooth transition in thickness of the part starts or stops.

The data associated with an Edge_zone_shape are the following:

— boundary.

4.2.43.1 boundary

The boundary specifies the geometry that defines the shape of the zone. The boundary shall be one of the following: Boundary_curve_representation (see 4.2.16), Loop (see 4.2.88). See 4.3.62 and 4.3.63 for the application assertions.

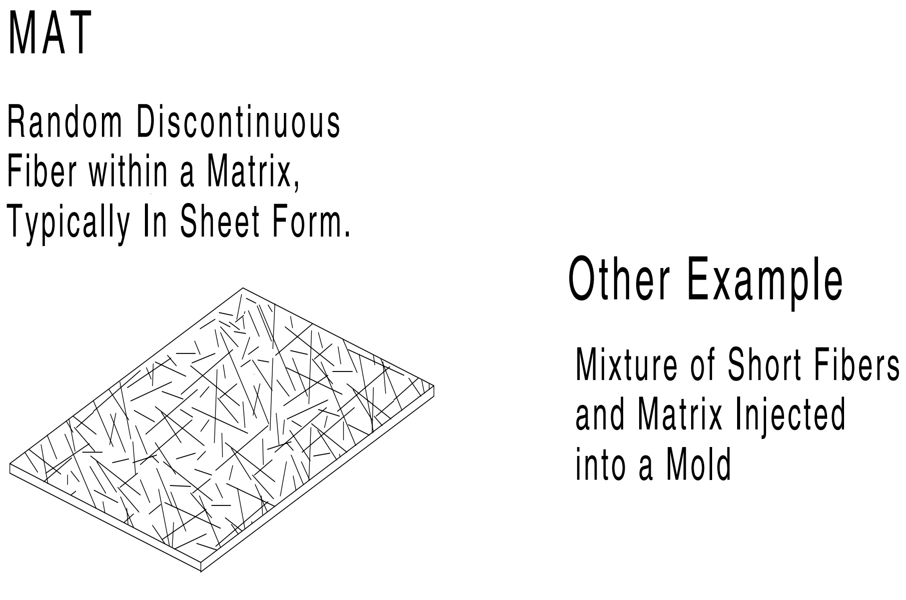

Figure 3 - Discontinuous_fiber_assembly

4.2.44 Effectivity

An Effectivity is the intended use of a part in a particular configuration of a product. An Effectivity shall be one of the following: Date_effectivity (see 4.2.35), Lot_effectivity (see 4.2.89), Sequence_effectivity (see 4.2.140). The data associated with an Effectivity are the following:

— affected_assemblies;

— approved_by;

— configuration_item.

4.2.44.1 affected_assemblies

The affected_assemblies specifies the set of Assembly (see 4.2.14) objects to which the Effectivity applies. See 4.3.65 for the application assertion.

4.2.44.2 approved_by

The approved_by specifies the approval of the Effectivity. The approved_by need not be specified for a particular Effectivity. See 4.3.64 for the application assertion.

4.2.44.3 configuration_item

The configuration_item specifies the Product_configuration (see 4.2.134) to which the Effectivity applies. See 4.3.66 for the application assertion.

4.2.45 Element

An Element is the basic building block of a finite_element_model. It defines the mathematical relationship between the finite element nodes. An Element shall be one of the following: Curve_element (see 4.2.30), Directionally_explicit_element (see 4.2.40), Explicit_element (see 4.2.53), Point_element (see 4.2.128), Substructure_element (see 4.2.156), Surface_element (see 4.2.161), Volume_element (see 4.2.172). The data associated with an Element are the following:

— description;

— element_identification;

— model_ref;

— node_list.

4.2.45.1 description

The description specifies the aspect of a product's shape that the Element represents. The description need not be specified for a particular Element. See 4.3.67 for the application assertion.

4.2.45.2 element_identification

The element_identification specifies an application-defined identifier for a finite element.

4.2.45.3 model_ref

The model_ref specifies the Fea_model (see 4.2.65) to which the Element belongs. See 4.3.68 for the application assertion.

4.2.45.4 node_list

The node_list specifies an ordered list of Node (see 4.2.105) objects that make up the Element. See 4.3.69 for the application assertion.

4.2.46 Element_description

An Element_description is a description of the aspect of a product's shape that an element represents. The data associated with an Element_description are the following:

— description.

4.2.46.1 description

The description describes the aspect of a product's shape that an element represents.

4.2.47 Element_field_variable_definition

An Element_field_variable_definition is a type of Fe_analysis_state_definition (see 4.2.61) that specifies values for a field variable pertaining to the elements of the model.

The field variable information may be:

— evaluated at a point within the volume of an element;

— aggregated through the section of a surface or curve element at a point on the element;

— aggregated within an element, a group of elements, or an entire model;

— evaluated at a point on an element boundary (face of a volume element, or edge of a surface element);

— aggregated over an element boundary.

4.2.48 Element_nodal_freedom_actions

An Element_nodal_freedom_actions is a type of Fe_analysis_state_definition (see 4.2.61) that represents the values of nodal action for an element. The action is applied by the element to the node.

4.2.49 Element_property_geometric_relationship

An Element_property_geometric_relationship is the association of an element property definition to a geometric shape representation. The data associated with an Element_property_geometric_relationship are the following:

— geometry_ref;

— property_ref.

4.2.49.1 geometry_ref

The geometry_ref specifies the Geometry_element (see 4.2.74) that is being associated. See 4.3.71 for the application assertion.

4.2.49.2 property_ref

The property_ref specifies the element property that is being associated to geometry. The property_ref shall be one of the following: Curve_property (see 4.2.31), Surface_property (see 4.2.164). See 4.3.70 and see 4.3.72 for the application assertions.

4.2.50 Element_shape_aspect

An Element_shape_aspect is an aspect of a finite element, such as an element edge, face, or volume.

4.2.51 Element_shape_relationship

An Element_shape_relationship is the association of an aspect of a finite element (an edge, face, or volume) to a corresponding type of geometry. An edge of an element shall correspond to a curve, a face to a surface, and the volume to a solid model. The data associated with an Element_shape_relationship are the following:

— element_aspect;

— element_reference;

— role;

— shape.

4.2.51.1 element_aspect

The element_aspect specifies the aspect of the element that is being associated to the corresponding type of geometry. See 4.3.74 for the application assertion.

4.2.51.2 element_reference

The element_reference specifies the Element (see 4.2.45) whose aspect is being associated. See 4.3.73 for the application assertion.

4.2.51.3 role

The role specifies the function that the Geometry_element (see 4.2.74) plays with respect to the Element.

EXAMPLE The geometry is used to generate the curved edge of an element.

4.2.51.4 shape

The shape specifies the geometric representation whose aspect of the element is being associated. See 4.3.75 for the application assertion.

4.2.52 Environment

An Environment is the temperature, pressure, moisture content and corrosive conditions, such as salt atmosphere, that can affect a material or part.

4.2.53 Explicit_element

An Explicit_element is a type of Element (see 4.2.45). It is a multi-node element in which the associated matrices (stiffness, mass, and damping) are specified directly. The data associated with an Explicit_-element are the following:

— associated_matrix.

4.2.53.1 associated_matrix

The associated_matrix specifies the stiffness, mass, or damping matrix associated with the element. See 4.3.76 for the application assertion.

4.2.54 Face_based_sheet_representation

A Face_based_sheet_representation is a type of Composite_sheet_representation (see 4.2.24) that is defined topologically. The data associated with a Face_based_sheet_representation are the following:

— face_surface;

— inner_bounds;

— outer_bound.

4.2.54.1 face_surface

The face_surface specifies the basis surface for the face. See 4.3.78 for the application assertion.

4.2.54.2 inner_bounds

The inner_bounds specifies a set of topologically defined loops that define any cutouts in the sheet. The inner_bounds need not be specified for a particular Face_based_sheet_representation. See 4.3.77 for the application assertion.

4.2.54.3 outer_bound

The outer_bound specifies the topologically defined loop that is the outer edge of the sheet. See 4.3.77 for the application assertion.

4.2.55 Faceted_boundary_representation

A Faceted_boundary_representation is a type of Geometric_model_representation (see 4.2.72) that represents the shape of a part or an aspect of the shape of a part by a faceted boundary representation solid model. This representation allows for the definition of shapes represented by planar surfaces as the bounding surfaces. Only points and planar polygons are used in the representation. Much of the topology information is implicit for this representation. Shells consist of faces bounded exclusively by planar polygons.

4.2.56 Fe_analysis

An Fe_analysis describes the operations carried out upon the finite element model as a series of analysis steps. A finite element model may have one or more sets of control information aggregated by the Fe_-analysis. The data associated with an Fe_analysis are the following:

— approved_by;

— intended_analysis_code;

— model_ref.

4.2.56.1 approved_by

The approved_by specifies the approval of the analysis operations. The approval indicates acceptance and establishes the Fe_analysis as a particular version of the approval date. The approved_by need not be specified for a particular Fe_analysis. See 4.3.79 for the application assertion.

4.2.56.2 intended_analysis_code

The intended_analysis_code identifies the name of the analysis code to carry out the analysis.

4.2.56.3 model_ref

The model_ref specifies the Fea_model (see 4.2.65) on which the operations specified by the Fe_analysis are carried out upon. See 4.3.80 for the application assertion.

4.2.57 Fe_analysis_control_step

An Fe_analysis_control_step is a single step in an Fe_analysis (see 4.2.56). An Fe_analysis_control_step shall be one of the following: Modes_and_frequencies_control_step (see 4.2.99), Static_control_step (see 4.2.150). The data associated with an Fe_analysis_control_step are the following:

— analysis;

— identification;

— initial_input_state;

— sequence.

4.2.57.1 analysis

The analysis specifies the Fe_analysis to which the step applies. See 4.3.81 for the application assertion.

4.2.57.2 identification

The identification specifies the identifier for the step in the analysis.

4.2.57.3 initial_input_state

The initial_input_state specifies the initial state of the node and element variables for the step. See 4.3.82 for the application assertion.

EXAMPLE Node variables include information such as loads and moments. Element variables include information such as initial strains and pressures.

4.2.57.4 sequence

The sequence identifies the sequence number for the step in the analysis.

4.2.58 Fe_analysis_results

An Fe_analysis_results is the administrative information for a finite element analysis. The data associated with an Fe_analysis_results are the following:

— approved_by.

4.2.58.1 approved_by

The approved_by specifies the approval of the analysis output. The approval indicates that these are acceptable results from an analysis view and establishes the Fe_analysis_results as a particular version of the approval date. The approved_by need not be specified for a particular Fe_analysis_results. See 4.3.83 for the application assertion.

4.2.59 Fe_analysis_results_step

An Fe_analysis_results_step is the state of the node and element analysis variables at the end of a step. An Fe_analysis_results_step shall be one of the following: Modes_and_frequencies_results_step (see 4.2.100), Static_results_step (see 4.2.151). The data associated with an Fe_analysis_results_step are the following:

— control;

— result.

4.2.59.1 control

The control specifies the Fe_analysis_control_step that specified the initial conditions and analysis process for the step. See 4.3.84 for the application assertion.

4.2.59.2 result

The result specifies the administrative information for the finite element analysis. See 4.3.85 for the application assertion.

4.2.60 Fe_analysis_state

An Fe_analysis_state represents the values of information defining the node and element variables within a finite element model at an instant in time. An Fe_analysis_state shall be one of the following: Calculated_state (see 4.2.17), Linearly_superimposed_state (see 4.2.86), Output_request_state (see 4.2.111), Specified_state (see 4.2.147). The data associated with an Fe_analysis_state are the following:

— definitions.

4.2.60.1 definitions

The definitions specifies the aggregation of the variables and their values for the nodes and elements in the finite element model. See 4.3.86 for the application assertion.

EXAMPLE Node variables include such information as loads, moments, displacements, rotations, and reaction forces. Element variables include such information as pressures, stresses, and strains.

4.2.61 Fe_analysis_state_definition

An Fe_analysis_state_definition defines the variables and their values for the nodes and elements in the finite element model. An Fe_analysis_state_definition is both information known before and after an analysis, such as external loadings, and prescribed values of variables at a boundary, and therefore either requests or represents information calculated by the analysis, such as, element stresses, and values of variables at unconstrained nodes. In this manner, state definition output from one step may be used as input for a subsequent state definition. An Fe_analysis_state_definition shall be one of the following: Analysis_message (see 4.2.7), Element_field_variable_definition (see 4.2.47), Element_nodal_freedom_-actions (see 4.2.48, 4.2.49), Linear_constraint_equation_value (see 4.2.85), Nodal_freedom_definitions (see 4.2.103), Single_point_constraint_values (see 4.2.144). The data associated with an Fe_analysis_-state_definition are the following:

— ref.

4.2.61.1 ref

The ref specifies the node, element, geometry_element, or group whose analysis variables, or analysis variables and their values are being specified. The ref shall be one of the following: Element (see 4.2.45), Geometry_element (see 4.2.74), Group (see 4.2.76), Node (see 4.2.105). See 4.3.87, 4.3.88, 4.3.89 and 4.3.90 for the application assertions.

4.2.62 Fea_material_definition

An Fea_material_definition aggregates the specifications and properties of matter that a finite element models.

NOTE For a surface element of layered composite material, the total thickness for the element is defined by adding up all the ply thicknesses in the material.

The data associated with an Fea_material_definition are the following:

— description;

— elements;

— material_identification.

4.2.62.1 description

The description specifies a text narrative about the substance.

4.2.62.2 elements

The elements specifies the aggregation of finite elements or geometric items whole material response is defined by the Fea_material_definition. The elements may be a set of one or more of the following: Curve_element (see 4.2.30), Shape_aspect (see 4.2.142), Surface_element (see 4.2.161), Volume_-element (see 4.2.172). See 4.3.91, 4.3.92, 4.3.93 and 4.3.94 for the application assertions.

4.2.62.3 material_identification

The material_identification specifies a unique application-defined identifier for the material to be used in the Fe_analysis (see 4.2.56).

4.2.63 Fea_material_property

An Fea_material_property is a type of Material_property (see 4.2.95) that is used in an Fea_model.

The mechanical properties for a fabric material include:

— Density;

— Warp Young's modulus - compression;

— Fill Young's modulus - compression;

— Warp Young's modulus - tension;

— Fill Young's modulus - tension;

— Through the thickness Young's modulus (33);

— Poisson's ratio (12, 23, 33);

— Shear modulus (12, 23, 33);

— Warp thermal coefficient;

— Fill thermal coefficient;

— Through the thickness thermal coefficient (33);

— Warp moisture absorption coefficient;

— Fill moisture absorption coefficient;

— Through the thickness moisture absorption coefficient (33).

The strain allowable properties for a fabric material include:

— Warp compression;

— Fill compression;

— Through the thickness compression (33);

— Warp tension;

— Fill tension;

— Through the thickness tension (33);

— Positive shear (12, 23, 13);

— Negative shear (12, 23, 13).

The stress allowable properties for a fabric material include:

— Warp compression;

— Fill compression;

— Through the thickness compression (33);

— Warp tension;

— Fill tension;

— Through the thickness tension (33);

— Positive shear (12, 23, 13);

— Negative shear (12, 23, 13).

The mechanical properties for general composite and metallic materials include:

— Density;

— Young’s modulus - compression (11, 22, 33);

— Young’s modulus - tension (11, 22, 33);

— Poisson's ratio (12, 23, 33);

— Shear modulus (12, 23, 33);

— Thermal coefficient (11, 22, 33);

— Moisture absorption coefficient (11, 22, 33).

The strain allowable properties for general composite and metallic materials include:

— Compression (11, 22, 33);

— Tension (11, 22, 33);

— Positive shear (12, 23, 13);

— Negative shear (12, 23, 13).

The stress allowable properties for general composite and metallic materials include:

— Compression (11, 22, 33);

— Tension (11, 22, 33);

— Positive shear (12, 23, 13);

— Negative shear (12, 23, 13).

The fastener analysis material properties include:

— Characteristic dimension - compression;

— Characteristic dimension - tension.

The response matrix material properties include:

— Shell membrane stiffness;

— Shell shear stiffness;

— Shell bending stiffness;

— Shell membrane-bending stiffness;

— 3D anisotropic stiffness;

— 3D isotropic stiffness;

— 3D orthotropic stiffness;

— 3D transverse orthotropic stiffness;

— 3D column normalized orthotropic stiffness;

— 3D column normalized monoclinic stiffness.

NOTE The above numbers refer to the three directions in the material.

The data associated with an Fea_material_property are the following:

— property_use.

4.2.63.1 property_use

The property_use specifies the Fea_material_definition (see 4.2.62) that uses the material property. See 4.3.95 for the application assertion.

4.2.64 Fea_material_specification

An Fea_material_specification is a type of Material_specification (see 4.2.96) used in a finite element model. The data associated with an Fea_material_specification are the following:

— specification_use.

4.2.64.1 specification_use

The specification_use specifies the Fea_material_definition (see 4.2.62) that uses the material specification. See 4.3.96 for the application assertion.

4.2.65 Fea_model

An Fea_model is a collection of information that represents the finite element analysis of a product. The information includes nodes, elements, materials, properties, and groups which are combined to form a discrete mesh model of the product.

The data associated with an Fea_model are the following:

— analysis_type;

— approved_by;

— creating_software;

— definition;

— description;

— fea_filename;

— identification;

— intended_analysis_code.

4.2.65.1 analysis_type

The analysis_type describes what type of analysis is to be performed with the Fea_model.

4.2.65.2 approved_by

The approved_by specifies the approval of the Fea_model. The approved_by need not be specified for a particular Fea_model. See 4.3.98 for the application assertion.

4.2.65.3 creating_software

The creating_software specifies the name of the software used to create the Fea_model. The version of the software should be specified.

4.2.65.4 definition

The definition specifies the Analysis_discipline_product_definition that the Fea_model defines for analysis purposes. See 4.3.97 for the application assertion.

4.2.65.5 description

The description specifies an aspect of a product's shape that the Fea_model represents. See 4.3.99, 4.3.100 for the application assertion.

4.2.65.6 fea_filename

The fea_filename references the file that contains the information describing the Fea_model. The fea_filename need not be specified for a particular Fea_model.

4.2.65.7 identification

The identification specifies the application-defined identifier for the Fea_model.

4.2.65.8 intended_analysis_code

The intended_analysis_code specifies the name of the analysis code that the Fea_model was created for.

4.2.66 Fea_model_description

An Fea_model_description is a description of the aspect of a product's shape that a finite element model represents. The data associated with an Fea_model_description are the following:

— description.

4.2.66.1 description

The description describes the aspect of a product's shape that a finite element model represents.

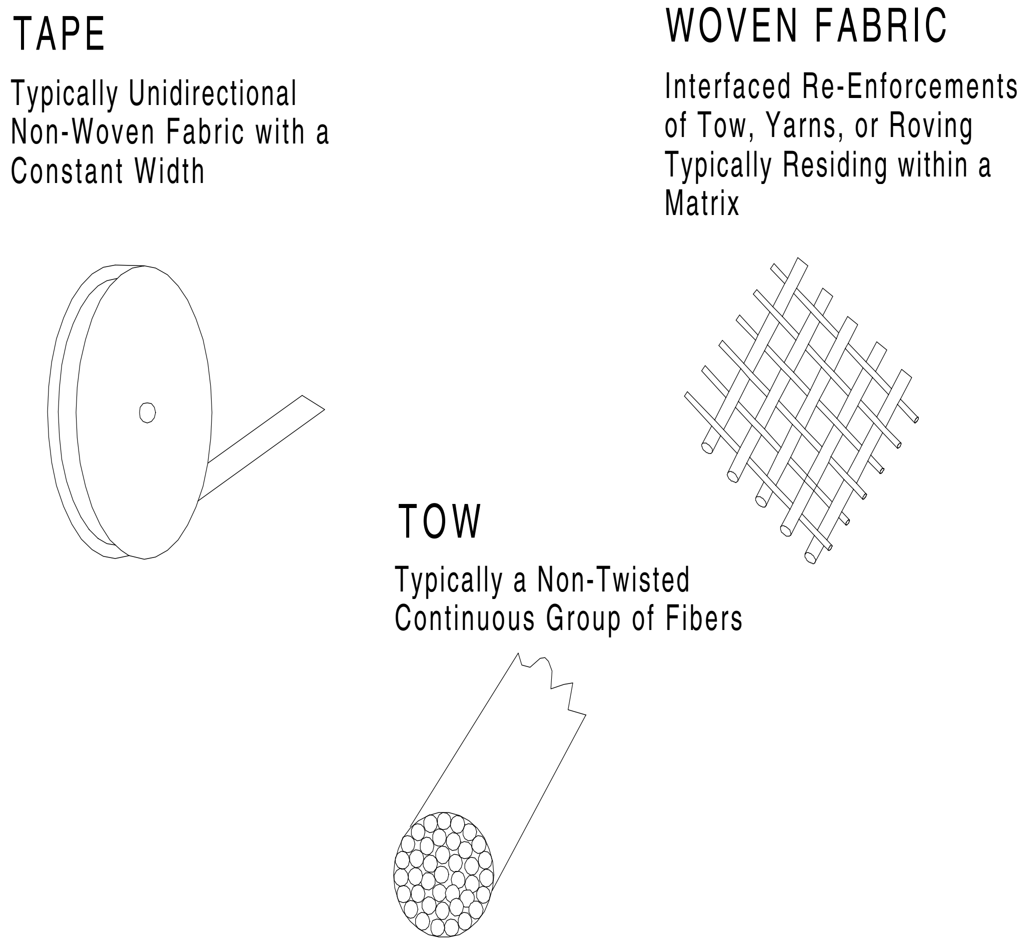

Figure 4 - Filament_assembly

4.2.67 Filament_assembly

A Filament_assembly is a type of Stock_material (see 4.2.154) that is a collection of yarns or tows combined together in some manner and frequently embedded in a matrix of homogenous material. The Filament_assembly is usually measured by weight and sometimes length to describe the amount needed for a Filament_laminate.

NOTE 1 When a ply is made from a Filament_assembly it is usually referred to as woven or non-woven fabric such as broad goods and tape. A ply can also be made from such things as a collection of tows or yarn provided the tow or yarn lay flat in the area described by the ply and they are a single thickness that does not stack on top of one another or overlap.

NOTE 2 The thickness or cross-sectional area of a Filament_assembly is specified in the ply or Filament_laminate that is made from the stock material. These are usually specified in cured thickness or cross-sectional area.

EXAMPLE Woven and non-woven fabric, individual tows, yarns, and roving are types of filament assemblies (see Figure 4).

4.2.68 Filament_laminate

A Filament_laminate is a type of Constituent_part (see 4.2.25) that is a bonded material of two or more bundles of filament construction. The Filament_laminate defines a volume to fill with filament assemblies.

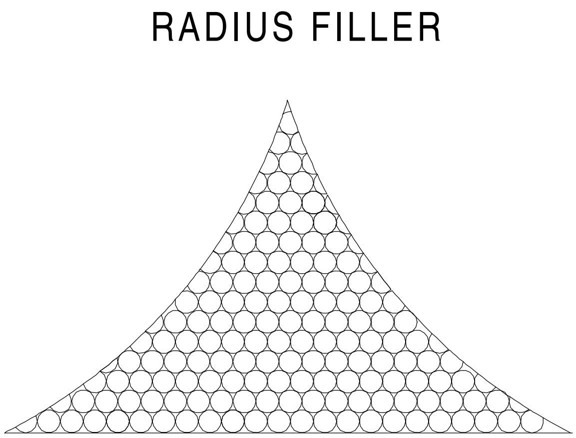

Figure 5 - Filament_laminate

NOTE 1 The shape of a Filament_laminate may be defined by a cross section and a length.

NOTE 2 When a Filament_laminate is made from a Filament_assembly (see 4.2.67), it usually refers to individual tows, yarns, and rovings. These volumes can be as simple as the cross section and length of a radius filler to as complex as a filament wound vessel. Filament_laminates that are made by pulling and forming processes, called pultrusions, may use a tape form of Filament_assembly.

NOTE 3 The difference between a ply and a Filament_laminate, when referencing a Filament_assembly, is that a ply describes a constant thickness area to be filled by a Filament_assembly and a Filament_laminate describes a general volume to be filled by a Filament_assembly.

EXAMPLE Pultruded shapes, radius fillers, and filament wound structure are types of Filament_laminate (see Figure 5).

The data associated with a Filament_laminate are the following:

— made_from;

— shape.

4.2.68.1 made_from

A made_from specifies the stock Filament_assembly that the Filament_laminate is made from. The Filament_assembly is in the form of a tow, yarn or roving, and sometimes tape. See 4.3.101 for the application assertion.

4.2.68.2 shape

The shape specifies the geometry that defines the cross-sectional area of a Filament_laminate. The shape shall be one of the following: Idealized_analysis_shape (see 4.2.78), Nominal_design_shape (see 4.2.109). The shape need not be specified for a particular Filament_laminate. See 4.3.102 for the application assertion.

4.2.69 Filament_laminate_shape

A Filament_laminate_shape is a type of Ply_shape (see 4.2.125). It is a two-dimensional geometric shape representation for the cross-sectional area of a Filament_laminate (see 4.2.68). The data associated with a Filament_laminate_shape are the following:

— cross_section.

4.2.69.1 cross_section

The cross-section represents the cross-sectional area of a Filament_laminate. See 4.3.103 and 4.3.104 for the application assertions.

4.2.70 File

A File is an ordered collection of data existing in a computer storage device.

4.2.71 Flat_pattern_ply_shape

A Flat_pattern_ply_shape defines the edge boundary of a ply in a plane. When the shape is laid up in the assembly, it will have the proper size to define the desired ply shape (see Figure 6).

The data associated with a Flat_pattern_ply_shape are the following:

— wrapup_origin_on_flat_pattern;

— wrapup_origin_on_surface.

4.2.71.1 wrapup_origin_on_flat_pattern

The wrapup_origin_on_flat_pattern specifies the point of the flat pattern plane that corresponds directly to the wrapup_origin_on_surface. See 4.3.105 for the application assertion.

4.2.71.2 wrapup_origin_on_surface

The wrapup_origin_on_surface specifies the point on the basis surface that corresponds directly to the wrapup_origin_on_flat_pattern. See 4.3.105 for the application assertion.

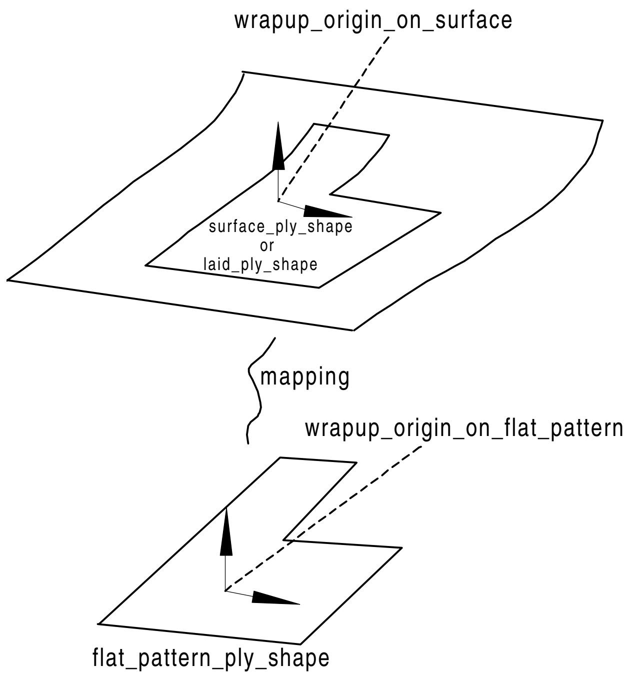

Figure 6 - Flat_pattern_ply_shape

4.2.72 Geometric_model_representation

A Geometric_model_representation defines the shape or a portion of the shape of a part. A Geometric_-model_representation shall be one of the following: Advanced_boundary_representation (see 4.2.2), Constituent_shape_representation (see 4.2.26), Faceted_boundary_representation (see 4.2.55), Manifold_surface_with_topology (see 4.2.91), Non_topological_surface_and_wireframe (see 4.2.110), Point_model (see 4.2.129), Wireframe_with_topology (see 4.2.173).
The data associated with a Geometric_model_representation are the following:

— elements;

— role.

4.2.72.1 elements

The elements specifies the set of Geometry_element (see 4.2.74) objects that make up the model. See 4.3.106 for the application assertion.

4.2.72.2 role

The role specifies the function that a geometric element plays with respect to the geometric model.

EXAMPLE The geometry is used to generate the position of a node.

4.2.73 Geometric_sheet_representation

A Geometric_sheet_representation is a type of Composite_sheet_representation (see 4.2.24) that is defined by a surface and a set of curves. These curves are related in the application to that surface. The curves define a closed area but are not necessarily restrained topologically. The data associated with a Geometric_sheet_representation are the following:

— basis_surface;

— cutouts;

— outer_edge.

4.2.73.1 basis_surface

The basis_surface specifies the Surface (see 4.2.160) used to define shape and having some known context to the part being represented. The surface may be the layup surface, inner-mold-line surface, outer-mold-line surface, or other. See 4.3.108 for the application assertion.

4.2.73.2 cutouts

The cutouts specifies the sets of curves that define the areas to be removed from the interior of the part. The cutouts need not be specified for a particular Geometric_sheet_representation. See 4.3.107 for the application assertion.

4.2.73.3 outer_edge

The outer_edge specifies the set of curves that define the outside edge (or boundary) of the part. See 4.3.107 for the application assertion.

4.2.74 Geometry_element

A Geometry_element is the geometry object that defines a portion of a part’s shape.

4.2.75 Graphical_representation

A Graphical_representation is a type of Analysis_report_representation (see 4.2.8). A Graphical_-representation may be a two-dimensional vector graphical representation of the following:

— finite element model maps;

— analysis output information displays upon finite element model mesh;

— line drawing sketches to document part components subjected to detail analyses.

The finite element model maps displayed on the finite element model mesh may include:

— the node numbers;

— the element numbers;

— the node attributes such as loads and constraints;

— the element attributes such as properties.

Analysis output displayed on the finite element model mesh may include:

— the deflected upon undeflected shape;

— the nodal value contours;

— the element value contours;

— the element value tags;

— the node force balance freebody loads.

The data associated with a Graphical_representation are the following:

— graphical_filename.

4.2.75.1 graphical_filename

The graphical_filename specifies the name of the file containing the graphical information.

4.2.76 Group

A Group is an aggregation of elements or nodes. The data associated with a Group are the following:

— description;

— elements;

— group_identification;

— nodes.

4.2.76.1 description

The description specifies the label used for group identification.

EXAMPLE Set1 and wing are labels used for group identification.

4.2.76.2 elements

The elements specifies the set of Element (see 4.2.45) objects that make up the Group. The elements need not be specified for a particular Group. See 4.3.109 for the application assertion.

4.2.76.3 group_identification

The group_identification specifies the application-defined identifier for the Group, and is unique with the finite element model.

4.2.76.4 nodes

The nodes specifies the set of Node (see 4.2.105) objects that make up the Group. The nodes need not be specified for a particular Group. See 4.3.110 for the application assertion.

4.2.77 Group_relationship

A Group_relationship is an association between Group (see 4.2.76) objects. A Group_relationship may be a parent-child relationship. The data associated with a Group_relationship are the following:

— related_group;

— relating_group.

4.2.77.1 related_group

The related_group specifies one of the two Group objects in the relationship. If one of the groups is dependent on the other, the related_group shall be the parent group. See 4.3.111 for the application assertion.

4.2.77.2 relating_group

The related_group specifies the other (child) Group in the relationship. See 4.3.111 for the application assertion.

4.2.78 Idealized_analysis_shape

An Idealized_analysis_shape is a type of Shape (see 4.2.141) and Analysis_shape (see 4.2.9). It represents a shape on which points are associated for the location of nodes in a finite element model, or a shape that is suitable for mesh generation. Mesh generation shape may include the topological information necessary to specify mesh generation curves, areas, or volumes, or may be just the bounding geometry of the mesh generation curves, areas, or volumes. The geometry may be the Nominal_design_-shape (see 4.2.109), or geometry that has been idealized by generating shape information that is derived from the Nominal_design_shape. The data associated with an Idealized_analysis_shape are the following:

— basis;

— defining_shape.

4.2.78.1 basis

The basis specifies the nominal_design_shape that the model idealizes for analysis purposes. See 4.3.113 for the application assertion.

4.2.78.2 defining_shape

The defining_shape specifies the geometric model that defines the shape. See 4.3.112 for the application assertion.

4.2.79 Isotropic_material

An Isotropic_material is a type of Stock_material (see 4.2.154) that is made up of a single uniform internal structure.

NOTE 1 When Isotropic_material such as resin plays the role of a matrix within other stock materials, the Isotropic_material is combined with filaments to make filament assemblies and with fibers to make discontinuous fiber assemblies.

NOTE 2 When Isotropic_material such as stabilizers, foaming and film adhesives are applied separately from other stock materials, a ply references the Isotropic_material to provide placement and thickness.

NOTE 3 When Isotropic_material are such things as blocks of material that can be described by a fixed shape, they can either be considered stock_core or remain Isotropic_material if they are not going to be combined with other stock_material to produce a composite part. The latter could take on the part types of extrusion, plate, bar, casting, forging, and tubing.

EXAMPLE Resins, stabilizers, foaming and film adhesives are types of Isotropic_material. Also metals in general can be considered Isotropic_material.

4.2.80 Laid_orientation_angle

A Laid_orientation_angle is type of Ply_orientation_angle (see 4.2.123) that explicitly defines a direction at a point within a ply.

4.2.81 Laid_ply_shape

A Laid_ply_shape is a type of Ply_shape (see 4.2.125) that defines the shape of the ply in the actual Ply_-laminate (see 4.2.120). This is the physical shape that will stair up and down over the build-up of previously laid plies (see Figure 7).

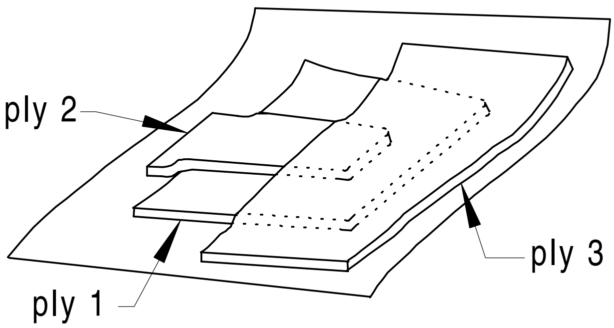

Figure 7 - Laid_ply_shape

4.2.82 Laminate_table

A Laminate_table defines the internal constituents or structural makeup of an overall part or a particular zone, area, or point on the part.

NOTE 1 There are two functions provided by a Laminate_table. The first is to describe the order in which the physical constituents of a laminate are assembled. This is called a Part_laminate_table (see 4.2.113). The second is to describe the structural makeup of a particular zone, area, or point within a laminate. This is called a Zone_-structural_makeup (see 4.2.176).

NOTE 2 A Part_laminate_table is usually formalized by a structural designer that prepares the part description information package for manufacturing. A Zone_structural_makeup is usually created to assist in the communication of a part's structural description between the analysis and design functions.

A Laminate_table shall be one of the following: Part_laminate_table (see 4.2.113), Zone_structural_-makeup (see 4.2.176). The data associated with a Laminate_table are the following:

— base_surface;

— basis;

— properties;

— resulting_surface.

4.2.82.1 base_surface

The base_surface specifies the surface used to define the lamination or layers. The layers or lamination are defined to start from the surface and to be laminated or layered in the specified direction and order. The base_surface may but need not be the layup_surface. The base_surface need not be specified for a particular Laminate_table. See 4.3.117 for the application assertion.

4.2.82.2 basis

The basis specifies the orientation direction for the load carrying reinforcements within the Laminate_-table. See 4.3.115 for the application assertion.

4.2.82.3 properties

The properties specifies the Fea_material_property that defines the physical response of the laminate. The properties need not be specified for a particular Laminate_table. See 4.3.114 for the application assertion.

4.2.82.4 resulting_surface

The resulting_surface specifies the opposing surface that forms from the physical material being placed on the base_surface. The resulting_surface need not be specified for a particular Laminate_table. See 4.3.116 for the application assertion.

4.2.83 Length_measure

A Length_measure is a measured distance or assigned dimension.

4.2.84 Linear_constraint_equation

A Linear_constraint_equation is a type of Constraint (see 4.2.27). It is a linear multi-point constraint equation that specifies the nodal freedoms and their coefficients. The value of the constraint equation coefficient is set by a Linear_constraint_equation_value (see 4.2.85) object. The data associated with a Linear_constraint_equation are the following:

— freedoms_and_coefficients_nodes.

4.2.84.1 freedoms_and_coefficients_nodes

The freedoms_and_coefficients_nodes specifies the set of nodes, groups, or geometric elements associated with the freedoms and their coefficients for each term of the Linear_constraint_equation. The freedoms_and_coefficients_nodes shall be a set of one or more of the following: Geometry_element (see 4.2.74), Group (see 4.2.76), Node (see 4.2.105). See 4.3.118, 4.3.119 and 4.3.120 for the application assertions.

4.2.85 Linear_constraint_equation_value

A Linear_constraint_equation_value is a type of Fe_analysis_state_definition (see 4.2.61) that defines the constraint equation coefficient associated with the freedoms and their coefficients. The data associated with a Linear_constraint_equation_value are the following:

— equation.

4.2.85.1 equation

The equation specifies the Linear_constraint_equation to which the coefficient is applied. See 4.3.121 for the application assertion.

4.2.86 Linearly_superimposed_state

A Linearly_superimposed_state is a type of Fe_analysis_state (see 4.2.60) representing information that is obtained by a combination of previously calculated and specified states.

4.2.87 Location

A Location is a definition of the physical position of an object often in relationship to another object. The indicates both a position and a direction.

4.2.88 Loop

A Loop is a topologically closed boundary made up of edges.

4.2.89 Lot_effectivity

A Lot_effectivity is a type of Effectivity (see 4.2.44). It is the specification of the use of a part in a product_configuration where the part is produced as one of a group. Lots are used when parts are produced in batches and/or when important characteristics might vary between production runs. The data associated with a Lot_effectivity are the following:

— lot_number;

— lot_size;

— lot_size_unit_of_measure.

4.2.89.1 lot_number

The lot_number specifies the identification of the group of parts that compose a lot.

4.2.89.2 lot_size

The lot_size specifies the quantity of parts within the lot.

4.2.89.3 lot_size_unit_of_measure

The lot_size_unit_of_measure specifies the fixed quantity amount, in terms of which the lot_size is expressed.

4.2.90 Make_from

A Make_from is a relationship between two parts wherein one part is used as the basis for the design of the other part.

EXAMPLE Company A buys a cup holder from company B and adds mounting holes to fasten the cup holder rather than using the gluing designed by company B. In this case, company A identifies the cup holder designed by company B as a Make_from.

The data associated with a Make_from are the following:

— basis;

— resultant.

4.2.90.1 basis

The basis specifies the part that serves as the basis for the design of the other part in the relationship. See 4.3.122 for the application assertion.

4.2.90.2 resultant

The resultant specifies the part whose design is based on the other part in the relationship. See 4.3.122 for the application assertion.

4.2.91 Manifold_surface_with_topology

A Manifold_surface_with_topology is a type of Geometric_model_representation (see 4.2.72) that represents the shape of a part or an aspect of the shape of a part using manifold surfaces with topology. Three-dimensional curves, surfaces, and topology are used to define the outer boundary of the part.

4.2.92 Mass_matrix

A Mass_matrix is a type of Matrix (see 4.2.97). It is an explicit representation of mass at a node, between two nodes in a direction, or for many nodes.

4.2.93 Mass_measure

A Mass_measure is a measure of the amount of matter that a body contains.

4.2.94 Material_direction

A Material_direction is the identification of the principal direction of a material.

NOTE A woven fabric will have both a warp and fill directions. The Material_direction shall identify the warp direction for orientation in the laminate. The Material_direction for a honeycomb core shall be that of the ribbon direction.

The data associated with a Material_direction are the following:

— material_orientation.

4.2.94.1 material_orientation

The material_orientation specifies the principal direction within the material. See 4.3.123 for the application assertion.

4.2.95 Material_property

A Material_property is a quality of a material that is as measured by specific procedure and obtained by an accepted analysis. A Material_property may be an Fea_material_property (see 4.2.63). The data associated with a Material_property are the following:

— ambient;

— property_name;

— property_value.

4.2.95.1 ambient

The ambient specifies the environmental conditions used to define the material property. See 4.3.124 for the application assertion.

4.2.95.2 property_name

The property_name specifies the label given to the property to describe what quality it represents.

4.2.95.3 property_value

The property_value specifies the actual value that represents the quality of the material. The property_-value need not be specified for a particular Material_property. See 4.3.125 for the application assertion.

4.2.96 Material_specification

A Material_specification is a type of Specification (see 4.2.146) that specifies properties of raw material, mixtures, or semi-fabricated material that are used in the fabrication of a product. It is an informal or formal contractual, internal, or institutional document that details the production of qualities of a material. For composite materials, the specification may include the following information:

— the percentages of fiber in the warp and fill directions;

— the text description of the weave of the fiber;

— the fiber volume content of the material;

— the resin content of the material; and

— the text description of sizing on tows prior to weaving and resin impregnation.

A Material_specification may be an Fea_material_specification (see 4.2.64).

4.2.97 Matrix

A Matrix is an explicit representation of the damping, mass, or stiffness information for a point or explicit element. A Matrix shall be one of the following: Damping_matrix (see 4.2.33), Mass_matrix (see 4.2.93), Stiffness_matrix (see 4.2.152).

4.2.98 Measure_value

A Measure_value specifies a value for a measured characteristic of an object.

4.2.99 Modes_and_frequencies_control_step

A Modes_and_frequencies_control_step is a type of Fe_analysis_control_step (see 4.2.57) that represents a single step in a modes and frequencies analysis. The data associated with a Modes_and_frequencies_-control_step are the following:

— final_input_state;

— frequency_range;

— number_of_modes.

4.2.99.1 final_input_state

The final_input_state specifies the final equilibrium state for a Modes_and_frequencies_control_step. See 4.3.126 for the application assertion.

4.2.99.2 frequency_range

The frequency_range specifies the values of the lower and upper frequency bounds.

4.2.99.3 number_of_modes

The number_of_modes specifies the number of modes to be calculated in the analysis.

4.2.100 Modes_and_frequencies_results_step

A Modes_and_frequencies_results_step is a type of Fe_analysis_results_step (see 4.2.58) that defines the state of node and element variables at the end of a modes and frequencies analysis step. The data associated with a Modes_and_frequencies_results_step are the following:

— resulting_states.

4.2.100.1 resulting_states

The resulting_states specifies the set of Calculated_state (see 4.2.17) objects that result from the modes and frequencies analysis. There is one state per mode and frequency pair. See 4.3.127 for the application assertion.

4.2.101 Next_higher_assembly

A Next_higher_assembly is a type of Assembly (see 4.2.14) that defines the relationship of a part to its immediate parent within an assembly hierarchy.

The data associated with a Next_higher_assembly are the following:

— as_required;

— component_quantity;

— reference_designator;

— unit_of_measure.

4.2.101.1 as_required

The as_required specifies whether the component quantity is explicitly defined (false), or is to be quantified as needed in a particular usage (true).

EXAMPLE When specifying the amount of sheet metal that is to be consumed in making a part, one roll might be specified, where one is the quantity and roll is the unit_of_measure. Alternately, rolls of sheet metal might be used as required. In this case, as_required is the quantity and roll is again the unit_of_measure.

4.2.101.2 component_quantity

The component_quantity specifies the count of a component part found in an immediately higher assembly if the as_required is false.

4.2.101.3 reference_designator

The reference_designator specifies the unique identifier that distinguishes a particular instance of a component in an assembly where more than one of a particular component part is used in an assembly.

EXAMPLE If four identical wheels are used in a car, the reference_designator distinguishes the left front wheel, for example, from the others.

The reference_designator need not be specified for a particular Next_higher_assembly.

4.2.101.4 unit_of_measure

The unit_of_measure specifies the fixed quantity in terms of which the component_quantity is expressed.

4.2.102 Nodal_degree_of_freedom_reduction

A Nodal_degree_of_freedom_reduction is a type of Constraint (see 4.2.27) that constrains the degrees of freedom for a node, for a group, or for the nodes associated with a geometry element, in an Fea_model. The data associated with a Nodal_degree_of_freedom_reduction are the following:

— node.

4.2.102.1 node

The node specifies where the Nodal_degree_of_freedom_reduction is applied. The node shall be one of the following: Geometry_element (see 4.2.74), Group (see 4.2.76), Node (see 4.2.105). See 4.3.128, 4.3.129 and 4.3.130 for the application assertions.

4.2.103 Nodal_freedom_definitions

A Nodal_freedom_definitions is a type of Fe_analysis_state_definition (see 4.2.61) that specifies information at a node of the model with respect to the solution degrees of freedom. The information may be:

— values for the solution degrees of freedom, such as x- translation, or z-rotation;

— loads applied in the direction of the solution degrees of freedom such as values for x-force, or z-moment. These may be loads applied to a node by elements of the model, or loads applied to a node from outside the model.

4.2.104 Nodal_results_coordinate_system

A Nodal_results_coordinate_system is the coordinate system for the node analysis variables in an Fea_model (see 4.2.65).

4.2.105 Node

A Node is a discretization point for the field variables of the Fea_model (see 4.2.65). The data associated with a Node are the following:

— description;

— location;

— model_ref;

— node_identification;

— results_coordinate_space.

4.2.105.1 description

The description specifies the aspect of a product's shape that the Node represents. The description need not be specified for a particular Node. See 4.3.134 for the application assertion.

4.2.105.2 location

The location specifies the coordinates of the Node with respect to the founding placement coordinate system. The point may be a position on a curve or surface in an idealized_analysis_shape, or a position in a Node_shape. See 4.3.132 for the application assertion.

4.2.105.3 model_ref

The model_ref specifies the Fea_model (see 4.2.65) that the Node is defined in. See 4.3.131 for the application assertion.

4.2.105.4 node_identification

The node_identification specifies the unique application-defined identifier for the Node.

4.2.105.5 results_coordinate_space

The results_coordinate_space specifies the coordinate system for the analysis variables at the Node. The results_coordinate_space need not be specified for a particular Node. See 4.3.133 for the application assertion.

4.2.106 Node_description

A Node_description describes the aspect of a product's shape that a node represents. The data associated with a Node_description are the following:

— description.

4.2.106.1 description

The description describes the aspect of a product's shape that a node represents.

4.2.107 Node_shape

A Node_shape is a type of Shape (see 4.2.141) and Analysis_shape (see 4.2.9) that is defined by the points of the nodes in a Fea_model. The data associated with a Node_shape are the following:

— defining_shape.

4.2.107.1 defining_shape

The defining_shape specifies the group of points that define the shape. See 4.3.135 for the application assertion.

4.2.108 Node_shape_relationship

A Node_shape_relationship is the association of the position of a Node (see 4.2.101) to any type of geometry. The data associated with a Node_shape_relationship are the following:

— node_reference;

— role;

— shape.

4.2.108.1 node_reference

The node_reference specifies the Node that is being associated. See 4.3.137 for the application assertion.

4.2.108.2 role

The role specifies the function that a geometric element plays with respect to the Node.

EXAMPLE The geometry is used to generate the location of a node.

4.2.108.3 shape

The shape specifies the geometric representation to which the Node is being related. See 4.3.136 for the application assertion.

4.2.109 Nominal_design_shape

A Nominal_design_shape is a type of Shape (see 4.2.141) that is defined by the design discipline. The data associated with a Nominal_design_shape are the following:

— defining_shape;

— design_view.

4.2.109.1 defining_shape

The defining_shape specifies the geometric model that defines the shape. See 4.3.139 for the application assertion.

4.2.109.2 design_view

The design_view specifies the Design_discipline_product_definition (see 4.2.35.2) for the Nominal_-design_shape. See 4.3.138 for the application assertion.

4.2.110 Non_topological_surface_and_wireframe

A Non_topological_surface_and_wireframe is a type of Geometric_model_representation (see 4.2.72) that represents the shape or portions of the shape of a part using surface or wireframe geometry without topology. These representations are formed by the use of points, curves, and surfaces only. The boundaries of the curves are defined explicitly by points on the curves and explicit associations between the points and the curves. The boundaries of the surfaces are defined by curves on the surfaces and explicit associations between the curves and the surfaces. Surfaces and curves must be explicitly trimmed unless they are closed.

4.2.111 Output_request_state

An Output_request_state is a type of Fe_analysis_state (see 4.2.60) requesting information in the form of computer readable Fe_analysis_state_definition (see 4.2.61) objects. The data associated with an Output_request_state are the following:

— steps.

4.2.111.1 steps

The steps specifies the set of Fe_analysis_control_step (see 4.2.57) objects to which the output requests apply. See 4.3.140 for the application assertion.

4.2.112 Part

A Part is an item that is intended to be produced or employed in a production process. The item refers not only to the finished part but also to any physical constituent or in process configuration that makes up the finished part. Part can also describe stock material where more than bill-of-material reference information is required.

NOTE The AP handles both composite structural parts and homogeneous (isotropic) structural parts.

The data associated with a Part are the following:

— owner;

— part_classification;

— part_nomenclature;

— part_number;

— part_type;

— standard_part.

4.2.112.1 owner

The owner specifies the Person_organization (see 4.2.118) that owns the Part. See 4.3.141 for the application assertion.

4.2.112.2 part_classification

The part_classification specifies a family of related parts that have common processes applied to them during manufacturing. The part_classification need not be specified for a particular Part.

4.2.112.3 part_nomenclature

The part_nomenclature specifies a name by which the Part is commonly known within an organization. Usually the name reflects the form and/or function of the Part.

NOTE The name can also have meaning outside of any single organization. A single nation or industry can also have naming conventions for different kinds of common parts.

EXAMPLE Skin Panel, "T" Stiffener, Access Door, Core Stiffened Panel, Leading Edge Fairing

4.2.112.4 part_number

A part_number specifies the unique identification of the Part for a particular organization.

EXAMPLE A drawing number with a dash number "BTES930032-001". A ply might have an identification of "P02".

4.2.112.5 part_type

The part_type specifies one of a set of kinds of parts. It shall always be one of four values: detail, assembly, inseparable_assembly, or customer_furnished_equipment. A Part whose part_type is detail exists at the lowest level of the bill-of-material. A Part whose part_type is assembly consists of a collection of other Part objects that are put together in order to satisfy a particular function. A Part whose part_type is inseparable_assembly is an assembly that cannot be dis-assembled without causing physical harm to at least one the components in the assembly. A Part whose part_type is customer_furnished_-equipment is a Part that has been furnished to the design from a customer source.

4.2.112.6 standard_part

The standard_part is an indicator that specifies whether or not a Part has a design specified externally to the bill-of-material in which an assembly is being defined.

NOTE A Part may be a standard part for a company, industry or other organization.

4.2.113 Part_laminate_table

A Part_laminate_table is a type of Laminate_table (see 4.2.82) that applies to the overall laminate. It is used to provide allocation of each physical constituent through the use of an ordered placement list. A Part_laminate_table shall be one of the following: Composite_assembly_table (see 4.2.23), Ply_-laminate_table (see 4.2.122).

NOTE 1 The information a Part_laminate_table associates together will satisfy the assembly specification requirements of composite constituents only if each constituent of a Part_laminate_table's shape representation has been properly established in its appropriate location in three-dimensional space in the laminate. If this is not the case, or if the composite constituent is a constituent of a Composite_assembly_table that has two or more instances in the table, then the associated assembly information in the product structure specified by Next_higher_assembly and Component_assembly_position entities will provide the necessary placement information to establish the three-dimensional location of the instance of the constituent.

EXAMPLE A composite "T" that is made up of two angles and a cap would have the angle constituent specified once, and then pointed to twice by the Composite_assembly_table. A single instance of the cap would also be pointed to by the table. The location of each of these constituents would then be specified by the assembly information in the product structure specified by the Next_higher_assembly and Component_assembly_poisition entities that reference the Design_discipline_product_defintion that defines the shape of the Part_version of each of the instances of the constituents in the Composite_assembly_table.

NOTE 2 A Composite_assembly_table (see 4.2.23) shall allow for the placement of two or more instances of the same constituent_part on the same sequence or layer. A Ply_laminate_table shall allow for the placement of only one instance of the same constituent_part (only plies) on the same sequence or layer.

4.2.114 Part_version

A Part_version is the identification of the representation for a Part (see 4.2.112) before, during, or after its design has undergone a formal release or change. Only changes that are formally tracked by the organization that is responsible for the definition of the part shall be identified by a Part_version. Iterations on designs that are not formally tracked by the organization responsible for the definition of the part shall not be identified by a Part_version but should be tracked by a Design_discipline_product_-definition (see 4.2.36). The data associated with a Part_version are the following:

— approved_by;

— contract_number;

— creator;

— make_or_buy_code;

— part_number;

— release_status;

— revision_letter;

— security_code;

— weight.

4.2.114.1 approved_by

The approved_by specifies the approval of the Part_version. The approved_by need not be specified for a particular Part_version. See 4.3.142 for the application assertion.

4.2.114.2 contract_number

The contract_number specifies the business contract under which the Part is designed. A design may or may not be developed under a contract: however, if the design is developed under a contract, the attribute shall specify the unique identification of that contract. The contract_number need not be specified for particular Part_version.

4.2.114.3 creator

The creator specifies the Person_organization (see 4.2.118) who originated the Part_version. See 4.3.145 for the application assertion.

4.2.114.4 make_or_buy_code

The make_or_buy_code specifies the design organization's plan for obtaining the Part. It shall always be one of two values: make or buy. If the value of the attribute is make, the design organization has plans to make the Part internally. If the value of the attribute is buy, the design organization plans for the Part to be purchased from a vendor. The make_or_buy_code need not be specified for a particular Part_-version.

4.2.114.5 part_number

The part_number specifies the Part that Part_version represents. See 4.3.144 for the application assertion.

4.2.114.6 release_status

The release_status specifies the status of the version of the Part with respect to the dissemination of design information. It shall always be one of two values: released or unreleased. Released are those versions that have been reviewed and approved for further use. Unreleased are those versions that have not been reviewed or approved for further use.

4.2.114.7 revision_letter

The revision_letter specifies the unique identification of the Part_version. Since Part_version is also used to manage pre-release engineering changes, the release_status attribute should always have the value of ‘unreleased’ during the initial design or analysis phase until the official release of the product information.

4.2.114.8 security_code

The security_code specifies the security classification of the Part_version.

4.2.114.9 weight

The weight specifies the mass of the Part_version. The weight need not be specified for a particular Part_version. See 4.3.143 for the application assertion.

4.2.115 Percentage

A Percentage is the portion of a whole expressed in hundredths.

4.2.116 Percentage_laminate_table

A Percentage_laminate_table a type of Zone_structural_makeup (see 4.2.176) that defines the percentage of the total thickness within a zone or point that each uniquely oriented constituent stock material makes up.

NOTE For a simple laminate just made out of plies (Ply_laminate), a Percentage_laminate_table is used to capture the percentages of different fiber orientations within the zone based on the overall thickness of the zone. An example would be 40% 0 degree fibers, 30% 90 degree fibers, 15% +45 degree fibers, and 15% -45 degree fibers. The sum of the percents (%) have to equal 100.

Informal propositions:

IP: The sum of percentage_ply.volume_percent must equal 100.

The data associated with a Percentage_laminate_table are the following:

— table_components;

— total_thickness.

4.2.116.1 table_components

The table_components specifies the set of Percentage_ply (see 4.2.117) that defines the makeup of the zone. See 4.3.147 for the application assertion.

4.2.116.2 total_thickness

The total_thickness specifies the thickness or depth of the part at the area defined by the Percentage_-laminate_table. The value can be derived from a corresponding Thickness_laminate_table (see 4.2.168) or Part_laminate_table (see 4.2.113). If either exists for the zone of the part, the values must be consistent. See 4.3.146 for the application assertion.

4.2.117 Percentage_ply

A Percentage_ply is an idealization of a ply that represents a portion (percentage) of the thickness of a Percentage_laminate_table (see 4.2.116) with a particular fiber orientation (if required) and material type. This represents a way to relate anisotropic material properties to the varied orientations that exist in a composite part. A Percentage_ply always corresponds to the area within the zone of the laminate_table of which it is a portion. The data associated with a Percentage_ply are the following:

— makeup_and_properties;

— material;

— strength_orientation;

— volume_percent.

4.2.117.1 makeup_and_properties

The makeup_and_properties specifies the structural makeup of the zone represented by the Percentage_-ply. A Percentage_laminate_table shall not be used as makeup_and_properties for a Percentage_ply. The makeup_and_properties need not be specified for a particular Percentage_ply. See 4.3.151 for the application assertion.

4.2.117.2 material

The material specifies the stock_material (see 4.2.154) for the zone represented by the Percentage_ply. See 4.3.150 for the application assertion.

4.2.117.3 strength_orientation

The strength_orientation specifies the angle giving the material direction (warp direction of a fabric) in a local coordinate system (Reinforcement_orientation_basis - see 4.2.138). The strength_orientation need not be specified for a particular Percentage_ply. See 4.3.149 for the application assertion.

4.2.117.4 volume_percent

The volume_percent specifies the amount of volume the Percentage_ply occupies relative to the total volume of the Percentage_laminate_table (see 4.2.116). See 4.3.148 for the application assertion.

4.2.118 Person_organization

A Person_organization is the identification of a particular individual and/or organization. The data associated with a Person_organization are the following:

— address;

— organization;

— person;

— person_organization_identification.

4.2.118.1 address

The address specifies the routing for paper and electronic mail, or a physical location for the Person_-organization.

4.2.118.2 organization

The organization specifies the collection of people grouped together for one or more common purposes.

4.2.118.3 person

The person specifies an individual.

4.2.118.4 person_organization_identification

The person_organization_identification specifies the unique identifier for the Person_organization.

4.2.119 Ply

A Ply is a type of Constituent_part (see 4.2.25) that defines one of the layers of constant thickness material that make up a composite part. A Ply is a contiguous structure with a defined boundary that is made of one or more Ply_piece (see 4.2.124) objects (see Figure 8).

NOTE 1 A Ply may be a sheet pre-impregnated with resin material or a single-pass in a filament winding. Ply pieces that comprise a Ply must be of the same constant thickness material and fiber orientation (if fiber orientation is required). A Ply can also be the application of an isotropic material of constant thickness to an area. A Ply may not necessarily be a physical piece of material but an idealization of material in a zone that shares common fiber orientation and boundary.

NOTE 2 The boundary of a Ply defines the edge of the material used to make the Ply, and includes an outer boundary and, if required, inner boundaries.

The data associated with a Ply are the following:

— constituents;

— material_type;

— material_orientation;

— ply_thickness;

— shape.

4.2.119.1 constituents

The constituents specifies the list of Ply_piece objects that make up the Ply. The constituents need not be specified for a particular Ply. See 4.3.157 for the application assertion.

4.2.119.2 material_orientation

The material_orientation specifies the direction of the reinforcement fibers within a Ply. For materials with multiple orientations of fibers, orientation shall correspond to the warp direction of the material. Warp identifies the highest load carrying direction of the material. See 4.3.156 for the application assertion.

4.2.119.3 material_type

The material_type specifies the Stock_material (see 4.2.154) that the Ply is made of. Stock_material of constant thickness or those applied as a constant thickness are valid materials. The material_type shall be one of the following: Discontinuous_fiber_assembly (see 4.2.41), Filament_assembly (see 4.2.67), Isotropic_material (see 4.2.79). See 4.3.152, 4.3.153 and 4.3.154 for the application assertions.

4.2.119.4 ply_thickness

The ply_thickness specifies the thickness that the Ply will contribute to part thickness after curing. The ply_thickness is constant over the whole area of the Ply. The shape need not be specified for a particular Ply. The shape need not be specified for a particular Ply. See 4.3.155 for the application assertion.

NOTE The value can not always be derived from the Stock_material that makes up the Ply. Applying formable material like potting compound over an area is an example where a uniform thickness is required.

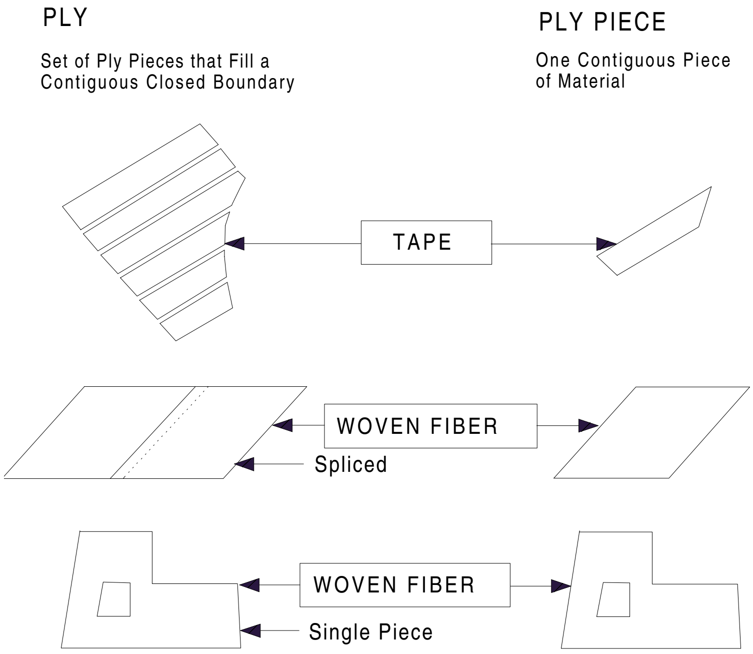

Figure 8 - Ply

4.2.119.5 shape

The shape specifies the geometric representation for the Ply. The shape need not be specified for a particular Ply. See 4.3.158 for the application assertion.

4.2.120 Ply_laminate

A Ply_laminate is a type of Constituent_part (see 4.2.25) that is formed by two or more Ply (see 4.2.119) objects that mate with each other. The plies have unique orientation and shape within the Ply_laminate.

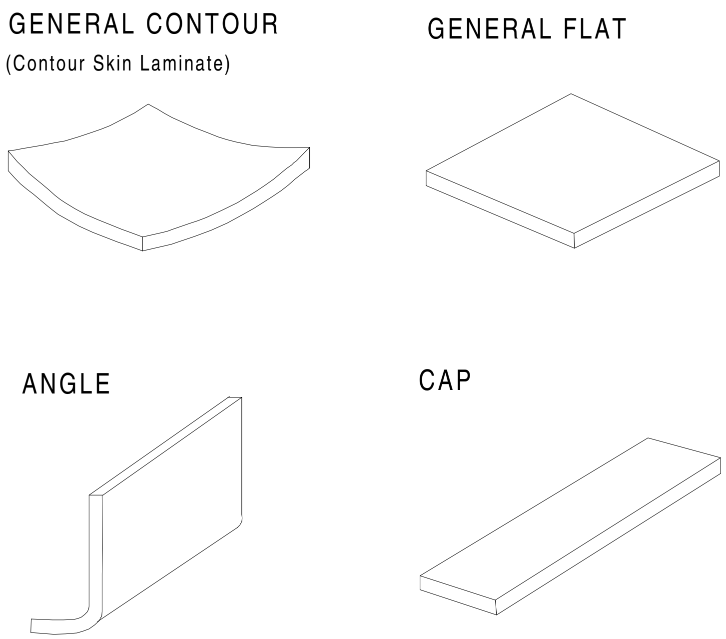

Figure 9 - Ply_laminate

NOTE Examples of Ply_laminate shape include angle, cap, flat, general contour, simple contour, complex contour, channel, etc. (see Figure 9).

The data associated with a Ply_laminate are the following:

— ply_table;

— shape.

4.2.120.1 ply_table

The ply_table specifies the ordered list of the Ply sets that make up the Ply_laminate. See 4.3.160 for the application assertion.

4.2.120.2 shape

The shape specifies the overall geometric representation of the Ply_laminate and/or shape_aspects of the Ply_laminate such as the edge of the Ply_laminate and any internal cutouts. The shape shall be one of the following: Composite_sheet_representation (see 4.2.24), Three_d_geometry_set (see 4.2.169). The shape need not be specified for a particular Ply_laminate. See 4.3.159 and 4.3.161 for the application assertions.

4.2.121 Ply_laminate_sequence_definition

A Ply_laminate_sequence_definition depicts a single unique sequence or layer within a Ply_laminate (see 4.2.120). The sequence or layer contains one or more plies.

NOTE When more than one ply resides in the same sequence or layer, the plies must not overlap. If plies do overlap they must be put on separate sequence or layers.

The data associated with a Ply_laminate_sequence_definition are the following:

— plies_in_sequence;

— properties.

4.2.121.1 plies_in_sequence

The plies_in_sequence specifies the set of Ply (see 4.2.119) objects that are laminated at the sequence. See 4.3.163 for the application assertion.

4.2.121.2 properties

The properties specify the Fea_material_property (see 4.2.63) that defines the physical response of the sequence. The properties need not be specified for a particular Ply_laminate_sequence_definition. See 4.3.162 for the application assertion.

4.2.122 Ply_laminate_table

A Ply_laminate_table is a type of Part_laminate_table (see 4.2.113) that provides an ordered list of Ply (see 4.2.119) objects. The ordered list provides basic placement position within the laminate (see Figure 10).

NOTE 1 The valid components of a Ply_laminate_table are plies only.

NOTE 2 The Ply_laminate_table is only valid to describe a Ply_laminate (see 4.2.120).

The data associated with a Ply_laminate_table are the following:

— sequence.

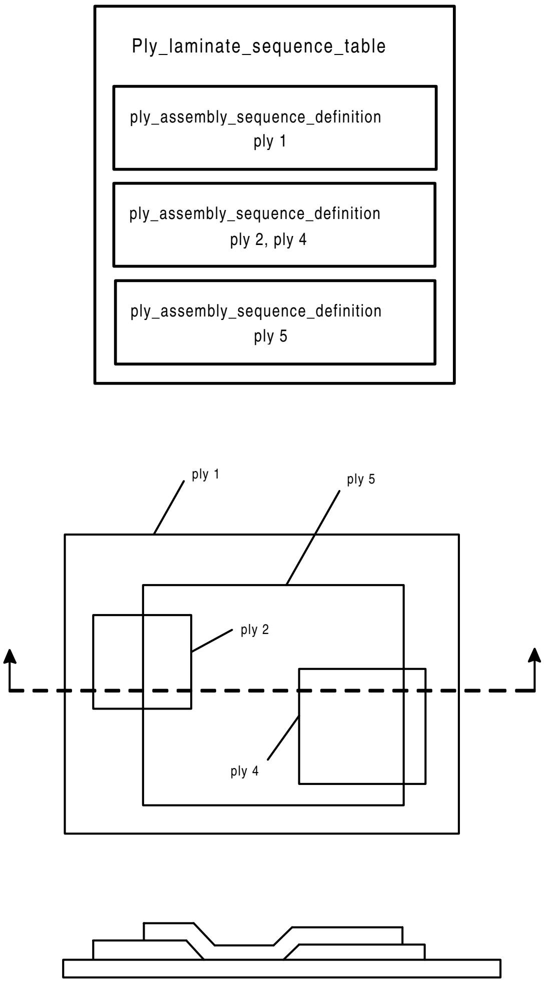

Figure 10 - Ply_laminate_table

4.2.122.1 sequence

The sequence specifies an ordered list of Ply sets that make up the Ply_laminate. See 4.3.164 for the application assertion.

NOTE A particular sequence can be for one or several ply(s) that make up the set.

4.2.123 Ply_orientation_angle

A Ply_orientation_angle defines a particular direction within a Ply (see 4.2.119). The direction is defined in the coordinate axis of a Reinforcement_orientation_basis (see 4.2.138). The measurement is in degrees between -90 and +90.

NOTE 1 For materials with multiple fiber orientations, the direction shall correspond to the warp direction of the material. Warp fibers are the primary load carrying fibers within a fabric material.

NOTE 2 In the design of a particular composite part, a finite set of Ply_orientation_angle values are established for use. An example of a set of commonly used Ply_orientation_angles is (0, 90, +45, -45).

A Ply_orientation_angle shall be one of the following: Draped_orientation_angle (see 4.2.42), Laid_-orientation_angle (see 4.2.80). The data associated with a Ply_orientation_angle are the following:

— angle_reference;

— basis.

4.2.123.1 angle_reference

The angle_reference specifies the curve, direction, or a point_path to serve that serves as a reference for ply orientation. The angle_reference shall be one of the following: Curve (see 4.2.28), Direction (see 4.2.39), Point_path (see 4.2.130). See 4.3.165, 4.3.166 and 4.3.167 for the application assertions.

4.2.123.2 basis

The basis specifies the Reinforcement_orientation_basis (see 4.2.138) used to define the direction. See 4.3.168 for the application assertion.

4.2.124 Ply_piece

A Ply_piece is a type of Constituent_part (see 4.2.25). It is a single portion of a Ply (see 4.2.119) that may be combined with other ply pieces on the same layer to make up the Ply. A Ply_piece has the same material and orientation direction as defined for the Ply of which it is a portion.

NOTE Stock_material may be too small to fully cover the area of a Ply. In this case, the application may allow two or more pieces to be used.

EXAMPLE Tape strips are a typical example of Ply_pieces that are grouped together to fill a Ply_shape.

The data associated with a Ply_piece are the following:

— shape;

— warp_surface.

4.2.124.1 shape

The shape specifies the geometric representation of the Ply_piece. The shape shall be a Ply_shape (see 4.2.125). The shape need not be specified for a particular Ply_piece. See 4.3.169 for the application assertion.

4.2.124.2 warp_surface

The warp_surface specifies whether or not the material warp surface is away from the basis surface, when the value of this attribute is set to true or false respectively. Material warp surface is the side of the material where the majority of the fibers are located.

4.2.125 Ply_shape

A Ply_shape is the association of a geometric representation to define the shape of a Ply (see 4.2.119).

NOTE Each type of Ply_shape can be generated from, or related to, another Ply_shape. For example, a Flat_-pattern_ply_shape (see 4.2.71) can be generated from a Laid_ply_shape (see 4.2.81); and a View_ply_shape (see 4.2.171) can be projected to the surface and create a Surface_ply_shape (see 4.2.163).

A Ply_shape shall be one of the following: Flat_Pattern_ply_shape (see 4.2.71), Laid_ply_shape (see 4.2.81), Projected_ply_shape (see 4.2.136). The data associated with a Ply_shape are the following:

— basis;

— defining_model.

4.2.125.1 basis

The basis specifies the shape representation that the Ply_shape is derived from or that is used as its basis of definition. The basis need not be specified for a particular Ply_shape. See 4.3.171 for the application assertion.

4.2.125.2 defining_model

The defining_model specifies the geometric representation used for the Ply_shape. See 4.3.170 for the application assertion.

4.2.126 Point

A Point is a specified spatial position in a coordinate space.

4.2.127 Point_and_vector

A Point_and_vector is a position in space combined with two directions. The data associated with a Point_and_vector are the following:

— location;

— vector.

4.2.127.1 location

The location specifies the position in space associated with a Point_and_vector. See 4.3.173 for the application assertion.

4.2.127.2 vector

The vector specifies the major and minor major fiber orientations, that are not necessarily orthogonal, associated with a Point_and_vector. See 4.3.172 for the application assertion.

4.2.128 Point_element

A Point_element is a type of Element (see 4.2.45) that represents a single node explicit element. The data associated with a Point_element are the following:

— associated_matrix.

4.2.128.1 associated_matrix

The associated_matrix specifies the stiffness, mass, or damping matrix associated with the element. See 4.3.174 for the application assertion.

4.2.129 Point_model

A Point_model is a type of Geometric_model_representation (see 4.2.72) that is composed of either points or vertices that are the defining points for the nodes of an Fea_model (see 4.2.64).

4.2.130 Point_path

A Point_path is a list of points and directions specifying the fiber orientations for a Ply (see 4.2.119). The data associated with a Point_path are the following:

— directions.

4.2.130.1 directions

The directions specifies the ordered list of fiber orientations for the Ply. See 4.3.175 for the application assertion.

4.2.131 Point_zone_shape

A Point_zone_shape is a type Zone_structural_makeup_shape_representation (see 4.2.177) that specifies a point on the surface of a part. It also specifies the singular volume a piercing line would make as it traveled through the part, normal to the part's surface at the point. The data associated with a Point_-zone_shape are the following:

— location.

4.2.131.1 location

The location specifies the point that defines the position of the local pierced point of the Zone_-structural_makeup (see 4.2.176). See 4.3.176 for the application assertion.

4.2.132 Process_specification

A Process_specification is a type of Specification (see 4.2.146) that defines a process or service to be performed on a product or material.

EXAMPLE Typical processes are heat treatment, welding, plating, packing, and marking.

4.2.133 Processed_core

A Processed_core is a type of Constituent_part (see 4.2.25) that is the central component of a sandwich construction to which faces or skins are attached. The primary structural purpose of Processed_core is the transfer of shear loads between the faces or skins. Processed_core is Stock_core (see 4.2.153) that has been machined, formed, stabilized and/or bonded together (see Figure 11). A core may have a potting compound or adhesive applied to it to make it more rigid and/or provide solid attachment points. The data associated with a Processed_core are the following:

— added_material;

— cell_direction;

— made_from_processed_core;

— made_from_stock;

— shape.

4.2.133.1 added_material

The added_material specifies the list of ply material such as stabilizer, potting compound, and adhesive, applied to a Stock_core (see 4.2.153) or Processed_core. The Ply (see 4.2.119) identifies the area where the added_material is applied. The added_material need not be specified for a particular Processed_core. See 4.3.180 for the application assertion.

4.2.133.2 cell_direction

The cell_direction identifies the ribbon direction of the honeycomb core. See 4.3.181 for the application assertion.

4.2.133.3 made_from_processed_core

The made_from_processed_core specifies a Processed_core that has another process performed on it. This is two or more Processed_core objects that are being assembled. The made_from_processed_core need not be specified for a particular Processed_core. See 4.3.182 for the application assertion.

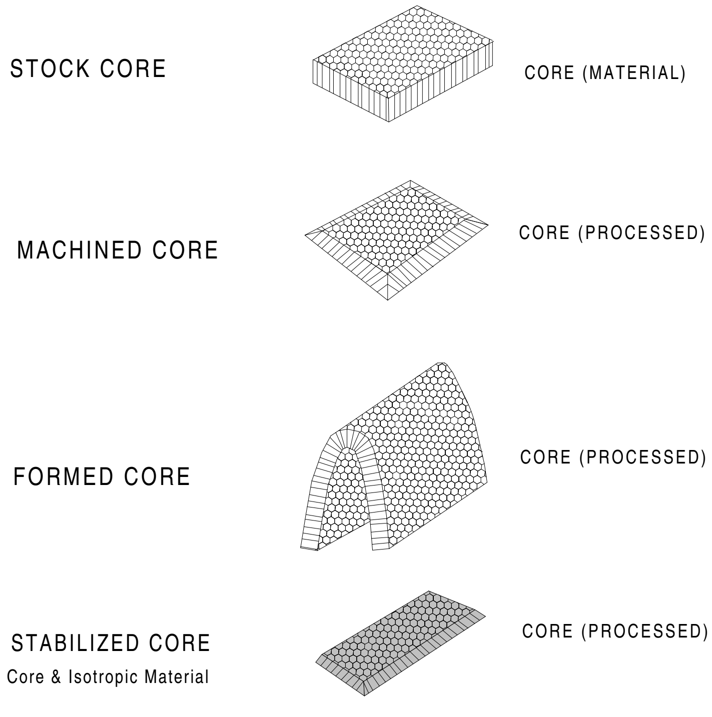

Figure 11 - Processed_core

4.2.133.4 made_from_stock

The made_from_stock specifies the Stock_core that is processed to get the Processed_core configuration. The types of processes include machining, forming, stabilizing, and bonding. See 4.3.183 for the application assertion.

4.2.133.5 shape

The shape specifies the geometric representation for the Stock_core. The shape shall be one of the following: Advanced_boundary_representation (see 4.2.2), Beveled_sheet_representation (see 4.2.15), Faceted_boundary_representation (see 4.2.55). The shape need not be specified for a particular Processed_core. See 4.3.177, 4.3.178 and 4.3.179 for the application assertions.

4.2.134 Product_configuration

A Product_configuration is a variation of a Product_model (see 4.2.135). Configuration management is based on this application object.

EXAMPLE 1 The organization has defined four configurations of the F-16: A, B, C and D. A Product_-configuration for the F-16 military aircraft is the D configuration of the F-16. In this instance, D is the configuration. A single product model might have many different configurations in order for the organization to identify variations of the product.

EXAMPLE 2 Product_configurations for the Camry automobile may be DS and LE, each defining different variations of the Camry product_model.

The data associated with a Product_configuration are the following:

— approved_by;

— item_identification;

— model_name;

— parts_configured;

— phase_of_product;

4.2.134.1 approved_by

The approved_by specifies the approval of the Product_configuration. The approved_by need not be specified for a particular Product_configuration. See 4.3.184 for the application assertion.

4.2.134.2 item_identification

The item_identification specifies the unique identification of a variation of the Product_model.

4.2.134.3 model_name

The model_name specifies the Product_model that the configuration is a part of. See 4.3.186 for the application assertion.

4.2.134.4 parts_configured

The parts_configured specifies the set of Part (see 4.2.112) objects that are being configured. The parts_-configured need not be specified for a particular Product_configuration. See 4.3.185 for the application assertion.

4.2.134.5 phase_of_product

The phase_of_product specifies the stage in the life cycle of the product in which a particular version of the design is planned to be produced.

EXAMPLE Within an organization, four phases of a product may be Technology and Concept Development, Development and Refinement, Manufacturing and Assembly Validation, and Production.

4.2.135 Product_model

A Product_model is the product the organization provides to its customers. The Product_model is identified for planning purposes in the design stage of a product.

EXAMPLE 1 F-22 is the identification of a product_model that is a military aircraft. The company that produces the aircraft uses F-22 to identify the actual deliverable product that it delivers to its customers.

EXAMPLE 2 Another example of a product_model is the Corvette. The name Corvette is used to identify an automobile that is delivered to customers.

The data associated with a Product_model are the following:

— model_name.

4.2.135.1 model_name

The model_name specifies the unique identification assigned by an organization to a product.

4.2.136 Projected_ply_shape

A Projected_ply_shape is a type of Ply_shape (see 4.2.125) that has its shape modeled by a projection on a surface or a plane. The surface has some defined context for the Ply (see 4.2.119).

NOTE A Projected_ply_shape is some projection of the actual ply. Projected shapes do not represent the build-up of the surface (Laid_ply_shape - see 4.2.81) but all lie on the same surface or plane. These shapes are defined during ply layout in design.

A Projected_ply_shape shall be one of the following: Surface_ply_shape (see 4.2.163), View_ply_shape (see 4.2.171). The data associated with a Projected_ply_shape are the following:

— method;

— projection_direction.

4.2.136.1 method

The method identifies the reference direction used in projecting the shape on the surface (see Figure 12). The value of method shall be one of the following:

— projection_reference;

— surface_normal.

See 4.2.136.1.1 and 4.2.136.1.2 for each allowable value for method.

4.2.136.1.1 projection_reference

projection_reference: to obtain the Laid_ply_shape (see 4.2.81), each ply added to the layup is updated in the projection_reference direction by a distance equal to the ply thickness divided by the cosine of theta. Theta is the angle between the normal vector on the surface and the projection direction. The method is typical of the way the View_ply_shape (see 4.2.171) objects are built, but it could also apply to Surface_ply_shape (see 4.2.163) objects.

4.2.136.1.2 surface_normal

surface_normal: to obtain the Laid_ply_shape, the ply edges are projected in the direction normal to the surface by a distance equal to the ply thickness.

4.2.136.2 projection_direction

The projection_direction defines the direction from the given representation to the actual Laid_ply_-shape. The projection_direction need not be specified for a particular Projected_ply_shape. See 4.3.187 for the application assertion.

4.2.137 Promissory_usage

A Promissory_usage is a type of Assembly (see 4.2.14) that associates a part to a higher level assembly when the next higher level assembly has not been defined.

4.2.138 Reinforcement_orientation_basis

A Reinforcement_orientation_basis is the definition of direction defined in terms of a Ply_laminate (see 4.2.120) or Composite_assembly (see 4.2.21). It is the basis for interpretation of the fiber orientation within a ply. The data associated with a Reinforcement_orientation_basis are the following:

— basis_location;

— orientation.

4.2.138.1 basis_location

The basis_location specifies the location of the orientation axis within the Ply_laminate. The location shall be on the base_surface for the Laminate_table (see 4.2.79). See 4.3.189 for the application assertion.

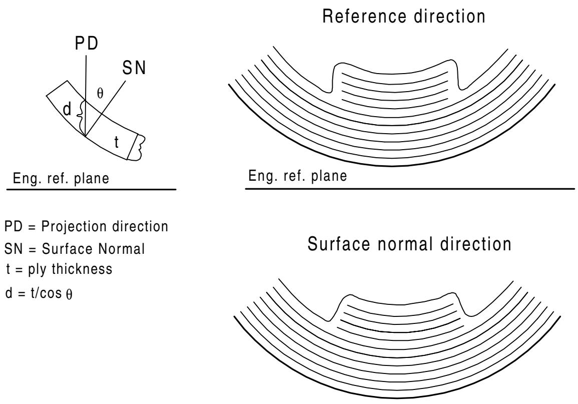

Figure 12 - Projection_method

4.2.138.2 orientation

The orientation specifies the direction for the basis. Positive placement x-axis is the fiber 0-degree direction, and the positive y-axis is the 90-degree direction. Angles are positive, counter-clockwise, in the xy-plane (in accordance to the right hand rule). See 4.3.188 for the application assertion.

4.2.139 Retention_period

A Retention_period is the definition of a period of time that product data needs to be maintained due to organizational policy or legal requirements. The data associated with a Retention_period are the following:

— approved_by;

— earliest_end_definition;

— is_applied_to;

— latest_end_definition;

— retention_purpose;

— start_definition.

4.2.139.1 approved_by

The approved_by specifies the approval of the Retention_period. The approved_by need not be specified for a particular Retention_period. See 4.3.195 for the application assertion.

4.2.139.2 earliest_end_definition

The earliest_end_definition specifies the time at which any object the Retention_period is applied to may be deleted. In this context, deletion applies to all subordinate objects that are not referenced by other objects. See 4.3.197 for the application assertion.

4.2.139.3 is_applied_to

The is_applied_to specifies the set of objects whose existence is controlled by the Retention_period. The is_applied_to shall be a set of one of the following: Additional_design_information (see 4.2.1), Analysis (see 4.2.4), Analysis_discipline_product_definition (see 4.2.6), Analysis_report_representation (see 4.2.8), Analysis_version (see 4.2.10), Assembly (see 4.2.14), Design_discipline_product_definition (see 4.2.35.2), Design_material (see 4.2.37), Fe_analysis (see 4.2.56), Fe_analysis_results (see 4.2.58), Fea_-model (see 4.2.65), Material_property (see 4.2.95), Material_specification (see 4.2.96), Part (see 4.2.112), Part_version (see 4.2.114). See 4.3.190, 4.3.191, 4.3.192, 4.3.193, 4.3.194, 4.3.196, 4.3.198, 4.3.199, 4.3.200, 4.3.201, 4.3.202, 4.3.203, 4.3.204, 4.3.205 and 4.3.206 for the application assertions.

4.2.139.4 latest_end_definition

The latest_end_definition specifies the time at which any object the Retention_period is applied to shall be deleted. In this context, deletion applies to all subordinate objects that are not referenced by other objects. See 4.3.197 for the application assertion.

4.2.139.5 retention_purpose

The retention_pupose specifies the rationale behind the Retention_period. The retention_purpose need not be specified for a particular Retention_period.

4.2.139.6 start_definition

The start_definition specifies the time when the Retention_period starts. See 4.3.197 for the application assertion.

4.2.140 Sequence_effectivity

A Sequence_effectivity is a type of Effectivity (see 4.2.44). It is the specification of the intended use by a design organization of a part within a range of product configurations identified by planned serial numbers. The data associated with a Sequence_effectivity are the following:

— component_quantity;

— from_effectivity_identification;

— quantity_unit_of_measure;

— to_effectivity_identification.

4.2.140.1 component_quantity

The component_quantity specifies the number of units that are effective for a particular part in a configuration.

4.2.140.2 from_effectivity_identification

The from_effectivity_identification specifies the beginning serial number of the range for the Sequence_-effectivity.

4.2.140.3 quantity_unit_of_measure

The quantity_unit_of_measure specifies the measure in terms of which the component_quantity is expressed.

EXAMPLE Quantity_unit_of_measure may be rolls, sheets, and bars, among other things.

4.2.140.4 to_effectivity_identification

The to_effectivity_identification specifies the ending serial number of the range for the Sequence_-effectivity.

4.2.141 Shape

A Shape is the mathematical representation of the form of a part. A Shape shall be one of the following: Idealized_analysis_shape (see 4.2.78), Node_shape (see 4.2.107), Nominal_design_shape (see 4.2.109). The data associated with a Shape are the following:

— role.

4.2.141.1 role

The function that the Shape plays in the definition of a part or analysis model, such as mesh generation geometry.

4.2.142 Shape_aspect

A Shape_aspect is a distinct portion of a part. The shape of a part consists of one or more Shape_aspects.

NOTE Often different types of geometric models will define the same shape or shape aspect of a part. This is usually caused by the diverse set of applications that may be utilized in the design, analysis, and manufacturing of the part.

The data associated with a Shape_aspect are the following:

— characteristics;

— geometry;

— parent_shape.

4.2.142.1 characteristics

The characteristics specifies the set of Specification (see 4.2.146) objects which define qualities or features of the Shape_aspect. See 4.3.209 for the application assertion.

4.2.142.2 geometry

The geometry specifies the geometric representation for the aspect of the shape. See 4.3.207 for the application assertion.

4.2.142.3 parent_shape

The parent_shape specifies the Shape (see 4.2.141) that the Shape_aspect is a part of. See 4.3.208 for the application assertion.

4.2.143 Single_point_constraint

A Single_point_constraint is a type of Constraint (see 4.2.27) that constrains for one or more degrees of freedom at a single node. The values of the constraint are set by the Single_point_constraint_values object. The data associated with a Single_point_constraint are the following:

— required_node.

4.2.143.1 required_node

The required_node specifies where the Single_point_constraint is applied. The node shall be one of the following: Geometry_element (see 4.2.74), Group (see 4.2.76), Node (see 4.2.105). See 4.3.210, 4.3.211 and 4.3.212 for the application assertions.

4.2.144 Single_point_constraint_values

A Single_point_constraint_values is a type of Fe_analysis_state_definition (see 4.2.61) that defines the value of a Single_point_constraint (see 4.2.143). The data associated with a Single_point_constraint_-values are the following:

— element.

4.2.144.1 element

The element specifies the Single_point_constraint to which the coefficient is applied. See 4.3.213 for the application assertion.

4.2.145 Smeared_material

A Smeared_material is a type of Zone_structural_makeup (see 4.2.176) that provides the material properties for an idealized zone or area of a part.

NOTE A Smeared_material can represent the material properties of a single Stock_material (see 4.2.154) by having the Percentage_laminate_table (see 4.2.116) reference one stock_material and set its percent value of that material's constituent to 100%.

The data associated with a Smeared_material are the following:

— total_thickness.

4.2.145.1 total_thickness

The total_thickness specifies the thickness or depth of the zone defined by the Smeared_material. See 4.3.214 for the application assertion.

4.2.146 Specification

A Specification is a document that contains definitions, processes, or rules related to a unique quality that a finished or in-process part shall possess. International and organizational standards are examples of specifications. A Specification shall be one of the following: Design_specification (see 4.2.38), Material_specification (see 4.2.96), Process_specification (see 4.2.132), Surface_finish_specification (see 4.2.162).

The data associated with a Specification are the following:

— specification_code;

— specification_source.

4.2.146.1 specification_code

The specification_code specifies the number or code that identifies the Specification.

4.2.146.2 specification_source

The specification_source specifies the organization that is responsible for the Specification.

4.2.147 Specified_state

A Specified_state is a type of Fe_analysis_state (see 4.2.60) representing information that is specified. None of the information in it is a result of a previous calculation.

4.2.148 Start_order

A Start_order is a type of Work_order (see 4.2.174) that authorizes work in the development of the initial design or analysis of a Part (see 4.2.112), resulting in the creation of an original Part_version (see 4.2.114) or Analysis_version (see 4.2.10).

4.2.149 Start_request

A Start_request is a type of Work_request (see 4.2.175) that identifies the work to be done to initiate the design or analysis of a Part (see 4.2.112).

4.2.150 Static_control_step

A Static_control_step is a type of Fe_analysis_control_step (see 4.2.57) that defines a single step in a static Fea_analysis (see 4.2.56). The data associated with a Static_control_step are the following:

— final_input_state.

4.2.150.1 final_input_state

The final_input_state specifies the final equilibrium state for the Static_control_step. See 4.3.215 for the application assertion.

4.2.151 Static_results_step

A Static_results_step is a type of Fe_analysis_results_step (see 4.2.59) that defines the state of node and element variables at the end of a static analysis step. The data associated with a Static_results_step are the following:

— resulting_state.

4.2.151.1 resulting_state

The resulting_state specifies the final state of the node and element variables for the step. See 4.3.216 for the application assertion.

EXAMPLE Node variables include such information as displacements, rotations, and reaction forces. Element variables include such information as stresses and strains.

4.2.152 Stiffness_matrix

A Stiffness_matrix is a type of Matrix (see 4.2.97) that is an explicit representation of stiffness at a node, between two nodes in a direction, or for many nodes.

4.2.153 Stock_core

A Stock_core is a type of Stock_material (see 4.2.154) that is processed to be placed within a laminate and play a role in enhancing the stiffness of the laminate.

EXAMPLE Honeycomb, phenolic block, and corrugation are types of stock_core.

4.2.154 Stock_material

A Stock_material is the material typically procured from a material supplier to fabricate a Part (see 4.2.112).

NOTE The Stock_material in this Part of 10303 is limited to the material types that physically become all or a portion of the finished part.

A Stock_material shall be one of the following: Anisotropic_material (see 4.2.12), Discontinuous_fiber_-assembly (see 4.2.41), Filament_assembly (see 4.2.67), Isotropic_material (see 4.2.79), Stock_core (see 4.2.153). The data associated with a Stock_material are the following:

— approved_by;

— of_part;

— property;

— reference_direction;

— specified_material.

4.2.154.1 approved_by

The approved_by specifies the approval of the Stock_material. The approved_by need not be specified for a particular Stock_material. See 4.3.217 for the application assertion.

4.2.154.2 property

The property specifies the Material_property (see 4.2.95) that defines the behavior of the Stock_material to an external response. See 4.3.219 for the application assertion.

4.2.154.3 reference_direction

The reference_direction specifies the direction of interest for the Stock_material such as cell direction, grain direction, etc. See 4.3.218 for the application assertion.

4.2.154.4 of_part

The of_part specifies the Part_version (see 4.2.114) representation for the Stock_material. See 4.3.221 for the application assertion.

4.2.154.5 specified_material

The specified_material identifies the specification that defines the material. See 4.3.220 for the application assertion.

4.2.155 Substitute_part

A Substitute_part is a component within an Assembly (see 4.2.14) whose form, fit, and function might be different from the component it replaces, but that can fulfill the requirements of the latter within the context of the assembly. The data associated with a Substitute_part are the following:

— base;

— substitute.

4.2.155.1 base

The base specifies the assembly-component relationship in which the substitute may be used. See 4.3.222 for the application assertion.

4.2.155.2 substitute

The substitute specifies the Part (see 4.2.112) which may be used in place of the base. See 4.3.223 for the application assertion.

4.2.156 Substructure_element

A Substructure_element is a type of Element (see 4.2.45) that contains a finite element model that may be assembled into another finite element model. The data associated with a Substructure_element are the following:

— substructure_model_ref.

4.2.156.1 substructure_model_ref

The substructure_model_ref specifies the Fea_model (see 4.2.65) representing the substructure. See 4.3.224 for the application assertion.

4.2.157 Substructure_node_relationship

A Substructure_node_relationship is the relationship of a node in one substructure to a node in another substructure. The data associated with a Substructure_node_relationship are the following:

— related_node;

— relating_node.

4.2.157.1 related_node

The related_node specifies the node to be associated to a node in another substructure. See 4.3.225 for the application assertion.

4.2.157.2 relating_node

The relating_node specifies the node that has a node related to it from another substructure. See 4.3.225 for the application assertion.

4.2.158 Supplied_part_version

A Supplied_part_version is a part that is defined by its part number and Supplier (see 4.2.159). The data associated with a Supplied_part_version are the following:

— approved_by;

— certification_required;

— is_identified_as;

— produced_by;

— supplier_part_number.

4.2.158.1 approved_by

The approved_by specifies the approval of the Supplied_part_version. The approved_by need not be specified for a particular Supplied_part_version. See 4.3.226 for the application assertion.

4.2.158.2 certification_required

The certification_required specifies whether or not a Supplier must be approved prior to the procurement of the part from that Supplier.

4.2.158.3 is_identified_as

The is_identified_as specifies the Part_version (see 4.2.114) that the part is used for. See 4.3.227 for the application assertion.

4.2.158.4 produced_by

The produced_by specifies the organization that supplies the part. See 4.3.228 for the application assertion.

4.2.158.5 supplier_part_number

The supplier_part_number specifies the part number used by the organization that produces the part. The supplier_part_number need not be specified for a particular Supplier_part_version.

4.2.159 Supplier

A Supplier identifies an organization that designs or produces a part. The data associated with a Supplier are the following:

— identified_as;

— supplier_identification.

4.2.159.1 identified_as

The identified_as specifies the organization that designs or manufactures a part. See 4.3.229 for the application assertion.

4.2.159.2 supplier_identification

The supplier_identification specifies the unique identifier for the supplier.

4.2.160 Surface

A Surface can be envisioned as a set of connected points in three-dimensional space which is always locally two-dimensional, but need not be a manifold.

4.2.161 Surface_element

A Surface_element is a type of Element (see 4.2.45) that models a surface with a specified section (thickness).

4.2.162 Surface_finish_specification

A Surface_finish_specification is a type of Specification (see 4.2.146) that defines properties applicable to surface textures or protective coatings for an in-process or completed part.

4.2.163 Surface_ply_shape

A Surface_ply_shape is a type of Projected_ply_shape (see 4.2.136) that is defined on a surface defining some aspect of a composite part (see Figure 13). The surface may be the layup surface, the outer mold line (OML), or the inner mold line (IML). The data associated with a Surface_ply_shape are the following:

— offset;

— surface_role.

4.2.163.1 offset

The offset specifies the distance from the modeled surface to the layup surface of the part. The offset distance would be zero if the modeled surface is the layup surface. See 4.3.230 for the application assertion.

4.2.163.2 surface_role

The surface_role specifies the context of the surface defined to the actual part and should be limited to text defining contexts such as 'layup surface', 'outer mold line', and 'inner mold line'.

4.2.164 Surface_property

A Surface_property is the material and section properties for a Surface_element (see 4.2.161). The data associated with a Surface_property are the following:

— defined_elements.

4.2.164.1 defined_elements

The defined_elements specifies the set of Surface_element (see 4.2.161) objects for which properties are being defined. See 4.3.231 for the application assertion.

4.2.165 Surface_thickness

A Surface_thickness is the thickness of a Surface_element (see 4.2.161). The data associated with a Surface_thickness are the following:

— property;

— thickness.

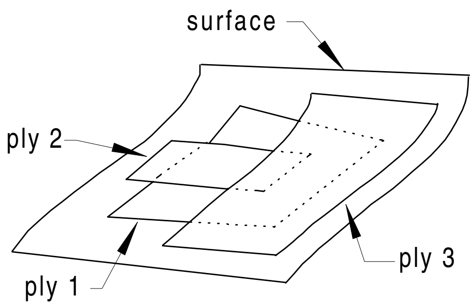

Figure 13 - Surface_ply_shape

4.2.165.1 property

The property specifies the Surface_property (see 4.2.164) for which a thickness is being defined. See 4.3.233 for the application assertion.

4.2.165.2 thickness

The thickness specifies the length value assigned as the thickness of a surface element. See 4.3.232 for the application assertion.

4.2.166 Surface_with_direction

A Surface_with_direction is the specification of a Surface (see 4.2.160) and a Direction (see 4.2.39). The Direction defines the side of the Surface that is the material side. The data associated with a Surface_-with_direction are the following:

— defining_surface;

— material_direction.

4.2.166.1 defining_surface

The defining_surface specifies the Surface that is used in the definition. See 4.3.235 for the application assertion.

4.2.166.2 material_direction

A material_direction specifies the side of the Surface that is the material side. See 4.3.234 for the application assertion.

4.2.167 Tabular_representation

A Tabular_representation is a type of Analysis_report_representation (see 4.2.8). It is a text documenting the analysis assumptions, the load cases for minimum and maximum margins of safety, the critical location and detail analysis to produce the margin of safety.

The analysis assumptions documented may include:

— Mesh discretization;

— Geometric property idealization;

— Material property idealization;

— Detail analysis;

— Information manipulation prior to graphical display;

— References for material properties, idealization methods; and

— Load and boundary condition idealization.

The critical location and margin of safety analyses may include the location and margin for:

— Composite or metallic joints;

— Fastener pull-through;

— Cutouts;

— Composite or metallic point stress;

— Stiffener buckling, crippling;

— Stiffener pulloff;

— Panel buckling, crippling, transverse loadings, flutter;

— Criterion for margin of safety; and

— Critical load condition(s) and environment.

A table of contents may be included. The data associated with a Tabular_representation are the following:

— tabular_filename.

4.2.167.1 tabular_filename

The tabular_filename specifies the name of the file containing the tabular information.

4.2.168 Thickness_laminate_table

A Thickness_laminate_table is a type of Zone_structural_makeup (see 4.2.176) that is used to capture the ordered list of constituents that make up a zone. The data associated with a Thickness_laminate_table are the following:

— table_components.

4.2.168.1 table_components

The table_components specifies the ordered list of constituents that make up the zone. The table_-components shall be a list of one or more of the following: Filament_laminate (see 4.2.68), Ply (see 4.2.119), Processed_core (see 4.2.133). See 4.3.236, 4.3.237 and 4.3.238 for the application assertions.

4.2.169 Three_d_geometry_set

A Three_d_geometry_set is a type of Constituent_shape_representation (see 4.2.26) that defines 3D boundary curve and surface geometry without topology. The surfaces used in the representation have a defined model surface context. The data associated with a Three_d_geometry_set are the following:

— basis_role;

— basis_surface;

— defining_boundary.

4.2.169.1 basis_role

The basis_role specifies the context of the basis surface for the part. 'layup surface', 'outer mold line', and 'inner mold line' are specific contexts.

4.2.169.2 basis_surface

The basis_surface specifies the surface used to define the part shape. The context of the surface is defined in the attribute basis_role. See 4.3.240 for the application assertion.

4.2.169.3 defining_boundary

The defining_boundary specifics the set of Curve (see 4.2.28) objects that define the edge aspect of a part. See 4.3.239 for the application assertion.

4.2.170 Usage_constraint

A Usage_constraint is a restriction of the application of a Specification (see 4.2.146). It associates a particular topic of a specification with specific information or text to be used for a particular application. The data associated with a Usage_constraint are the following:

— constrains;

— usage_element;

— usage_value.

4.2.170.1 constrains

The constrains identifies the Specification being restricted in its application. See 4.3.241 for the application assertion.

4.2.170.2 usage_element

The usage_element specifies the particular section or topic in a Specification that is being constrained or clarified.

4.2.170.3 usage_value

The usage_value specifies the limiting or particular data to be assigned to the particular section or topic in a Specification that is being constrained or clarified.

EXAMPLE A part may require painting per a company written surface finish specification. The population would be: specification_code = ABCD-1; specification_source = ACME Company; the paired Usage_constraint would be: usage_element = colour; usage_value = red; usage_element = coats; usage_value = 3.

4.2.171 View_ply_shape

A View_ply_shape is a type of Projected_ply_shape (see 4.2.136) in which the projection is to a plane such as a drawing view (see Figure 14). The projection direction establishes the relationship of the plane to the actual part.

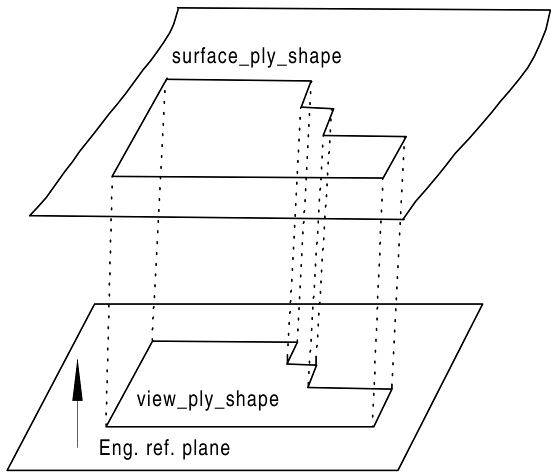

Figure 14 - View_ply_shape

4.2.172 Volume_element

A Volume_element is a type of Element (see 4.2.45) that models an elemental portion of continuum.

4.2.173 Wireframe_with_topology

A Wireframe_with_topology is a type of Geometric_model_representation (see 4.2.72) for prismatic parts that is defined by the curves representing the part edge and is topologically constrained.

4.2.174 Work_order

A Work_order is a document that authorizes work in the development of an initial or modified design or analysis of a part. It is the result of the processing of one or more Work_request (see 4.2.175) objects. A Work_order shall be one of the following: Change_order (see 4.2.19), Start_order (see 4.2.148). The data associated with a Work_order are the following:

— additional_data;

— analysis_data;

— approved_by;

— incorporates;

— versions;

— work_order_identification;

4.2.174.1 additional_data

The additional_data specifies additional pertinent information that was compiled during the development of the related set of proposed work requests. The additional_data need not be specified for a particular Work_order. See 4.3.247 for the application assertion.

4.2.174.2 analysis_data

The analysis_data specifies the results of an assessment that was performed to validate the design or modification in a related set of proposed work requests. The analysis_data need not be specified for a particular Work_order. See 4.3.247 for the application assertion.

4.2.174.3 approved_by

The approved_by specifies the approval of the Work_order. The approved_by need not be specified for a particular Work_order. See 4.3.244 for the application assertion.

4.2.174.4 incorporates

The incorporates specifies the set of Work_request objects that the Work_order incorporates. See 4.3.249 for the application assertion.

4.2.174.5 versions

The versions specifies the set of product definition or product version objects to which the Work_order applies. The versions shall be a set of one or more of the following: Analysis_discipline_product_-definition (see 4.2.6), Analysis_version (see 4.2.10), Assembly (see 4.2.14), Design_discipline_product_-definition (see 4.2.35.2), Part_version (see 4.2.114). See 4.3.242, 4.3.243, 4.3.245, 4.3.246 and 4.3.248 for the application assertions.

4.2.174.6 work_order_identification

The work_order_identification specifies a unique identifier for the Work_order.

4.2.175 Work_request

A Work_request is a document that identifies the work to be done to initiate or redefine the design or analysis of a part. A Work_request shall be one of the following: Change_request (see 4.2.19), Start_-request (see 4.2.149). The data associated with a Work_request are the following:

— affective_parts;

— approved_by;

— description;

— reason;

— recipients;

— request_date;

— status;

— work_request_identification.

4.2.175.1 affective_parts

The affective_parts specifies the set of product definition or product version objects to which the Work_-request applies. The affective_parts shall be a set of one or more of the following: Analysis_discipline_-product_definition (see 4.2.6), Analysis_version (see 4.2.10), Assembly (see 4.2.14), Design_discipline_-product_definition (see 4.2.35.2), Part_version (see 4.2.114). The affective_parts need not be specified for a Work_request that is a start request. See 4.3.250, 4.3.251, 4.3.253, 4.3.255, and 4.3.256 for the application assertions.

4.2.175.2 approved_by

The approved_by specifies the approval of the Work_request. The approved_by need not be specified for a particular Work_request. See 4.3.252 for the application assertion.

4.2.175.3 description

The description specifies an explanation of the work to be done to develop or redefine the design or analysis of a part.

4.2.175.4 reason

The reason specifies the purpose for generating the Work_request.

4.2.175.5 recipients

The recipients specifies the set of Person_organization (see 4.2.118) objects which receive the Work_-request. See 4.3.257 for the application assertion.

4.2.175.6 request_date

The request_date specifies the date when the Work_request was created. See 4.3.254 for the application assertion.

4.2.175.7 status

The status specifies the current level of completion of the Work_request. It shall always be one of four values: proposed, in_work, issued, and hold. If the value of the attribute is proposed**,** the Work_request has been completed and is awaiting review and authorization. If the value of the attribute is in_work**,** the Work_request is being developed for possible inclusion in the design or analysis. If the value of the attribute is issued, the Work_request has been authorized for inclusion in the design or analysis. If the value of the attribute is hold**,** the Work_request has been reviewed and not authorized to be included in the design or analysis.

4.2.175.8 work_request_identification

The work_request_identification specifies a unique identifier for the proposed work.

4.2.176 Zone_structural_makeup

A Zone_structural_makeup is a type of Laminate_table (see 4.2.82) that describes the structural makeup of a particular zone, area, or point within a laminate.

NOTE 1 Structural makeup information consists of: 1) the smear material properties based on one or more constituents that reside in that zone or point; 2) the percentage of the total thickness within a zone or point that each constituent type makes up; and 3) the ordered list of physical constituents based on position within a zone or point. These three pieces of structural makeup information can coexist in any order to describe a zone or point within a laminate.

NOTE 2 A single Zone_structural_makeup could apply to the whole part. The would be the case for a homogeneous structural part or a simple laminate of constant thickness and material constituents.

A Zone_structural_makeup shall be one of the following: Percentage_laminate_table (see 4.2.116), Smeared_material (see 4.2.145), Thickness_laminate_table (see 4.2.168), Percentage_laminate_table and Smeared_material, Thickness_laminate_table and Smeared_material. The data associated with a Zone_structural_makeup are the following:

— boundary.

4.2.176.1 boundary

The boundary specifies the shape of the Zone_structural_makeup as either a point, one or more unique list of curves, or a face. When the boundary is not defined for an instance of a Zone_structural_makeup, the Zone_structural_makeup applies to the whole part. The would be the case for a homogeneous structural part. See 4.3.258 for the application assertion.

4.2.177 Zone_structural_makeup_shape_representation

A Zone_structural_makeup_shape_representation is the shape of a zone or area of a part.

NOTE The zone can be a point, a specific area, or the entire part.

A Zone_structural_makeup_shape_representation shall be one of the following: Edge_zone_shape (see 4.2.43), Point_zone_shape (see 4.2.131).

4.3 Application assertions

This subclause specifies the application assertions for the Composite and Metallic Structural Analysis and Related Design application protocol. Application assertions specify all relationships among application objects, the cardinality of the relationships, and the rules required for the integrity and validity of the application objects and UoFs. The application assertions and their definitions are given below.

4.3.1 Additional_design_information to Design_discipline_product_definition

Additional_design_information has design defined by exactly one Design_discipline_product_definition. Design_discipline_product_definition defines design for zero, one, or many additional_design_-information objects.

4.3.2 Additional_design_information to Specification

Additional_design_information has additional_information defined by a set of one to many Specification objects. Specification defines additional_information for zero, one, or many Additional_design_-information objects.

4.3.3 Alternate_part to Part

Alternate_part has alternate defined by a set of one to many Part objects. Part defines alternate for zero, one, or many Alternate_part objects.

4.3.4 Analysis to Person_organization

Analysis has owner defined by exactly one Person_organization. Person_organization is the owner of zero, one, or many Analysis objects.

4.3.5 Analysis_design_version_relationship to Analysis_version

Analysis_design_version_relationship has analysis defined by exactly one Analysis_version. Analysis_-version defines analysis for zero, one, or many Analysis_design_version_relationship objects.

4.3.6 Analysis_design_version_relationship to Part_version

Analysis_design_version_relationship has design defined by exactly one Part_version. Part_version defines design for zero, one, or many Analysis_design_version_relationship objects.

4.3.7 Analysis_discipline_product_definition to Analysis_version

Analysis_discipline_product_definition has version defined by exactly one Analysis_version. Analysis_-version defines version for zero, one, or many Analysis_discipline_product_definition objects.

4.3.8 Analysis_discipline_product_definition to Approval

Analysis_discipline_product_definition is approved by zero or one Approval. Approval approves zero, one, or many Analysis_discipline_product_definition objects.

4.3.9 Analysis_discipline_product_definition to Date

Analysis_discipline_product_definition has creation_date defined by exactly one Date. Date defines creation_date for zero, one, or many Analysis_discipline_product_definition objects.

4.3.10 Analysis_discipline_product_definition to File

Analysis_discipline_product_definition has cae_filename defined by zero or one File. File defines cae_filename for zero, one, or many Analysis_discipline_product_definition objects.

4.3.11 Analysis_discipline_product_definition to Person_organization

Analysis_discipline_product_definition is created by exactly one Person_organization. Person_-organization defines creator for zero, one, or many Analysis_discipline_product_definition objects.

4.3.12 Analysis_report_representation to Fe_analysis

Analysis_report_representation has control defined by zero or one Fe_analysis. Fe_analysis defines control for zero, one, or many Analysis_report_representation objects.

4.3.13 Analysis_report_representation to Fe_analysis_results

Analysis_report_representation has result defined by zero or one Fe_analysis_results. Fe_analysis_results defines result for zero, one, or many Analysis_report_representation objects.

4.3.14 Analysis_report_representation to Fea_model

Analysis_report_representation has model_ref defined by exactly one Fea_model. Fea_model defines model_ref for zero, one, or many Analysis_report_representation objects.

4.3.15 Analysis_shape to Analysis_discipline_product_definition

Analysis_shape has analysis_view defined by exactly one Analysis_discipline_product_definition. Analysis_discipline_product_definition defines analysis_view for zero, one, or many Analysis_shape objects.

4.3.16 Analysis_version to Analysis

Analysis_version has analysis_number defined by exactly one Analysis. Analysis defines analysis_-number for zero, one, or many Analysis_version objects.

4.3.17 Analysis_version to Approval

Analysis_version is approved by zero or one Approval. Approval approves zero, one, or many Analysis_-version objects.

4.3.18 Analysis_version to Person_organization

Analysis_version is created by exactly one Person_organization. Person_organization defines creator for zero, one, or many Analysis_version objects.

4.3.19 Approval to Date

Approval has effective_date defined by exactly one Date. Date defines effective_date for zero, one, or many Approval objects.

4.3.20 Approval to Person_organization

Approval is authorized by one or more Person_organization objects. Person_organization authorizes zero, one, or many Approval objects.

4.3.21 Assembly to Design_discipline_product_definition

Assembly has assembly_part defined by exactly one Design_discipline_product_definition. Design_-discipline_product_definition defines assembly_part for zero, one, or many Assembly objects.

Assembly has component_part defined by exactly one Design_discipline_product_definition. Design_-discipline_product_definition defines component_part for zero, one, or many Assembly objects.

4.3.22 Beveled_sheet_representation to Angle_measure

Beveled_sheet_representation has bevel_angle defined by exactly one Angle_measure. Angle_measure defines bevel_angle for zero, one, or many Beveled_sheet_representation objects.

4.3.23 Beveled_sheet_representation to Composite_sheet_representation

Beveled_sheet_representation has boundary_surface defined by exactly one Composite_sheet_-representation. Composite_sheet_representation defines boundary_surface for zero, one, or many Beveled_sheet_representation objects.

4.3.24 Beveled_sheet_representation to Length_measure

Beveled_sheet_representation has vertical_profile_height defined by exactly one Length_measure. Length_measure defines vertical_profile_height for zero, one, or many Beveled_sheet_representation objects.

4.3.25 Boundary_curve_representation to Curve

Boundary_curve_representation has bounds defined by one or many Curve objects. Curve defines bounds for zero, one, or many Boundary_curve_representation objects.

4.3.26 Change_order to Date

Change_order has change_date defined by exactly one Date. Date defines change_date for zero, one, or many Change_order objects.

4.3.27 Component_assembly_position to Geometric_model_representation

Component_assembly_position has assembly_shape defined by exactly one Geometric_model_-representation. Geometric_model_representation defines assembly_shape for zero, one, or many Component_assembly_position objects.

Component_assembly_position has component_shape defined by exactly one Geometric_model_-representation. Geometric_model_representation defines component_shape for zero, one, or many component_assembly_position objects.

4.3.28 Component_assembly_position to Next_higher_assembly

Component_assembly_position references exactly one Next_higher_assembly in the role of definition. Next_higher_assembly serves as definition for zero, one, or many Component_assembly_position objects.

4.3.29 Composite_assembly to Advanced_boundary_representation

Composite_assembly has shape defined by exactly one Advanced_boundary_representation. Advanced_-boundary_representation defines shape for zero, one, or many Composite_assembly objects.

NOTE This assertion is established through composite_assembly_shape.

4.3.30 Composite_assembly to Composite_assembly_table

Composite_assembly has layup_part defined by exactly one Composite_assembly_table. Composite_-assembly_table defines layup_part for zero, one, or many Composite_assembly objects.

4.3.31 Composite_assembly to Faceted_boundary_representation

Composite_assembly has shape defined by exactly one Faceted_boundary_representation. Faceted_-boundary_representation defines shape for zero, one, or many Composite_assembly objects.

NOTE This assertion is established through composite_assembly_shape.

4.3.32 Composite_assembly to Manifold_surface_with_topology

Composite_assembly has shape defined by exactly one Manifold_surface_with_topology. Manifold_-surface_with_topology defines shape for zero, one, or many Composite_assembly objects.

NOTE This assertion is established through composite_assembly_shape.

4.3.33 Composite_assembly to Non_topological_surface_and_wireframe

Composite_assembly has shape defined by exactly one Non_topological_surface_and_wireframe. Non_-topological_surface_and_wireframe defines shape for zero, one, or many Composite_assembly objects.

NOTE This assertion is established through composite_assembly_shape.

4.3.34 Composite_assembly to Three_d_geometry_set

Composite_assembly has shape defined by exactly one Three_d_geometry_set. Three_d_geometry_set defines shape for zero, one, or many Composite_assembly objects.

NOTE This assertion is established through composite_assembly_shape.

4.3.35 Composite_assembly to Wireframe_with_topology

Composite_assembly has shape defined by exactly one Wireframe_with_topology. Wireframe_with_-topology defines shape for zero, one, or many Composite_assembly objects.

NOTE This assertion is established through composite_assembly_shape.

4.3.36 Composite_assembly_sequence_definition to Composite_assembly

Composite_assembly_sequence_definition has components_in_sequence defined by a bag of one or many Composite_assemble objects. Composite_assembly defines components_in_sequence for zero, one, or many Composite_assembly_sequence_definition objects.

NOTE This assertion is established through component_class_for_assembly.

4.3.37 Composite_assembly_sequence_definition to Fea_material_property

Composite_assembly_sequence_definition has properties defined by zero or one Fea_material_property. Fea_material_property defines properties for zero, one, or many Composite_assembly_sequence_-definition objects.

4.3.38 Composite_assembly_sequence_definition to Filament_laminate

Composite_assembly_sequence_definition has components_in_sequence defined by a bag of one or many Filament_laminate objects. Filament_laminate defines components_in_sequence for zero, one, or many Composite_assembly_sequence_definition objects.

NOTE This assertion is established through component_class_for_assembly.

4.3.39 Composite_assembly_sequence_definition to Ply

Composite_assembly_sequence_definition has components_in_sequence defined by a bag of one or many Ply objects. Ply defines components_in_sequence for zero, one, or many Composite_assembly_-sequence_definition objects.

NOTE This assertion is established through component_class_for_assembly.

4.3.40 Composite_assembly_sequence_definition to Ply_laminate

Composite_assembly_sequence_definition has components_in_sequence defined by a bag of one or many Ply_laminate objects. Ply_laminate defines components_in_sequence for zero, one, or many Composite_-assembly_sequence_definition objects.

NOTE This assertion is established through component_class_for_assembly.

4.3.41 Composite_assembly_sequence_definition to Processed_core

Composite_assembly_sequence_definition has components_in_sequence defined by a bag of one or many Processed_core objects. Processed_core defines components_in_sequence for zero, one, or many Composite_assembly_sequence_definition objects.

NOTE This assertion is established through component_class_for_assembly.

4.3.42 Composite_assembly_table to Composite_assembly_sequence_definition

Composite_assembly_table has sequence_groups defined by a list of two or many Composite_assembly_-sequence_definition objects. Composite_assembly_sequence_definition defines sequence_groups for zero, one, or many Composite_assembly_table objects.

4.3.43 Constituent_part to Mass_measure

Constituent_part has weight defined by zero or one Mass_measure. Mass_measure defines weight for zero, one, or many Constituent_part objects.

4.3.44 Constituent_part to Part_version

Constituent_part is a constituent of exactly one Part_version. Part_version constituents of zero, one, or many Constituent_part objects.

4.3.45 Constraint to Fe_analysis_control_step

Constraint has steps defined by a set of one to many Fe_analysis_control_step objects. Fe_analysis_-control_step defines steps for zero, one, or many constraint objects.

4.3.46 Curve_cross_section to Curve_property

Curve_cross_section has property defined by exactly one Curve_property. Curve_property defines property for zero, one, or many Curve_cross_section objects.

4.3.47 Curve_cross_section to Idealized_analysis_shape

Curve_cross_section has section defined by zero or one Idealized_analysis_shape. Idealized_analysis_-shape defines section for zero, one, or many Curve_cross_section objects.

NOTE This assertion is established through curve_section_shape_select.

4.3.48 Curve_cross_section to Nominal_design_shape

Curve_cross_section has section defined by zero or one Nominal_design_shape. Nominal_design_shape defines section for zero, one, or many Curve_cross_section objects.

NOTE This assertion is established through curve_section_shape_select.

4.3.49 Curve_property to Curve_element

Curve_property refers to a set of one or many Curve_element objects in the role of defined_elements. Curve_element serves as defined_elements for exactly one Curve_property.

4.3.50 Curve_section_properties to Curve_property

Curve_section_properties has property defined by exactly one Curve_property. Curve_property defines property for zero, one, or many Curve_section_properties objects.

4.3.51 Date_effectivity to Date

Date_effectivity has end_date defined by zero or one Date. Date defines end_date for zero, one, or many Date_effectivity objects.

Date_effectivity has start_date defined by exactly one Date. Date defines start_date for zero, one, or many Date_effectivity objects.

4.3.52 Design_discipline_product_definition to Approval

Design_discipline_product_definition is approved by zero or one Approval. Approval approves zero, one, or many Design_discipline_product_definition objects.

4.3.53 Design_discipline_product_definition to Date

Design_discipline_product_definition has creation_date defined by exactly one Date. Date defines creation_date for zero, one, or many Design_discipline_product_definition objects.

4.3.54 Design_discipline_product_definition to File

Design_discipline_product_definition has cad_filename defined by zero or one File. File defines cad_-filename for zero, one, or many Design_discipline_product_definition objects.

4.3.55 Design_discipline_product_definition to Part_version

Design_discipline_product_definition has version defined by exactly one Part_version. Part_version defines version for zero, one, or many Design_discipline_product_definition objects.

4.3.56 Design_discipline_product_definition to Person_organization

Design_discipline_product_definition is created by exactly one Person_organization. Person_-organization defines creator for zero, one, or many Design_discipline_product_definition objects.

4.3.57 Design_material to Anisotropic_material

Design_material has material_callout defined by exactly one Anisotropic_material. Anisotropic_material defines material_callout for zero, one, or many Design_material objects.

NOTE This assertion is established through material_select.

4.3.58 Design_material to Isotropic_material

Design_material has material_callout defined by exactly one Isotropic_material. Isotropic_material defines material_callout for zero, one, or many Design_material objects.

NOTE This assertion is established through material_select.

4.3.59 Design_material to Laminate_table

Design_material has material_callout defined by exactly one Laminate_table. Laminate_table defines material_callout for zero, one, or many Design_material objects.

NOTE This assertion is established through material_select.

4.3.60 Design_material to Design_discipline_product_definition

Design_material applies to a part defined by exactly one Design_discipline_product_definition. Design_-discipline_product_definition serves as part_defined for zero, one, or many Design_material objects.

4.3.61 Directionally_explicit_element to Matrix

Directionally_explicit_element has an associated matrix defined by exactly one Matrix. Matrix defines associated_matrix for zero, one, or many Directionally_explicit_element objects.

4.3.62 Edge_zone_shape to Boundary_curve_representation

Edge_zone_shape has boundary defined by exactly one Boundary_curve_representation. Boundary_-curve_representation defines boundary for zero, one, or many Edge_zone_shape objects.

NOTE This assertion is established through boundary_class_zone.

4.3.63 Edge_zone_shape to Loop

Edge_zone_shape has boundary defined by exactly one Loop. Loop defines boundary for zero, one, or many Edge_zone_shape objects.

NOTE This assertion is established through boundary_class_zone.

4.3.64 Effectivity to Approval

Effectivity is approved by zero or one Approval. Approval approves zero, one, or many Effectivity objects.

4.3.65 Effectivity to Assembly

Effectivity has affected_assemblies defined by one or many Assembly objects. Assembly defines affected_assemblies for zero, one, or many Effectivity objects.

4.3.66 Effectivity to Product_configuration

Effectivity has configuration_item defined by exactly one Product_configuration. Product_configuration defines configuration_item for zero, one, or many Effectivity objects.

4.3.67 Element to Element_description

Element has description defined by zero or one Element_description. Element_description defines description for zero, one, or many Element objects.

4.3.68 Element to Fea_model

Element has model_ref defined by exactly one Fea_model. Fea_model defines model_ref for zero, one, or many Element objects.

4.3.69 Element to Node

Element has node_list defined by a list of one or many Node objects. Node defines node_list for zero, one, or many Element objects.

4.3.70 Element_property_geometric_relationship to Curve_property

Element_property_geometric_relationship has property_ref defined by exactly one Curve_property. Curve_property defines property_ref for zero, one, or many Element_property_geometric_relationship objects.

NOTE This assertion is established through element_property_select.

4.3.71 Element_property_geometric_relationship to Geometry_element

Element_property_geometric_relationship has geometry_ref defined by exactly one Geometry_element. Geometry_element defines geometry_ref for zero, one, or many Element_property_geometric_relationship objects.

4.3.72 Element_property_geometric_relationship to Surface_property

Element_property_geometric_relationship has property_ref defined by exactly one Surface_property. Surface_property defines property_ref for zero, one, or many Element_property_geometric_relationship objects.

NOTE This assertion is established through element_property_select.

4.3.73 Element_shape_relationship to Element

Element_shape_relationship has element_reference defined by exactly one Element. Element defines element_reference for zero, one, or many Element_shape_relationship objects.

4.3.74 Element_shape_relationship to Element_shape_aspect

Element_shape_relationship has element_aspect defined by exactly one Element_shape_aspect. Element_shape_aspect defines element_aspect for zero, one, or many Element_shape_relationship objects.

4.3.75 Element_shape_relationship to Geometry_element

Element_shape_relationship has shape defined by exactly one Geometry_element. Geometry_element defines shape for zero, one, or many Element_shape_relationship objects.

4.3.76 Explicit_element to Matrix

Explicit_element has an associated matrix defined by exactly one Matrix. Matrix defines associated_matrix for zero, one, or many Explicit_element objects.

4.3.77 Face_based_sheet_representation to Loop

Face_based_sheet_representation has inner_bounds defined by zero, one, or many Loop objects. Loop defines inner_bounds for zero, one, or many Face_based_sheet_representation objects.

Face_based_sheet_representation has outer_bound defined by exactly one Loop. Loop defines outer_bound for zero, one, or many Face_based_sheet_representation objects.

4.3.78 Face_based_sheet_representation to Surface

Face_based_sheet_representation has face_surface defined by exactly one Surface. Surface defines face_-surface for zero, one, or many Face_based_sheet_representation objects.

4.3.79 Fe_analysis to Approval

Fe_analysis is approved by zero or one Approval. Approval approves zero, one, or many Fe_analysis objects.

4.3.80 Fe_analysis to Fea_model

Fe_analysis has model_ref defined by exactly one Fea_model. Fea_model defines model_ref for zero, one, or many Fe_analysis objects.

4.3.81 Fe_analysis_control_step to Fe_analysis

Fe_analysis_control_step has analysis defined by exactly one Fe_analysis. Fe_analysis defines analysis for one or many Fe_analysis_control_step objects.

4.3.82 Fe_analysis_control_step to Fe_analysis_state

Fe_analysis_control_step has initial_input_state defined by exactly one Fe_analysis_state. Fe_analysis_-state defines initial_input_state for exactly one Fe_analysis_control_step.

4.3.83 Fe_analysis_results to Approval

Fe_analysis_results is approved by zero or one Approval. Approval approves zero, one, or many Fe_-analysis_results objects.

4.3.84 Fe_analysis_results_step to Fe_analysis_control_step

Fe_analysis_results_step has control defined by exactly one Fe_analysis_control_step. Fe_analysis_-control_step defines control for exactly one Fe_analysis_results_step.

4.3.85 Fe_analysis_results_step to Fe_analysis_results

Fe_analysis_results_step has result defined by exactly one Fe_analysis_results. Fe_analysis_results defines result for one or many Fe_analysis_results_step objects.

4.3.86 Fe_analysis_state to Fe_analysis_state_definition

Fe_analysis_state is defined by one or many Fe_analysis_state_definition objects. Fe_analysis_state_-definition defines exactly one Fe_analysis_state.

4.3.87 Fe_analysis_state_definition to Element

Fe_analysis_state_definition references exactly one Element. Element is referenced by zero, one, or many Fe_analysis_state_definition objects.

NOTE This assertion is established through definition_reference.

4.3.88 Fe_analysis_state_definition to Geometry_element

Fe_analysis_state_definition references exactly one Geometry_element. Geometry_element is referenced by zero, one, or many Fe_analysis_state_definition objects.

NOTE This assertion is established through definition_reference.

4.3.89 Fe_analysis_state_definition to Group

Fe_analysis_state_definition references exactly one Group. Group_element is referenced by zero, one, or many Fe_analysis_state_definition objects.

NOTE This assertion is established through definition_reference.

4.3.90 Fe_analysis_state_definition to Node

Fe_analysis_state_definition references exactly one Node. Node is referenced by zero, one, or many Fe_-analysis_state_definition objects.

NOTE This assertion is established through definition_reference.

4.3.91 Fea_material_definition to Curve_element

Fea_material_definition has elements defined by zero, one, or many of Curve_element objects. Curve_-element defines elements for zero, one, or many Fea_material_definition objects.

NOTE This assertion is established through definition_element.

4.3.92 Fea_material_definition to Shape_aspect

Fea_material_definition has elements defined by zero, one, or many of Shape_aspect objects. Shape_-aspect defines elements for zero, one, or many Fea_material_definition objects.

NOTE This assertion is established through definition_element.

4.3.93 Fea_material_definition to Surface_element

Fea_material_definition has elements defined by zero, one, or many of Surface_element objects. Surface_element defines elements for zero, one, or many Fea_material_definition objects.

NOTE This assertion is established through definition_element.

4.3.94 Fea_material_definition to Volume_element

Fea_material_definition has elements defined by zero, one, or many of Volume_element objects. Volume_element defines elements for zero, one, or many Fea_material_definition objects.

NOTE This assertion is established through definition_element.

4.3.95 Fea_material_property to Fea_material_definition

Fea_material_property has property_use defined by exactly one Fea_material_definition. Fea_material_-definition defines property_use for zero, one, or many Fea_material_property objects.

4.3.96 Fea_material_specification to Fea_material_definition

Fea_material_specification has specification_use defined by exactly one Fea_material_definition. Fea_-material_definition defines specification_use for zero, one, or many Fea_material_specification objects.

4.3.97 Fea_model to Analysis_discipline_product_definition

Fea_model applies to exactly one Analysis_discipline_product_definition. Analysis_discipline_product_-definition is modeled by zero, one, or many Fea_model objects.

4.3.98 Fea_model to Approval

Fea_model is approved by zero or one Approval. Approval approves zero, one, or many Fea_model objects.

4.3.99 Fea_model to Fea_model_description

Fea_model has description defined by exactly one Fea_model_description. Fea_model_description defines description for exactly one Fea_model.

4.3.100 Fea_model to File

Fea_model has fea_filename defined by zero or one File. File defines fea_filename for zero, one, or many Fea_model objects.

4.3.101 Filament_laminate to Filament_assembly

Filament_laminate has made_from defined by exactly one Filament_assembly. Filament_assembly defines made_from for zero, one, or many Filament_laminate objects.

4.3.102 Filament_laminate to Filament_laminate_shape

Filament_laminate has shape defined by exactly one Filament_laminate_shape. Filament_laminate_shape defines shape for zero, one, or many Filament_laminate objects.

4.3.103 Filament_laminate_shape to Idealized_analysis_shape

Filament_laminate_shape has cross_section defined by exactly one Idealized_analysis_shape. Idealized_-analysis_shape defines cross_section for zero, one, or many Filament_laminate_shape objects.

NOTE This assertion is established through curve_section_shape_select.

4.3.104 Filament_laminate_shape to Nominal_design_shape

Filament_laminate_shape has cross_section defined by exactly one Nominal_design_shape. Nominal_-design_shape defines cross_section for zero, one, or many Filament_laminate_shape objects.

NOTE This assertion is established through curve_section_shape_select.

4.3.105 Flat_pattern_ply_shape to Location

Flat_pattern_ply_shape has wrapup_origin_on_flat_pattern defined by exactly one Location. Location defines wrapup_origin_on_flat_pattern for zero, one, or many Flat_pattern_ply_shape objects.

Flat_pattern_ply_shape has wrapup_origin_on_surface defined by exactly one Location. Location defines wrapup_origin_on_surface for zero, one, or many Flat_pattern_ply_shape objects.

4.3.106 Geometric_model_representation to Geometry_element

Geometric_model_representation has elements defined by one or many Geometry_element objects. Geometry_element defines elements for zero, one, or many Geometric_model_representation objects.

4.3.107 Geometric_sheet_representation to Boundary_curve_representation

Geometric_sheet_representation has cutouts defined by zero, one, or many Boundary_curve_-representation objects. Boundary_curve_representation defines cutouts for zero, one, or many Geometric_sheet_representation objects.

Geometric_sheet_representation has outer_edge defined by exactly one Boundary_curve_representation. Boundary_curve_representation defines outer_edge for zero, one, or many Geometric_sheet_-representation objects.

4.3.108 Geometric_sheet_representation to Surface

Geometric_sheet_representation has basis_surface defined by exactly one Surface. Surface defines basis_surface for zero, one, or many Geometric_sheet_representation objects.

4.3.109 Group to Element

Group has elements defined by zero, or one, or many Element objects. Element defines elements for zero, one, or many Group objects.

4.3.110 Group to Node

Group has nodes defined by zero, or one, or many Node objects. Node defines nodes for zero, one, or many Group objects.

4.3.111 Group_relationship to Group

Group_relationship has related_group defined by exactly one Group. Group defines related_group for zero, one, or many Group_relationship objects.

Group_relationship has relating_group defined by exactly one Group. Group defines relating_group for zero, one, or many Group_relationship objects.

4.3.112 Idealized_analysis_shape to Geometric_model_representation

Idealized_analysis_shape has defining_shape defined by exactly one Geometric_model_representation. Geometric_model_representation defines defining_shape for zero, one, or many Idealized_analysis_shape objects.

4.3.113 Idealized_analysis_shape to Nominal_design_shape

Idealized_analysis_shape has basis defined by exactly one Nominal_design_shape. Nominal_design_-shape defines basis for zero, one, or many Idealized_analysis_shape objects.

4.3.114 Laminate_table to Fea_material_property

Laminate_table has properties defined by zero or one Fea_material_property. Fea_material_property defines properties for zero, one, or many Laminate_table objects.

4.3.115 Laminate_table to Reinforcement_orientation_basis

Laminate_table has basis defined by exactly one Reinforcement_orientation_basis. Reinforcement_-orientation_basis defines basis for zero, one, or many Laminate_table objects.

4.3.116 Laminate_table to Surface

Laminate_table has resulting_surface defined by zero or one Surface. Surface defines resulting_surface for zero, one, or many Laminate_table objects.

4.3.117 Laminate_table to Surface_with_direction

Laminate_table has base_surface defined by exactly one Surface_with_direction. Surface_with_direction defines base_surface for zero, one, or many Laminate_table objects.

4.3.118 Linear_constraint_equation to Geometry_element

Linear_constraint_equation has freedoms_and_coefficients_nodes defined by one or many Geometry_-element objects. Geometry_element defines freedoms_and_coefficients_nodes for zero or one Linear_-constraint_equation.

NOTE This assertion is established through constraint_definition_reference.

4.3.119 Linear_constraint_equation to Group

Linear_constraint_equation has freedoms_and_coefficients_nodes defined by one or many Group objects. Group defines freedoms_and_coefficients_nodes for zero or one Linear_constraint_equation.

NOTE This assertion is established through constraint_definition_reference.

4.3.120 Linear_constraint_equation to Node

Linear_constraint_equation has freedoms_and_coefficients_nodes defined by one or many Node objects. Node defines freedoms_and_coefficients_nodes for zero or one Linear_constraint_equation.

NOTE This assertion is established through constraint_definition_reference.

4.3.121 Linear_constraint_equation_value to Linear_constraint_equation

Linear_constraint_equation_value has equation defined by exactly one Linear_constraint_equation. Linear_constraint_equation defines equation for exactly one Linear_constraint_equation_value.

4.3.122 Make_from to Design_discipline_product_definition

Make_from has basis defined by exactly one Design_discipline_product_definition. Design_discipline_-product_definition defines basis for zero, one, or many Make_from objects.

Make_from has resultant defined by exactly one Design_discipline_product_definition. Design_-discipline_product_definition defines resultant for zero, one, or many Make_from objects.

4.3.123 Material_direction to Direction

Material_direction has material_orientation defined by exactly one Direction. Direction defines material_orientation for zero, one, or many Material_direction objects.

4.3.124 Material_property to Environment

Material_property has ambient defined by exactly one Environment. Environment defines ambient for zero, one, or many Material_property objects.

4.3.125 Material_property to Measure_value

Material_property has property_value defined by zero or one Measure_value. Measure_value defines property_value for zero, one, or many Material_property objects.

4.3.126 Modes_and_frequencies_control_step to Fe_analysis_state

Modes_and_frequencies_control_step has final_input_state defined by exactly one Fe_analysis_state. Fe_analysis_state defines final_input_state for exactly one Modes_and_frequencies_control_step.

4.3.127 Modes_and_frequencies_results_step to Calculated_state

Modes_and_frequencies_results_step has resulting_states defined by one or many Calculated_state objects. Calculated_state defines resulting_states for exactly one Modes_and_frequencies_results_step.

4.3.128 Nodal_degree_of_freedom_reduction to Geometry_element

Nodal_degree_of_freedom_reduction has node defined by exactly one Geometry_element. Geometry_-element defines node for zero or one Nodal_degree_of_freedom_reduction.

NOTE This assertion is established through constraint_definition_reference.

4.3.129 Nodal_degree_of_freedom_reduction to Group

Nodal_degree_of_freedom_reduction has node defined by exactly one Group. Group defines node for zero or one Nodal_degree_of_freedom_reduction.

NOTE This assertion is established through constraint_definition_reference.

4.3.130 Nodal_degree_of_freedom_reduction to Node

Nodal_degree_of_freedom_reduction has node defined by exactly one Node. Node defines node for zero or one Nodal_degree_of_freedom_reduction.

NOTE This assertion is established through constraint_definition_reference.

4.3.131 Node to Fea_model

Node has model_ref defined by exactly one Fea_model. Fea_model defines model_ref for zero, one, or many Node objects.

4.3.132 Node to Geometry_element

Node has location defined by exactly one Geometry_element. Geometry_element defines location for zero, one, or many Node objects.

4.3.133 Node to Nodal_results_coordinate_system

Node has results_coordinate_space defined by zero or one Nodal_results_coordinate_system. Nodal_-results_coordinate_system defines results_coordinate_space for zero, one, or many Node objects.

4.3.134 Node to Node_description

Node has description defined by zero or one Node_description. Node_description defines description for zero or one Node.

4.3.135 Node_shape to Point_model

Node_shape has a defining shape of exactly one Point_model. Point_model serves as defining_shape for zero, one, or many Node_shape objects.

4.3.136 Node_shape_relationship to Geometry_element

Node_shape_relationship has shape defined by exactly one Geometry_element. Geometry_element defines shape for zero, one, or many Node_shape_relationship objects.

4.3.137 Node_shape_relationship to Node

Node_shape_relationship has node_reference defined by exactly one Node. Node defines node_reference for zero, one, or many Node_shape_relationship objects.

4.3.138 Nominal_design_shape to Design_discipline_product_definition

Nominal_design_shape has design_view defined by exactly one Design_discipline_product_definition. Design_discipline_product_definition defines design_view for zero, one, or many Nominal_design_shape objects.

4.3.139 Nominal_design_shape to Geometric_model_representation

Nominal_design_shape has a defining shape of exactly one Geometric_model_representation. Geometric_model_representation serves as defining_shape for zero, one, or many Nominal_design_shape objects.

4.3.140 Output_request_state to Fe_analysis_control_step

Output_request_state has steps defined by one or many Fe_analysis_control_step objects. Fe_analysis_-control_step defines steps for zero, one, or many Output_request_state objects.

4.3.141 Part to Person_organization

Part has owner defined by exactly one Person_organization. Person_organization is the owner of zero, one, or many Part objects.

4.3.142 Part_version to Approval

Part_version is approved by zero or one Approval. Approval approves zero, one, or many Part_version objects.

4.3.143 Part_version to Mass_measure

Part_version has weight defined by zero or one Mass_measure. Mass_measure defines weight for zero, one, or many Part_version objects.

4.3.144 Part_version to Part

Part_version has part_number defined by exactly one Part. Part defines part_number for zero, one, or many Part_version objects.

4.3.145 Part_version to Person_organization

Part_version is created by exactly one Person_organization. Person_organization defines creator for zero, one, or many Part_version objects.

4.3.146 Percentage_laminate_table to Length_measure

Percentage_laminate_table has total_thickness defined by exactly one Length_measure. Length_measure defines total_thickness for zero, one, or many Percentage_laminate_table objects.

4.3.147 Percentage_laminate_table to Percentage_ply

Percentage_laminate_table has table_components defined by one or many Percentage_ply objects. Percentage_ply defines table_components for exactly one Percentage_laminate_table.

4.3.148 Percentage_ply to Percentage

Percentage_ply has volume_percent defined by exactly one Percentage. Percentage defines volume_-percent value for zero, one, or many Percentage_ply objects.

4.3.149 Percentage_ply to Ply_orientation_angle

Percentage_ply has strength_orientation defined by zero or one Ply_orientation_angle. Ply_orientation_-angle defines strength orientation angle for zero, one, or many Percentage_ply objects.

4.3.150 Percentage_ply to Stock_material

Percentage_ply has material defined by exactly one Stock_material. Stock_material defines material for zero, one, or many Percentage_ply objects.

4.3.151 Percentage_ply to Zone_structural_makeup

Percentage_ply has makeup_and_properties defined by zero or one Zone_structural_makeup. Zone_-structural_makeup defines makeup_and_properties for zero, one, or many Percentage_ply objects.

4.3.152 Ply to Discontinuous_fiber_assembly

Ply has material_type defined by exactly one Discontinuous_fiber_assembly. Discontinuous_fiber_-assembly defines material_type for zero, one, or many Ply objects.

NOTE This assertion is established through ply_stock_material.

4.3.153 Ply to Filament_assembly

Ply has material_type defined by exactly one Filament_assembly. Filament_assembly defines material_-type for zero, one, or many Ply objects.

NOTE This assertion is established through ply_stock_material.

4.3.154 Ply to Isotropic_material

Ply has material_type defined by exactly one Isotropic_material. Isotropic_material defines material_type for zero, one, or many Ply objects.

NOTE This assertion is established through ply_stock_material.

4.3.155 Ply to Length_measure

Ply has ply_thickness defined by exactly one Length_measure. Length_measure defines ply_thickness for zero, one, or many Ply objects.

4.3.156 Ply to Ply_orientation_angle

Ply has material_orientation defined by exactly one Ply_orientation_angle. Ply_orientation_angle defines material_orientation for zero, one, or many Ply objects.

4.3.157 Ply to Ply_piece

Ply has constituents defined by zero, or one, or many Ply_piece objects. Ply_piece defines constituents for zero, one, or many Ply objects.

4.3.158 Ply to Ply_shape

Ply has shape defined by exactly one Ply_shape. Ply_shape defines shape for zero, one, or many Ply objects.

4.3.159 Ply_laminate to Composite_sheet_representation

Ply_laminate has shape defined by exactly one Composite_sheet_representation. Composite_sheet_-representation defines shape for zero, one, or many Ply_laminate objects.

NOTE This assertion is established through ply_laminate_shape.

4.3.160 Ply_laminate to Ply_laminate_table

Ply_laminate has ply_table defined by exactly one Ply_laminate_table. Ply_laminate_table defines ply_-table for zero, one, or many Ply_laminate objects.

4.3.161 Ply_laminate to Three_d_geometry_set

Ply_laminate has shape defined by exactly one Three_d_geometry_set. Three_d_geometry_set defines shape for zero, one, or many Ply_laminate objects.

NOTE This assertion is established through ply_laminate_shape.

4.3.162 Ply_laminate_sequence_definition to Fea_material_property

Ply_laminate_sequence_definition has properties defined by zero or one Fea_material_property. Fea_-material_property defines properties for zero, one, or many Ply_laminate_sequence_definition objects.

4.3.163 Ply_laminate_sequence_definition to Ply

Ply_laminate_sequence_definition has plies_in_sequence defined by one or many Ply objects. Ply defines plies_in_sequence for zero, one, or many Ply_laminate_sequence_definition objects.

4.3.164 Ply_laminate_table to Ply_laminate_sequence_definition

Ply_laminate_table has sequence defined by a list of two or many Ply_laminate_sequence_definition objects. Ply_laminate_sequence_definition defines sequence for zero, one, or many Ply_laminate_table objects.

4.3.165 Ply_orientation_angle to Curve

Ply_orientation_angle has angle_reference defined by exactly one Curve. Curve defines angle_reference for zero, one, or many Ply_orientation_angle objects.

NOTE This assertion is established through angle_select.

4.3.166 Ply_orientation_angle to Direction

Ply_orientation_angle has angle_reference defined by exactly one Direction. Direction defines angle_-reference for zero, one, or many Ply_orientation_angle objects.

NOTE This assertion is established through angle_select.

4.3.167 Ply_orientation_angle to Point_path

Ply_orientation_angle has angle_reference defined by exactly one Point_path. Point_path defines angle_-reference for zero, one, or many Ply_orientation_angle objects.

NOTE This assertion is established through angle_select.

4.3.168 Ply_orientation_angle to Reinforcement_orientation_basis

Ply_orientation_angle has basis defined by exactly one Reinforcement_orientation_basis. Reinforcement_orientation_basis defines basis for zero, one, or many Ply_orientation_angle objects.

4.3.169 Ply_piece to Ply_shape

Ply_piece has shape defined by exactly one Ply_shape. Ply_shape defines shape for zero, one, or many Ply_piece objects.

NOTE This assertion is established through ply_piece_shape.

4.3.170 Ply_shape to Composite_sheet_representation

Ply_shape has a defining model represented by exactly one Composite_sheet_representation. Composite_sheet_representation serves as defining_model for zero, one, or many Ply_shape objects.

4.3.171 Ply_shape to Ply_shape

Ply_shape has basis defined by zero or one Ply_shape. Ply_shape defines basis for zero, one, or many Ply_shape objects.

4.3.172 Point_and_vector to Direction

Point_and_vector has vector defined by a list of two Direction objects. Direction defines vector for zero, one, or many Point_and_vector objects.

4.3.173 Point_and_vector to Point

Point_and_vector has location defined by exactly one Point. Point defines location for zero, one, or many Point_and_vector objects.

4.3.174 Point_element to Matrix

Point_element has an associated matrix defined by exactly one Matrix. Matrix defines associated_matrix for zero, one, or many Point_element objects.

4.3.175 Point_path to Point_and_vector

Point_path has directions defined by one or many Point_and_vector objects. Point_and_vector defines directions for zero, one, or many Point_path objects.

4.3.176 Point_zone_shape to Point

Point_zone_shape has location defined by exactly one Point. Point defines location for zero, one, or many Point_zone_shape objects.

4.3.177 Processed_core to Advanced_boundary_representation

Processed_core has shape defined by exactly one Advanced_boundary_representation. Advanced_-boundary_representation defines shape for zero, one, or many Processed_core objects.

NOTE This assertion is established through core_shape.

4.3.178 Processed_core to Beveled_sheet_representation

Processed_core has shape defined by exactly one Beveled_sheet_representation. Beveled_sheet_-representation defines shape for zero, one, or many Processed_core objects.

NOTE This assertion is established through core_shape.

4.3.179 Processed_core to Faceted_boundary_representation

Processed_core has shape defined by exactly one Faceted_boundary_representation. Faceted_boundary_-representation defines shape for zero, one, or many Processed_core objects.

NOTE This assertion is established through core_shape.

4.3.180 Processed_core to Ply

Processed_core has added_material defined by zero, one, or many Ply objects. Ply defines added_-material for zero, one, or many Processed_core objects.

4.3.181 Processed_core to Ply_orientation_angle

Processed_core has cell_direction defined by exactly one Ply_orientation_angle. Ply_orientation_angle defines cell_direction for zero, one, or many Processed_core objects.

4.3.182 Processed_core to Processed_core

Processed_core is made from zero, one, or many Processed_core objects. Processed_core makes zero, one, or many Processed_core objects.

4.3.183 Processed_core to Stock_core

Processed_core is made from core stock defined by exactly one Stock_core. Stock_core makes zero, one, or many Processed_core objects.

4.3.184 Product_configuration to Approval

Product_configuration is approved by zero or one Approval. Approval approves zero, one, or many Product_configuration objects.

4.3.185 Product_configuration to Part

Product_configuration refers to zero, one or many Part objects in the role of parts_configured. Part serves as parts_configured for zero, one, or many Product_configuration objects.

4.3.186 Product_configuration to Product_model

Product_configuration refers to exactly one Product_model in the role of model_name. Product_model serves as model_name for zero, one, or many Product_configuration objects.

4.3.187 Projected_ply_shape to Direction

Projected_ply_shape has projection_direction defined by zero or one Direction. Direction defines projection_direction for zero, one, or many Projected_ply_shape objects.

4.3.188 Reinforcement_orientation_basis to Direction

Reinforcement_orientation_basis has orientation defined by exactly one Direction. Direction defines orientation for zero, one, or many Reinforcement_orientation_basis objects.

4.3.189 Reinforcement_orientation_basis to Location

Reinforcement_orientation_basis has basis_location defined by exactly one location. location defines basis_location for zero, one, or many Reinforcement_orientation_basis objects.

4.3.190 Retention_period to Additional_design_information

Retention_period is applied to one or many Additional_design_information objects. Additional_design_-information is applied to by zero or one Retention_period.

NOTE This assertion is established through retention_data_select.

4.3.191 Retention_period to Analysis

Retention_period is applied to one or many Analysis objects. Analysis is applied to by zero or one Retention_period.

NOTE This assertion is established through retention_data_select.

4.3.192 Retention_period to Analysis_discipline_product_definition

Retention_period is applied to one or many Analysis_discipline_product_definition objects. Analysis_-discipline_product_definition is applied to by zero or one Retention_period.

NOTE This assertion is established through retention_data_select and design_or_analysis.

4.3.193 Retention_period to Analysis_report_representation

Retention_period is applied to one or many Analysis_report_representation objects. Analysis_report_-representation is applied to by zero or one Retention_period.

NOTE This assertion is established through retention_data_select.

4.3.194 Retention_period to Analysis_version

Retention_period is applied to one or many Analysis_version objects. Analysis_version is applied to by zero or one Retention_period.

NOTE This assertion is established through retention_data_select and design_or_analysis.

4.3.195 Retention_period to Approval

Retention_period is approved by zero or one Approval. Approval approves zero, one, or many Retention_period objects.

4.3.196 Retention_period to Assembly

Retention_period is applied to one or many Assembly objects. Assembly is applied to by zero or one Retention_period.

NOTE This assertion is established through retention_data_select and design_or_analysis.

4.3.197 Retention_period to Date

Retention_period has earliest_end_definition defined by exactly one Date. Date defines earliest_end_-definition for zero, one, or many Retention_period objects.

Retention_period has latest_end_definition defined by exactly one Date. Date defines latest_end_-definition for zero, one, or many Retention_period objects.

Retention_period has start_definition defined by exactly one Date. Date defines start_definition for zero, one, or many Retention_period objects.

4.3.198 Retention_period to Design_discipline_product_definition

Retention_period is applied to one or many Design_discipline_product_definition objects. Design_-discipline_product_definition is applied to by zero or one Retention_period.

NOTE This assertion is established through retention_data_select and design_or_analysis.

4.3.199 Retention_period to Design_material

Retention_period is applied to one or many Design_material objects. Design_material is applied to by zero or one Retention_period.

NOTE This assertion is established through retention_data_select.

4.3.200 Retention_period to Fe_analysis

Retention_period is applied to one or many Fe_analysis objects. Fe_analysis is applied to by zero or one Retention_period.

NOTE This assertion is established through retention_data_select.

4.3.201 Retention_period to Fe_analysis_results

Retention_period is applied to one or many Fe_analysis_results objects. Fe_analysis_results is applied to by zero or one Retention_period.

NOTE This assertion is established through retention_data_select.

4.3.202 Retention_period to Fea_model

Retention_period is applied to one or many Fea_model objects. Fea_model is applied to by zero or one Retention_period.

NOTE This assertion is established through retention_data_select.

4.3.203 Retention_period to Material_property

Retention_period is applied to one or many Material_property objects. Material_property is applied to by zero or one Retention_period.

NOTE This assertion is established through retention_data_select.

4.3.204 Retention_period to Material_specification

Retention_period is applied to one or many Material_specification objects. Material_specification is applied to by zero or one Retention_period.

NOTE This assertion is established through retention_data_select.

4.3.205 Retention_period to Part

Retention_period is applied to one or many Part objects. Part is applied to by zero or one Retention_-period.

NOTE This assertion is established through retention_data_select.

4.3.206 Retention_period to Part_version

Retention_period is applied to one or many Part_version objects. Part_version is applied to by zero or one Retention_period.

NOTE This assertion is established through retention_data_select and design_or_analysis.

4.3.207 Shape_aspect to Geometric_model_representation

Shape_aspect has geometry defined by exactly one Geometric_model_representation. Geometric_-model_representation defines geometry for zero, one, or many Shape_aspect objects.

4.3.208 Shape_aspect to Shape

Shape_aspect has parent_shape defined by exactly one Shape. Shape defines parent_shape for zero, one, or many Shape_aspect objects.

4.3.209 Shape_aspect to Specification

Shape_aspect has characteristics defined by zero, one, or many Specification objects. Specification defines characteristics for zero, one, or many Shape_aspect objects.

4.3.210 Single_point_constraint to Geometry_element

Single_point_constraint has required_node defined by exactly one Geometry_element. Geometry_-element defines required_node for zero or one Single_point_constraint.

NOTE This assertion is established through constraint_definition_reference.

4.3.211 Single_point_constraint to Group

Single_point_constraint has required_node defined by exactly one Group. Group defines required_node for zero or one Single_point_constraint.

NOTE This assertion is established through constraint_definition_reference.

4.3.212 Single_point_constraint to Node

Single_point_constraint has required_node defined by exactly one Node. Node defines required_node for zero or one Single_point_constraint.

NOTE This assertion is established through constraint_definition_reference.

4.3.213 Single_point_constraint_values to Single_point_constraint

Single_point_constraint_values has element defined by exactly one Single_point_constraint. Single_-point_constraint defines element for one or many Single_point_constraint_values objects.

4.3.214 Smeared_material to Length_measure

Smeared_material has total_thickness defined by Length_measure. Length_measure defines total_-thickness for zero, one, or many Smeared_material objects.

4.3.215 Static_control_step to Fe_analysis_state

Static_control_step has final_input_state defined by exactly one Fe_analysis_state. Fe_analysis_state defines final_input_state for zero, one, or many Static_control_step objects.

4.3.216 Static_results_step to Calculated_state

Static_results_step has resulting_state defined by exactly one Calculated_state. Calculated_state defines resulting_state for exactly one Static_results_step.

4.3.217 Stock_material to Approval

Stock_material is approved by zero or one Approval. Approval approves zero, one, or many Stock_-material objects.

4.3.218 Stock_material to Material_direction

Stock_material has reference_direction defined by exactly one Material_direction. Material_direction defines reference_direction for zero, one, or many Stock_material objects.

4.3.219 Stock_material to Material_property

Stock_material has property defined by exactly one Material_property. Material_property defines property for zero, one, or many Stock_material objects.

4.3.220 Stock_material to Material_specification

Stock_material has specified_material defined by exactly one Material_specification. Material_-specification defines specified_material for zero, one, or many Stock_material objects.

4.3.221 Stock_material to Part_version

Stock_material has of_part defined by exactly one Part_version. Part_version represents zero, one, or many Stock_material objects.

4.3.222 Substitute_part to Assembly

Substitute_part has base defined by exactly one Assembly. Assembly defines base for zero, one, or many Substitute_part objects.

4.3.223 Substitute_part to Part

Substitute_part has substitute defined by exactly one Part. Part defines substitute for zero, one, or many Substitute_part objects.

4.3.224 Substructure_element to Fea_model

Substructure_element has substructure_model_ref defined by exactly one Fea_model. Fea_model defines substructure_model_ref for zero, one, or many Substructure_element objects.

4.3.225 Substructure_node_relationship to Node

Substructure_element has related_node defined by exactly one Node. Node defines related_to for zero, one, or many Substructure_node_relationship objects.

Substructure_element has relating_node defined by exactly one Node. Node defines relating_to for zero, one, or many Substructure_node_relationship objects.

4.3.226 Supplied_part_version to Approval

Supplied_part_version is approved by zero or one Approval. Approval approves zero, one, or many Supplied_part_version objects.

4.3.227 Supplied_part_version to Part_version

Supplied_part_version is identified as exactly one Part_version. Part_version identifies zero, one, or many Supplied_part_version objects.

4.3.228 Supplied_part_version to Supplier

Supplied_part_version is produced by exactly one Supplier. Supplier produces zero, one, or many Supplied_part_version objects.

4.3.229 Supplier to Person_organization

Supplier has identified_as defined by exactly one Person_organization. Person_organization defines identified_as for zero, one, or many Supplier objects.

4.3.230 Surface_ply_shape to Length_measure

Surface_ply_shape has offset defined by exactly one Length_measure. Length_measure defines offset for zero, one, or many Surface_ply_shape objects.

4.3.231 Surface_property to Surface_element

Surface_property refers to one or many Surface_element objects in the role of defined_elements. Surface_element serves as defined_elements for exactly one Surface_property.

4.3.232 Surface_thickness to Length_measure

Surface_thickness has thickness defined by exactly one Length_measure. Length_measure defines thickness for zero, one, or many Surface_thickness objects.

4.3.233 Surface_thickness to Surface_property

Surface_thickness has property defined by exactly one Surface_property. Surface_property defines property for one or many Surface_thickness objects.

4.3.234 Surface_with_direction to Direction

Surface_with_direction has material_direction defined by exactly one Direction. Direction defines material_direction for zero, one, or many Surface_with_direction objects.

4.3.235 Surface_with_direction to Surface

Surface_with_direction has a defining surface of exactly one Surface. Surface serves as defining_surface for zero, one, or many Surface_with_direction objects.

4.3.236 Thickness_laminate_table to Filament_laminate

Thickness_laminate_table has table_components defined by two or many Filament_laminate objects. Filament_laminate defines table_components for zero, one, or many Thickness_laminate_table objects.

NOTE This assertion is established through thickness_laminate_table_component.

4.3.237 Thickness_laminate_table to Ply

Thickness_laminate_table has table_components defined by two or many Ply objects. Ply defines table_-components for zero, one, or many Thickness_laminate_table objects.

NOTE This assertion is established through thickness_laminate_table_component.

4.3.238 Thickness_laminate_table to Processed_core

Thickness_laminate_table has table_components defined by two or many Processed_core objects. Processed_core defines table_components for zero, one, or many Thickness_laminate_table objects.

NOTE This assertion is established through thickness_laminate_table_component.

4.3.239 Three_d_geometry_set to Curve

Three_d_geometry_set has a defining boundary of one or many Curve objects. Curve serves as defining_-boundary for zero, one, or many Three_d_geometry_set objects.

4.3.240 Three_d_geometry_set to Surface

Three_d_geometry_set has basis_surface defined by exactly one Surface. Surface defines basis_surface for zero, one, or many Three_d_geometry_set objects.

4.3.241 Usage_constraint to Specification

Usage_constraint constrains exactly one Specification. Specification is constrained by zero, one, or many Usage_constraint objects.

4.3.242 Work_order to Analysis_discipline_product_definition

Work_order has versions defined by one or many Analysis_discipline_product_definition objects. Analysis_discipline_product_definition defines versions for zero, one, or many Work_order objects.

NOTE This assertion is established through design_or_analysis.

4.3.243 Work_order to Analysis_version

Work_order has versions defined by one or many Analysis_version objects. Analysis_version defines versions for zero, one, or many Work_order objects.

NOTE This assertion is established through design_or_analysis.

4.3.244 Work_order to Approval

Work_order is approved by zero or one Approval. Approval approves zero, one, or many Work_order objects.

4.3.245 Work_order to Assembly

Work_order has versions defined by one or many Assembly objects. Assembly defines versions for zero, one, or many Work_order objects.

NOTE This assertion is established through design_or_analysis.

4.3.246 Work_order to Design_discipline_product_definition

Work_order has versions defined by one or many Design_discipline_product_definition objects. Design_discipline_product_definition defines versions for zero, one, or many Work_order objects.

NOTE This assertion is established through design_or_analysis.

4.3.247 Work_order to File

Work_order has additional_data defined by zero or one File. File defines additional_data for zero, one, or many Work_order objects.

Work_order has analysis_data defined by zero or one File. File defines analysis_data for zero, one, or many Work_order objects.

4.3.248 Work_order to Part_version

Work_order has versions defined by one or many Part_version objects. Part_version defines versions for zero, one, or many Work_order objects.

NOTE This assertion is established through design_or_analysis.

4.3.249 Work_order to Work_request

Work_order incorporates one or many Work_request objects. Work_request is incorporated by zero or one Work_order.

4.3.250 Work_request to Analysis_discipline_product_definition

Work_request that is a start request has affective_parts defined by zero, one or many Analysis_-discipline_product_definition objects. Work_request that is a change request has affective_parts defined by one or many Analysis_discipline_product_definition objects. Analysis_discipline_product_definition defines affective_parts for zero, one, or many Work_request objects.

NOTE This assertion is established through design_or_analysis.

4.3.251 Work_request to Analysis_version

Work_request that is a start request has affective_parts defined by zero, one or many Analysis_version objects. Work_request that is a change request has affective_parts defined by one or many Analysis_-version objects. Analysis_version defines affective_parts for zero, one, or many Work_request objects.

NOTE This assertion is established through design_or_analysis.

4.3.252 Work_request to Approval

Work_request is approved by zero or one Approval. Approval approves zero, one, or many Work_request objects.

4.3.253 Work_request to Assembly

Work_request that is a start request has affective_parts defined by zero, one or many Assembly objects. Work_request that is a change request has affective_parts defined by one or many Assembly objects. Assembly defines affective_parts for zero, one, or many Work_request objects.

NOTE This assertion is established through design_or_analysis.

4.3.254 Work_request to Date

Work_request has request_date defined by exactly one Date. Date defines request_date for zero, one, or many Work_request objects.

4.3.255 Work_request to Design_discipline_product_definition

Work_request that is a start request has affective_parts defined by zero, one or many Design_discipline_-product_definition objects. Work_request that is a change request has affective_parts defined by one or many Design_discipline_product_definition objects. Design_discipline_product_definition defines affective_parts for zero, one, or many Work_request objects.

NOTE This assertion is established through design_or_analysis.

4.3.256 Work_request to Part_version

Work_request that is a start request has affective_parts defined by zero, one or many Part_version objects. Work_request that is a change request has affective_parts defined by one or many Part_version objects. Part_version defines affective_parts for zero, one, or many Work_request objects.

NOTE This assertion is established through design_or_analysis.

4.3.257 Work_request to Person_organization

Work_request has recipients defined by one or many Person_organization objects. Person_organization defines recipients for zero, one, or many Work_request objects.

4.3.258 Zone_structural_makeup to Zone_structural_makeup_shape_-representation

Zone_structural_makeup has boundary defined by zero or one Zone_structural_makeup_shape_-representation. Zone_structural_makeup_shape_representation defines boundary for one or many Zone_-structural_makeup objects.

5 Application interpreted model

5.1 Mapping table

This clause contains the mapping table 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). The mapping table is organized in five columns:

Column 1) Application element: Name of an application element as it appears in the application object definition in 4.2. Application object names are written in uppercase. Attribute names and assertions are listed after the application object to which they belong and are written in lower case.

Column 2) AIM element: Name of an AIM element as it appears in the AIM (see Annex A), the term ‘IDENTICAL MAPPING’, or the term ‘PATH’. AIM entities are written in lower case. Attribute names of AIM entities are referred to as <entity name>,<attribute name>. The mapping of an application element may result in several related AIM elements. Each of these AIM elements requires a line of its own in the table. The term ‘IDENTICAL MAPPING’ indicates that both application objects of an application assertion map to the same AIM element. The term ‘PATH’ indicates that the application assertion maps to the entire reference path.

Column 3) Source: For those AIM elements that are interpreted from the integrated resources, this is the number of the corresponding part of ISO 10303. For those AIM elements that are created for the purpose of this part of ISO 10303, this is the number of this part. For those AIM elements that are directly incorporated from an AIC, this is the AIC reference.

Column 4) Rules: One or more numbers may be given that refer to rules that apply to the current AIM element or 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. The expanded names of the rules are listed after the table.

Column 5) Reference path: To describe fully the mapping of an application object, it may be necessary to specify a reference path through several related AIM elements. The reference path column documents the role of an AIM element relative to the AIM element in the row succeeding 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 up to its supertype from an integrated resource or an AIC is specified. In the case of a bi-directional reference from an AIM element with two attributes, each of which spawns a reference path, each reference path is enclosed by a set of parentheses. The AIM element that is the root of both reference paths is documented either between the sets of parentheses or preceding each set of parentheses.

For the expression of reference paths and the relationships between AIM elements the following notational conventions apply:

a) [] : enclosed section constrains multiple AIM elements or sections of the reference path are required to satisfy an information requirement;

b) () : enclosed section constrains multiple AIM elements or sections of the reference path are identified as alternatives to the mapping to satisfy an information requirement;

c) {} : enclosed section constrains the reference path to satisfy an information requirement;

d) <> : enclosed section constrains at one or more required reference path;

e) || : enclosed section constrains the supertype entity;

f) -> : attribute references the entity or select type given the following row;

g) <- : entity or select type is referenced by the attribute in the following row;

h) [i] : attribute is an aggregation of which a single member is given in the following row;

i) [n] : attribute is an aggregation of which member n is given in the following row;

j) => : entity is a supertype of the entity given in the following row;

k) <= : entity is a subtype of the entity given in the following row;

l) = : the string, select or enumeration type is constrained to a choice of value;

m) \ : the reference path expression continues on the next line.

Table 1 - Mapping table for activity_control UoF

Application element	AIM element	Source	Rules	Reference Path
CHANGE_ORDER change_order <= work_order	action_directive	41	1, 49	action_directive <- directed_action.directive directed_action <= executed_action <= action (action.name = 'design change') (action.name = 'analysis change')
adopted_solution	action_method	41		action_directive <- directed_action.directive directed_action <= executed_action <= action action.chosen_method -> action_method
change_order TO date (as change_date)	PATH		1, 16, 17, 40	(action_directive <- directed_action.directive directed_action <= executed_action <= action date_time_item = action date_time_item <- applied_date_and_time_assignment.items[i] applied_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment date_and_time_assignment.role -> date_time_role date_time_role.name = 'change date'} date_and_time_assignment.assigned_date_and_time -> date_and_time) *(continued on the following page)*

Table 1 - Mapping table for activity_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
change_order TO date (as change_date) *(continued)*				(action_directive <- directed_action.directive directed_action <= executed_action <= action date_item = action date_item <- applied_date_assignment.items[i] applied_date_assignment <= date_assignment {date_assignment date_assignment.role -> date_role date_role.name = 'change date'} date_assignment.assigned_date -> date)
CHANGE_REQUEST change_request <= work_request	versioned_action_request	41	2, 3	
consequence	action_method.consequence	41		versioned_action_request <- action_request_solution.request action_request_solution action_request_solution.method -> action_method action_method.consequence
recommended_solution	action_method	41		versioned_action_request <- action_request_solution.request action_request_solution action_request_solution.method -> action_method
version	versioned_action_request.version	41		
DATE	(date_and_time) (date)	41 41	14, 15, 40	

Table 1 - Mapping table for activity_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
START_ORDER start_order <= work_order	action_directive	41	1	action_directive <- directed_action.directive directed_action <= executed_action <= action (action.name = 'design') (action.name = 'analysis')
START_REQUEST start_request <= work_request	versioned_action_request	41	2,3	versioned_action_request versioned_action_request.version = 'initial'
WORK_ORDER	action_directive	41	46	
work_order_identification (UNIQUE)	action_directive.name	41		
work_order TO (OPTIONAL) file (as additional_data)	action_directive.comment	41		
work_order TO (OPTIONAL) file (as analysis_data)	action_directive.analysis	41		
work_order TO (OPTIONAL) approval FROM authorization UoF (as approved_by)	PATH			action_directive <- directed_action.directive directed_action <= executed_action <= action approval_item = action approval_item <- applied_approval_assignment.items[i] applied_approval_assignment)
work_order TO (ONE OR MANY UNIQUE) work_request (as incorporates)	PATH			action_directive {action_directive <- directed_action.directive directed_action <= executed_action <= action} action_directive.requests[i] -> versioned_action_request

Table 1 - Mapping table for activity_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
work_order TO (ONE OR MANY UNIQUE) analysis_discipline_product_definition (as versions)	PATH			action_directive <- directed_action.directive directed_action <= executed_action <= action <- action_assignment.assigned_action action_assignment => applied_action_assignment applied_action_assignment.items[i] -> action_item action_item = product_definition product_definition
work_order TO (ONE OR MANY UNIQUE) analysis_version (as versions)	PATH			action_directive <- directed_action.directive directed_action <= executed_action <= action <- action_assignment.assigned_action action_assignment => applied_action_assignment applied_action_assignment.items[i] -> action_item action_item = product_definition_formation product_definition_formation
work_order TO (ONE OR MANY UNIQUE) assembly (as versions)	PATH			action_directive <- directed_action.directive directed_action <= executed_action <= action <- action_assignment.assigned_action action_assignment => applied_action_assignment applied_action_assignment.items[i] -> action_item action_item = product_definition_relationship product_definition_relationship => product_definition_usage => assembly_component_usage)

Table 1 - Mapping table for activity_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
work_order TO (ONE OR MANY UNIQUE) design_discipline_product_definition (as versions)	PATH			action_directive <- directed_action.directive directed_action <= executed_action <= action <- action_assignment.assigned_action action_assignment => applied_action_assignment applied_action_assignment.items[i] -> action_item action_item = product_definition product_definition
work_order TO (ONE OR MANY UNIQUE) part_version (as versions)	PATH			action_directive <- directed_action.directive directed_action <= executed_action <= action <- action_assignment.assigned_action action_assignment => applied_action_assignment applied_action_assignment.items[i] -> action_item action_item = product_definition_formation product_definition_formation

Table 1 - Mapping table for activity_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
WORK_REQUEST	versioned_action_request	41	50	
description	versioned_action_request.description	41		
reason	versioned_action_request.purpose	41		
status	action_request_status	41	38, 50	versioned_action_request <- action_request_status.assigned_request action_request_status
work_request_identification (UNIQUE)	version_action_request.id	41		
work_request TO (ZERO, ONE OR MANY UNIQUE) analysis_discipline_product_definition (as affective_parts)	PATH			versioned_action_request <- action_request_assignment.assigned_action_request action_request_assignment => applied_action_request_assignment applied_action_request_assignment.items[i] -> action_request_item action_request_item = product_definition product_definition
work_request TO (ZERO, ONE OR MANY UNIQUE) analysis_version (as affective_parts)	PATH			versioned_action_request <- action_request_assignment.assigned_action_request action_request_assignment => applied_action_request_assignment applied_action_request_assignment.items[i] -> action_request_item action_request_item = product_definition_formation product_definition_formation
work_request TO (ZERO, ONE OR MANY UNIQUE) assembly (as affective_parts)	PATH			versioned_action_request <- action_request_assignment.assigned_action_request action_request_assignment => applied_action_request_assignment applied_action_request_assignment.items[i] -> action_request_item action_request_item = product_definition_relationship product_definition_relationship => product_definition_usage => assembly_component_usage

Table 1 - Mapping table for activity_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
work_request TO (ZERO, ONE OR MANY UNIQUE) design_discipline_product_definition (as affective_parts)	PATH			versioned_action_request <- action_request_assignment.assigned_action_request action_request_assignment => applied_action_request_assignment applied_action_request_assignment.items[i] -> action_request_item action_request_item = product_definition product_definition
work_request TO (ZERO, ONE OR MANY UNIQUE) part_version (as affective_parts)	PATH			versioned_action_request <- action_request_assignment.assigned_action_request action_request_assignment => applied_action_request_assignment applied_action_request_assignment.items[i] -> action_request_item action_request_item = product_definition_formation product_definition_formation
work_request TO (OPTIONAL) approval FROM authorization UoF (as approved_by)	PATH			versioned_action_request approval_item = versioned_action_request approval_item <- applied_approval_assignment.items[i] applied_approval_assignment
work_request TO (ONE OR MANY UNIQUE) person_organization (as recipients)	PATH			(versioned_action_request person_organization_item = versioned_action_request person_organization_item <- applied_person_and_organization_assignment.items[i] applied_person_and_organization_assignment) (versioned_action_request organization_item = versioned_action_request organization_item <- applied_organization_assignment.items[i] applied_organization_assignment)

Table 1 - Mapping table for activity_control UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
work_request TO date (as request_date)	PATH		16, 17, 40	(versioned_action_request date_time_item = versioned_action_request date_time_item <- applied_date_and_time_assignment.items[i] applied_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment date_and_time_assignment.role -> date_time_role date_time_role.name = 'request date'} date_and_time_assignment.assigned_date_and_time -> date_and_time) (versioned_action_request date_item = versioned_action_request date_item <- applied_date_assignment.items[i] applied_date_assignment <= date_assignment {date_assignment date_assignment.role -> date_role date_role.name = 'request date'} date_assignment.assigned_date -> date)

Table 2 - Mapping table for advanced_boundary_representation UoF

Application element	AIM element	Source	Rules	Reference Path
ADVANCED_BOUNDARY_REPRESENTATION advanced_boundary_representation <= geometric_model_representation	advanced_brep_shape_representation	514	47	advanced_brep_shape_representation <= shape_representation
GEOMETRIC_MODEL_REPRESENTATION	shape_representation	41	21, 23, 25, 26, 47	
role	representation.name	43		shape_representation <= representation representation.name
geometric_model_representation TO (ONE OR MANY UNIQUE) geometry_element (as elements)	PATH			shape_representation <= representation representation.items -> representation_item => geometric_representation_item
GEOMETRY_ELEMENT	geometric_representation_item	43		

Table 3 - Mapping table for analysis_report UoF

Application element	AIM element	Source	Rules	Reference Path
ANALYSIS_REPORT_REPRESENTATION	document	41	18	
analysis_report_representation TO (OPTIONAL) fe_analysis (as control)	PATH			document <- product_definition_with_associated_documents.\ documentation_ids[i] product_definition_with_associated_documents <= product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect {shape_aspect => fea_model_definition} shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition {property_definition => structural_response_property} property_definition <- property_definition_representation.definition property_definition_representation {property_definition_representation => structural_response_property_definition_representation} property_definition_representation.used_representation -> representation => fea_model <- control.model_ref control

Table 3 - Mapping table for analysis_report UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
analysis_report_representation TO fea_model (as model_ref)	PATH			document <- product_definition_with_associated_documents.\ documentation_ids[i] product_definition_with_associated_documents <= product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect {shape_aspect => fea_model_definition} shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition {property_definition => structural_response_property} property_definition <- property_definition_representation.definition property_definition_representation {property_definition_representation => structural_response_property_definition_representation} property_definition_representation.used_representation -> representation => fea_model

Table 3 - Mapping table for analysis_report UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
analysis_report_representation TO (OPTIONAL) fe_analysis_results (as result)	PATH			document <- product_definition_with_associated_documents.\ documentation_ids[i] product_definition_with_associated_documents <= product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect {shape_aspect => fea_model_definition} shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition {property_definition => structural_response_property} property_definition <- property_definition_representation.definition property_definition_representation {property_definition_representation => structural_response_property_definition_representation} property_definition_representation.used_representation -> representation => fea_model <- control.model_ref control <- analysis_step.analysis_control analysis_step => result_analysis_step result_analysis_step.analysis_result -> result

Table 4 - Mapping table for analysis_report UoF

Application element	AIM element	Source	Rules	Reference Path
GRAPHICAL_REPRESENTATION graphical_representation <= analysis_report_representation	document	41	18	document
graphical_filename	document.id	41	18	document {document document.kind -> document_type document_type.product_data_type = \\ 'graphical report file'} document.id
TABULAR_REPRESENTATION tabular_representation <= analysis_report_representation	document	41	18	document
tabular_filename	document.id	41	18	document {document document.kind -> document_type document_type.product_data_type = \\ 'tabular report file'} document.id

Table 4 - Mapping table for assembly UoF

Application element	AIM element	Source	Rules	Reference Path
ASSEMBLY	assembly_component_usage	44	10	
security_code	security_classification_level.name	41	24, 44, 27	assembly_component_usage <= product_definition_usage security_classification_item = product_definition_usage security_classification_item <- applied_security_classification.items[i] applied_security_classification <= security_classification_assignment security_classification_assignment.\ applied_security_classification -> security_classification security_classification.security_level -> security_classification_level security_classification_level.name
assembly TO design_discipline_product_definition FROM part_identification UoF (as assembly_part)	PATH			assembly_component_usage <= product_definition_usage <= product_definition_relationship product_definition_relationship.relating_product_definition -> product_definition
assembly TO design_discipline_product_definition FROM part_identification UoF (as component_part)	PATH			assembly_component_usage <= product_definition_usage <= product_definition_relationship product_definition_relationship.related_product_definition -> product_definition
COMPONENT_ASSEMBLY_POSITION #1: Locate component in an upper level assembly using a transformation #2: Locate component in an upper level assembly using a mapped item	#1 ([shape_representation_relationship] [representation_relationship_with_\ transformation]) #2 (mapped_item)	43 43	10 47	

Table 4 - Mapping table for assembly UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
transformation #1: Locate component in an upper level assembly using a transformation #2: Locate component in an upper level assembly using a mapped item	#1 (transformation) #2 ([mapped_item.mapping_target] [representation_map.mapping.origin])	43 43 43	10 47	#1 (shape_representation_relationship <= representation_relationship => representation_relationship_with_transformation representation_relationship_with_transformation.\ transformation_operator - > transformation) #2 ([mapped_item mapped_item.mapping_target] [mapped_item mapped_item.mapping_source -> representation_map representation_map.mapping_origin]
component_assembly_position TO next_higher_assembly (as definition) #1: as representation_relationship #2: as mapped_item	PATH		10, 47	#1 ([representation_relationship_with_transformation <= representation_relationship => shape_representation_relationship] [shape_representation_relationship <- context_dependent_shape_representation.\ representation_relation context_dependent_shape_representation context_dependent_shape_representation.\ represented_product_relation -> product_definition_shape <= property_definition property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition = \ product_definition_relationship product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence])

Table 4 - Mapping table for assembly UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
component_assembly_position TO next_higher_assembly (as definition) (*continued*)	PATH			#2 (mapped_item <= representation_item <- representation.items[i] representation <- {representation => shape_representation} property_definition_representation.used_representation property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.definition -> property_definition {property_definition => product_definition_shape} property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition = \ product_definition_relationship product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence)

Table 4 - Mapping table for assembly UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
component_assembly_position TO geometric_model_representation (as assembly_shape) #1: as item_defined_transformation #2: as functionally_defined_transformation #3: as mapped_item	PATH		10	#1 ([shape_representation_relationship <= representation_relationship representation_relationship.rep_1 -> representation => shape_representation] [representation_relationship_with_transformation representation_relationship_with_transformation.\ transformation_operator -> transformation transformation = item_defined_transformation item_defined_transformation item_defined_transformation.transform_item_1 -> representation_item <- representation.items[i] representation => shape_representation]) #2 ([shape_representation_relationship <= representation_relationship representation_relationship.rep_1 -> representation => shape_representation] [representation_relationship_with_transformation representation_relationship_with_transformation.\ transformation_operator -> transformation transformation = functionally_defined_transformation functionally_defined_transformation])

Table 4 - Mapping table for assembly UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
component_assembly_position TO geometric_model_representation (as assembly_shape) (*continued*)	PATH			#3 (mapped_item [mapped_item.mapping_target -> representation_item <- representation.items[i] representation representation => shape_representation] [mapped_item <= representation_item <- representation.items[i] representation => shape_representation])
component_assembly_position TO geometric_model_representation (as component_shape) #1: as item_defined_transformation #2: as functionally_defined_transformation #3: as mapped_item	PATH		10	#1 ([shape_representation_relationship <= representation_relationship representation_relationship.rep_2 -> representation => shape_representation] [representation_relationship_with_transformation representation_relationship_with_transformation.\ transformation_operator -> transformation transformation = item_defined_transformation item_defined_transformation item_defined_transformation.transform_item_2 -> representation_item <- representation.items[i] representation => shape_representation])

Table 4 - Mapping table for assembly UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
component_assembly_position TO geometric_model_representation (as component_shape) (*continued*)	PATH			#2 ([shape_representation_relationship <= representation_relationship representation_relationship.rep_2 -> representation => shape_representation] [representation_relationship_with_transformation representation_relationship_with_transformation.\ transformation_operator -> transformation transformation = functionally_defined_transformation functionally_defined_transformation]) #3 (mapped_item mapped_item.mapping_source -> representation_map [representation_map.mapping_origin -> representation_item <- representation.items[i] representation] [representation_map.mapped_representation -> representation] representation => shape_representation)
MAKE_FROM	design_make_from_relationship	209		
make_from TO design_discipline_product_definition (as basis)	PATH			design_make_from_relationship <= product_definition_relationship product_definition_relationship.relating_product_definition -> product_definition
make_from TO design_discipline_product_definition (as resultant)	PATH			design_make_from_relationship <= product_definition_relationship product_definition_relationship.relating_product_definition -> product_definition

Table 4 - Mapping table for assembly UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
NEXT_HIGHER_ASSEMBLY	next_assembly_usage_occurrence	44		next_assembly_usage_occurrence <= assembly_component_usage
next_higher_assembly <= assembly				
as_required	descriptive_measure	41	7	next_assembly_usage_occurrence <= assembly_component_usage => quantified_assembly_component_usage quantified_assembly_component_usage.quantity -> measure_with_unit measure_with_unit.value_component -> measure_value measure_value = descriptive_measure descriptive_measure
component_quantity	context_dependent_measure	41		next_assembly_usage_occurrence <= assembly_component_usage => quantified_assembly_component_usage quantified_assembly_component_usage.quantity -> measure_with_unit measure_with_unit.value_component -> measure_value measure_value = context_dependent_measure context_dependent_measure
reference_designator (OPTIONAL)	assembly_component_usage.\ reference_designator	44		next_assembly_usage_occurrence <= assembly_component_usage assembly_component_usage.reference_designator
unit_of_measure	context_dependent_unit.name	41	19	next_assembly_usage_occurrence <= assembly_component_usage => quantified_assembly_component_usage quantified_assembly_component_usage.quantity -> measure_with_unit measure_with_unit.unit_component -> unit unit = named_unit named_unit => context_dependent_unit context_dependent_unit.name
PROMISSORY_USAGE	promissory_usage_occurrence	44		promissory_usage_occurrence <= assembly_component_usage
promissory_usage <= assembly				

Table 4 - Mapping table for assembly UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
SUBSTITUTE_PART	product	41		
substitute_part TO assembly (as base)	PATH			product <- product_definition_formation.of_product product_definition_formation <- product_definition.formation product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship => product_definition_usage => assembly_component_usage <- assembly_component_usage_substitute.substitute assembly_component_usage_substitute assembly_component_usage_substitute.base -> assembly_component_usage
substitute_part TO part FROM part_identification UoF (as substitute)	PATH			product <- product_definition_formation.of_product product_definition_formation <- product_definition.formation product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship => product_definition_usage => assembly_component_usage <- assembly_component_usage_substitute.substitute assembly_component_usage_substitute assembly_component_usage_substitute.base -> assembly_component_usage <= product_definition_usage <= product_definition_relationship product_definition_relationship.relating_product_definition -> product_definition product_definition.formation -> product_definition_formation {product_definition_formation => product_definition_formation_with_specified_source} product_definition_formation.of_product -> product

Table 4 - Mapping table for assembly UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
SUPPLIED_PART_VERSION	product_definition_formation_\ with_specified_source	41		
certification_required	applied_certification_assignment	209	12	product_definition_formation_with_specified_source <= product_definition_formation <- product_definition.formation product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship certification_item = product_definition_relationship certification_item <- applied_certification_assignment.items[i] applied_certification_assignment
supplier_part_number (OPTIONAL)	product.id	41		product_definition_formation_with_specified_source <= product_definition_formation <- product_definition.formation product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship product_definition_relationship.related_product_definition -> product_definition product_definition.formation -> product_definition_formation {product_definition_formation => product_definition_formation_with_specified_source} product_definition_formation.of_product -> product product.id
supplied_part_version TO (OPTIONAL) approval FROM authorization UoF (as approved_by)	PATH			product_definition_formation_with_specified_source <= product_definition_formation approval_item = product_definition_formation approval_item <- applied_approval_assignment.items[i] applied_approval_assignment

Table 4 - Mapping table for assembly UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
supplied_part_version TO part_version (as is_identified_as)	PATH			product_definition_formation_with_specified_source <= product_definition_formation <- product_definition.formation product_definition <- product_definition_relationship.related_product_definition product_definition_relationship {product_definition_relationship => supplied_part_relationship} product_definition_relationship product_definition_relationship.relating_product_definition product_definition product_definition.formation -> product_definition_formation
supplied_part_version TO supplier (as produced_by)	PATH			(product_definition_formation_with_specified_source <= product_definition_formation person_organization_item = product_definition_formation person_organization_item <- applied_person_and_organization_assignment.items[i] applied_person_and_organization_assignment <= person_and_organization_assignment person_and_organization_assignment.\ assigned_person_and_organization -> person_and_organization person_and_organization.the_organization -> organization) (product_definition_formation_with_specified_source <= product_definition_formation organization_item = product_definition_formation organization_item <- applied_organization_assignment.items[i] applied_organization_assignment <= organization_assignment organization_assignment.assigned_organization -> organization)

Table 4 - Mapping table for assembly UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
SUPPLIER	organization	41	20	
supplier_identification	organization.id	41		
supplier TO person_organization FROM authorization UoF (as identified_as)	PATH			(organization <- person_and_organization.the_organization person_and_organization <- person_and_organization_assignment.\ assigned_person_and_organization person_and_organization_assignment => applied_person_and_organization_assignment) (organization <- organization_assignment.assigned_organization organization_assignment => applied_organization_assignment)

Table 5 - Mapping table for authorization UoF

Application element	AIM element	Source	Rules	Reference Path
APPROVAL	applied_approval_assignment	209	4, 5, 6	
purpose	approval.level	41		applied_approval_assignment <= approval_assignment approval_assignment.assigned_approval -> approval approval.level
status	approval_status.name	41	11, 39	applied_approval_assignment <= approval_assignment approval_assignment.assigned_approval -> approval approval.status -> approval_status approval_status.name
approval TO (ONE OR MANY UNIQUE) person_organization (as authorized_by)	PATH		5	applied_approval_assignment <= approval_assignment approval_assignment.assigned_approval -> approval approval.authorized_approval -> approval_person_organization approval_person_organization.person_organization -> person_organization_select (person_organization_select = person_and_organization person_and_organization <- person_and_organization_assignment.\ assigned_person_and_organization person_and_organization_assignment => applied_person_and_organization_assignment) (person_organization_select = organization organization <- organization_assignment.assigned_organization organization_assignment => applied_organization_assignment)

Table 5 - Mapping table for authorization UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
approval TO date (as effective_date)	PATH		4	applied_approval_assignment <= approval_assignment approval_assignment.assigned_approval -> approval <- approval_date_time.dated_approval approval_date_time approval_date_time.date_time -> date_time_select (date_time_select = date_and_time date_and_time) (date_time_select = date date)
DATE	(date_and_time) (date)	41 41	14, 15, 40	
PERSON_ORGANIZATION	(applied_person_and_\ organization_assignment) ([person] [applied_organization_assignment])	209 41 209	20, 22, 41	

Table 5 - Mapping table for authorization UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
address #1: Identify address of the person #2: Identify the address of the organization	#1 (address) #2 (address)	41 41		(applied_person_and_organization_assignment <= person_and_organization_assignment person_and_organization_assignment.\ assigned_person_and_organization -> person_and_organization #1 (person_and_organization.the_person -> person <- personal_address.people[i] personal_address <= address) #2 (person_and_organization.the_organization -> organization <- organizational_address.organizations[i] organizational_address <= address)) (#1 ([person <- personal_address.people[i] personal_address <= address] [applied_organization_assignment]) #2 ([person] [applied_organization_assignment <= organization_assignment organization_assignment.assigned_organization -> organization <- organizational_address.organizations[i] organizational_address <= address]))

Table 5 - Mapping table for authorization UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
organization	organization	41		(applied_person_and_organization_assignment <= person_and_organization_assignment person_and_organization_assignment.\ assigned_person_and_organization -> person_and_organization person_and_organization.the_organization -> organization) ([person] [applied_organization_assignment <= organization_assignment organization_assignment.assigned_organization -> organization])
person	person	41		(applied_person_and_organization_assignment <= person_and_organization_assignment person_and_organization_assignment.\ assigned_person_and_organization -> person_and_organization person_and_organization.the_person -> person) ([person] [applied_organization_assignment])

Table 5 - Mapping table for authorization UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
person_organization_identification #1: Identify the person #2: Identify the organization	#1 (person.id) #2 (organization.id)	41 41		(applied_person_and_organization_assignment <= person_and_organization_assignment person_and_organization_assignment.\ assigned_person_and_organization -> person_and_organization #1 (person_and_organization.the_person -> person person.id) #2 (person_and_organization.the_organization -> organization organization.id)) (#1 ([person person.id] [applied_organization_assignment]) #2 ([person] [applied_organization_assignment <= organization_assignment organization_assignment.assigned_organization -> organization organization.id]))

Table 6 - Mapping table for composite_constituent_representation UoF

Application element	AIM element	Source	Rules	Reference Path
ANGLE_MEASURE	plane_angle_measure_with_unit	41		
BEVELED_SHEET_REPRESENTATION beveled_sheet_representation <= constituent_shape_representation	beveled_sheet_representation	209	47	beveled_sheet_representation <= shape_representation
beveled_sheet_representation TO angle_measure (as bevel_angle)	PATH			beveled_sheet_representation <= shape_representation <= representation representation.items[1] -> representation_item => measure_representation_item <= measure_with_unit => plane_angle_measure_with_unit
beveled_sheet_representation TO length_measure (as vertical_profile_height)	PATH		19	beveled_sheet_representation <= shape_representation <= representation representation.items[2] -> representation_item => measure_representation_item <= measure_with_unit {measure_with_unit measure_with_unit.value_component -> measure_value measure_value = positive_length_measure} {measure_with_unit measure_with_unit.unit_component -> unit unit = named_unit named_unit => length_unit} measure_with_unit => length_measure_with_unit

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
beveled_sheet_representation TO composite_sheet_representation (as boundary_surface)	PATH			beveled_sheet_representation <= shape_representation <= representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation => shape_representation => composite_sheet_representation
BOUNDARY_CURVE_REPRESENTATION	geometric_set	42		
boundary_curve_representation TO (ONE OR MANY UNIQUE) curve (as bounds) #1: represent boundary as a parametric curve #2: represent boundary as a curve on a surface #3: represent boundary as a composite curve on a surface	PATH			geometric_set geometric_set[i].elements -> geometric_set_select = curve #1 (curve => pcurve) #2 (curve => surface_curve) #3 (curve => bounded_curve => composite_curve => composite_curve_on_surface)

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
COMPOSITE_SHEET_REPRESENTATION composite_sheet_representation <= constituent_shape_representation	composite_sheet_representation	209	47	composite_sheet_representation <= shape_representation
CONSTITUENT_SHAPE_REPRESENTATION constituent_shape_representation <= geometric_model_representation	shape_representation	41	47	shape_representation
CURVE #1: as a parametric curve #2: as curve on a surface #3: as a composite curve on a surface	#1 (pcurve) #2 (surface_curve) #3 (composite_curve_on_surface)	42 42 42		
DIRECTION	direction	42		

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
FACE_BASED_SHEET_REPRESENTATION face_based_sheet_representation <= composite_sheet_representation	manifold_surface_shape_\ representation	509	47	[manifold_surface_shape_representation <= shape_representation] [shape_representation => composite_sheet_representation]
face_based_sheet_representation TO surface (as face_surface)	PATH			manifold_surface_shape_representation <= shape_representation <= representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation {representation => shape_representation} {representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition => property_definition property_definition.definition -> characterized_definition characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect.name = 'face surface'} representation representation.items[i] -> representation_item geometric_representation_item => surface

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
face_based_sheet_representation TO (ZERO, ONE OR MANY UNIQUE) loop (as inner_bounds)	PATH			manifold_surface_shape_representation <= shape_representation <= representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation {representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition => property_definition property_definition.definition -> characterized_definition characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect.name = 'inner bounds'} {representation => shape_representation} representation representation.items[i] -> representation_item topological_representation_item => loop

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
face_based_sheet_representation TO loop (as outer_bound)	PATH			manifold_surface_shape_representation <= shape_representation <= representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation {representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition => property_definition property_definition.definition -> characterized_definition characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect.name = 'outer bound'} {representation => shape_representation} representation representation.items[i] -> representation_item topological_representation_item => loop

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
FILAMENT_LAMINATE_SHAPE	shape_aspect	41		shape_aspect
filament_laminate_shape TO idealized_analysis_shape (as cross_section)	PATH			shape_aspect {shape_aspect.name = \ 'filament laminate idealized analysis shape'} shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> shape_representation <= representation representation_relationship.rep_1 representation_relationship representation_relationship.rep_2 -> representation => shape_representation
filament_laminate_shape TO nominal_design_shape (as cross_section)	PATH			shape_aspect {shape_aspect.name = \ 'filament laminate nominal design shape'} shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> shape_representation <= representation representation_relationship.rep_1 representation_relationship representation_relationship.rep_2 -> representation => shape_representation

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
FLAT_PATTERN_PLY_SHAPE flat_pattern_ply_shape <= ply_shape	shape_aspect	41		shape_aspect shape_aspect.name = 'flat pattern ply shape'
flat_pattern_ply_shape TO location (as wrapup_origin_on_flat_pattern)	PATH			shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation {representation => shape_representation} representation <- representation_relationship.rep_2 representation_relationship {representation_relationship => shape_representation_relationship => flat_pattern_ply_representation_relationship} representation_relationship => representation_relationship_with_transformation representation_relationship_with_\ transformation.transformation_operator -> transformation = item_defined_transformation item_defined_transformation.transformation_item_2 -> representation_item => geometric_representation_item => placement placement.location -> cartesian_point

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
flat_pattern_ply_shape TO location (as wrapup_origin_on_surface)	PATH			shape_aspect shape_definition=shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation {representation => shape_representation} {representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship => flat_pattern_ply_representation_relationship} representation_relationship=> representation_relationship_with_transformation representation_relationship_with_\ transformation.transformation_operator-> transformation=item_defined_transformation item_defined_transformation.transformation_item_1-> representation_item => geometric_representation_item => placement placement.location -> cartesian_point
GEOMETRIC_MODEL_REPRESENTATION	shape_representation	41	21, 23, 25, 26, 47	
role	representation.name	43		shape_representation <= representation representation.name

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
geometric_model_representation TO (ONE OR MANY UNIQUE) geometry_element (as elements)	PATH			shape_representation <= representation representation.items[i] -> representation_item
GEOMETRIC_SHEET_REPRESENTATION geometric_sheet_representation <= composite_sheet_representation	geometrically_bounded_surface_\ shape_representation	507	47	[geometrically_bounded_surface_shape_representation <= shape_representation] [shape_representation => composite_sheet_representation]
geometric_sheet_representation TO surface (as basis_surface)	PATH			geometrically_bounded_surface_shape_representation <= shape_representation <= representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation {representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition => property_definition property_definition.definition -> characterized_definition characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect.name = 'basis surface'} {representation => shape_representation} representation representation.items[i] -> representation_item geometric_representation_item => surface

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
geometric_sheet_representation TO (ZERO, ONE OR MANY UNIQUE) boundary_curve_representation (as cutouts)	PATH			geometrically_bounded_surface_shape_representation <= shape_representation <= representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation {representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition => property_definition property_definition.definition -> characterized_definition characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect.name = 'cutouts'} {representation => shape_representation} representation representation.items[i] -> representation_item geometric_representation_item => geometric_set

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
geometric_sheet_representation TO boundary_curve_representation (as outer_edge)	PATH			geometrically_bounded_surface_shape_representation <= shape_representation <= representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation {representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition => property_definition property_definition.definition -> characterized_definition characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect.name = 'outer edge'} {representation => shape_representation} representation representation.items[i] -> representation_item geometric_representation_item => geometric_set
GEOMETRY_ELEMENT	geometric_representation_item	43		

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
LAID_PLY_SHAPE laid_ply_shape <= ply_shape	shape_aspect	41		shape_aspect shape_aspect.name = 'laid ply shape'
LENGTH_MEASURE	length_measure_with_unit	41		
LOCATION	cartesian_point	42		
LOOP	loop	42		
PLY_SHAPE	shape_aspect	41		
ply_shape TO (OPTIONAL) ply_shape (as basis)	PATH			shape_aspect <- shape_aspect_relationship.related_shape_aspect shape_aspect_relationship shape_aspect_relationship.relating_shape_aspect -> shape_aspect
ply_shape TO composite_sheet_representation (as defining_model)	PATH			shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation property_definition_representation.used_representation -> representation => shape_representation => composite_sheet_representation

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
PROJECTED_PLY_SHAPE projected_ply_shape <= ply_shape	shape_aspect	41		
method	shape_aspect.name	41		
projected_ply_shape TO (OPTIONAL) direction (as projection_direction)	PATH			shape_aspect shape_aspect.of_shape -> product_definition_shape <- shape_aspect.of_shape shape_aspect {shape_aspect.name = 'projection direction'} shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation {representation => shape_representation} representation.items[i] -> geometric_representation_item => placement
SURFACE	surface	42		
SURFACE_PLY_SHAPE surface_ply_shape <= projected_ply_shape #1: projection method is reference direction #2: projection method is surface normal	shape_aspect	41		#1 (shape_aspect shape_aspect.name = \\ 'reference direction projected surface ply shape') #2 (shape_aspect shape_aspect.name = \\ 'surface normal projected surface ply shape')
surface_role	shape_aspect.description	41		

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
surface_ply_shape TO length_measure (as offset)	PATH		19	shape_aspect shape_aspect.of_shape -> product_definition_shape <- shape_aspect.of_shape shape_aspect {shape_aspect.name = 'offset'} shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation {representation => shape_representation representation.items[i] -> representation_item => measure_representation_item <= measure_with_unit {measure_with_unit measure_with_unit.value_component -> measure_value measure_value = positive_length_measure} {measure_with_unit measure_with_unit.unit_component -> unit unit = named_unit named_unit => length_unit} measure_with_unit => length_measure_with_unit

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
THREE_D_GEOMETRY_SET three_d_geometry_set <= constituent_shape_representation	geometrically_bounded_surface_\ shape_representation	507	47	geometrically_bounded_surface_shape_representation <= shape_representation
basis_role	shape_aspect.description	41		geometrically_bounded_surface_shape_representation <= shape_representation <= representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition -> property_definition property_definition.definition -> characterized_definition characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect shape_aspect.description

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
three_d_geometry_set TO surface (as basis_surface)	PATH			geometrically_bounded_surface_shape_representation <= shape_representation <= representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation {representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition => property_definition property_definition.definition -> characterized_definition characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect.name = 'basis surface'} {representation => shape_representation} representation representation.items[i] -> representation_item geometric_representation_item => surface

Table 6 - Mapping table for composite_constituent_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
three_d_geometry_set TO (ONE OR MANY UNIQUE) curve (as defining_boundary) #1: to a parametric curve #2: to a surface curve #3: to a composite curve on surface	PATH			geometrically_bounded_surface_shape_representation <= shape_representation <= representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation {representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition => property_definition property_definition.definition -> characterized_definition characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect.name = 'defining boundary'} {representation => shape_representation} representation representation.items[i] -> representation_item => geometric_representation_item => geometric_set geometric_set.elements -> geometric_set_select geometric_set_select = curve (*continued on the following page*)

Table 6 - Mapping table for composite_constituent_representation UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
three_d_geometry_set TO (ONE OR MANY UNIQUE) curve (as defining_boundary) (*continued*)	PATH			#1 (curve => pcurve) #2 (curve => surface_curve) #3 (curve => bounded_curve => composite_curve => composite_curve_on_surface)
VIEW_PLY_SHAPE view_ply_shape <= projected_ply_shape #1: projection method is reference direction #2: projection method is surface normal	shape_aspect	41		#1 (shape_aspect shape_aspect.name = \ 'reference direction projected view ply shape') #2 (shape_aspect shape_aspect.name = \ 'surface normal projected view ply shape')

Table 7 - Mapping table for effectivity UoF

Application element	AIM element	Source	Rules	Reference Path
DATE	(date_and_time) (date)	41 41	14, 15, 40	
DATE_EFFECTIVITY date_effectivity <= effectivity	dated_effectivity	41		dated_effectivity <= effectivity => product_definition_effectivity => configuration_effectivity
date_effectivity TO (OPTIONAL) date (as end_date)	PATH		16, 17, 40	dated_effectivity dated_effectivity.effectivity_end_date -> date_and_time
date_effectivity TO date (as start_date)	PATH		16, 17, 40	dated_effectivity dated_effectivity.effectivity_start_date -> date_and_time
EFFECTIVITY	effectivity	41		
effectivity TO (ONE OR MANY UNIQUE) assembly FROM assembly UoF (as affected_assemblies)	PATH			effectivity => product_definition_effectivity product_definition_effectivity.usage -> product_definition_relationship product_definition_usage => assembly_component_usage
effectivity TO (OPTIONAL) approval FROM authorization UoF (as approved_by)	PATH			effectivity => product_definition_effectivity => configuration_effectivity approval_item = configuration_effectivity approval_item <- applied_approval_assignment.items[i] applied_approval_assignment
effectivity TO product_configuration FROM end_item_identification UoF (as configuration_item)	PATH			effectivity => product_definition_effectivity => configuration_effectivity configuration_effectivity.configuration -> configuration_design configuration_design.configuration -> configuration_item

Table 7 - Mapping table for effectivity UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
LOT_EFFECTIVITY lot_effectivity <= effectivity	lot_effectivity	41		lot_effectivity <= effectivity => product_definition_effectivity => configuration_effectivity
lot_number	lot_effectivity.effectivity_lot_id	41		
lot_size	measure_with_unit.\ value_component	41		lot_effectivity lot_effectivity.effectivity_lot_size -> measure_with_unit measure_with_unit.value_component
lot_size_unit_of_measure	measure_with_unit.\ unit_component	41	19	lot_effectivity lot_effectivity.effectivity_lot_size -> measure_with_unit measure_with_unit.unit_component
RETENTION_PERIOD	retention	209		retention <= action {action <- action_assignment.assigned_action action_assignment -> retention_assignment}
retention_purpose	action.description	41		retention <= action action.description
retention_period TO (OPTIONAL) approval FROM authorization UoF (as approved_by)	PATH			retention approval_item = retention approval_item <- applied_approval_assignment.items[i] applied_approval_assignment

Table 7 - Mapping table for effectivity UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
retention_period TO date (as earliest_end_definition)	action.description	41	16, 17, 40	(retention date_time_item = retention date_time_item <- applied_date_and_time_assignment.items[i] applied_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment date_and_time_assignment.role -> date_time_role date_time_role.name = 'earliest end date'} date_and_time_assignment.assigned_date_and_time -> date_and_time) (retention date_item = retention date_item <- applied_date_assignment.items[i] applied_date_assignment <= date_assignment {date_assignment date_assignment.role -> date_role date_role.name = 'earliest end date'} date_assignment.assigned_date -> date)
retention_period TO (ONE OR MANY UNIQUE) additional_design_information (as is_applied_to)	PATH			retention <= action <- action_assignment.assigned_action action_assignment => retention_assignment retention_assignment.items[i] -> retention_item retention_item = document document

Table 7 - Mapping table for effectivity UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
retention_period TO (ONE OR MANY UNIQUE) analysis (as is_applied_to)	PATH			retention<= action<- action_assignment.assigned_action action_assignement=> retention_assignment rentention_assignment.items[i]-> retention_item rentention_item=product_definition product_definition
retention_period TO (ONE OR MANY UNIQUE) analysis_discipline_product_definition (as is_applied_to)	PATH			retention <= action<- action_assignment.assigned_action action_assignment=> retention_assignment retention_assignment.items[i]-> retention_item retention_item=product_definition product_definition
retention_period TO (ONE OR MANY UNIQUE) analysis_report_representation (as is_applied_to)	PATH			retention <= action <- action_assignment.assigned_action action_assignment => retention_assignment retention_assignment.items[i] -> retention_item retention_item = document document
retention_period TO (ONE OR MANY UNIQUE) analysis_version (as is_applied_to)	PATH			retention <= action <- action_assignment.assigned_action action_assignment => retention_assignment retention_assignment.items[i] -> retention_item retention_item = product_definition_formation product_definition_formation

Table 7 - Mapping table for effectivity UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
retention_period TO (ONE OR MANY UNIQUE) assembly (as is_applied_to)	PATH			retention <= action <- action_assignment.assigned_action action_assignment => retention_assignment retention_assignment.items[i] -> retention_item retention_item = product_definition_relationship product_definition_relationship => product_definition_usage => assembly_component_usage
retention_period TO (ONE OR MANY UNIQUE) design_discipline_product_definition (as is_applied_to)	PATH			retention <= action <- action_assignment.assigned_action action_assignment => retention_assignment retention_assignment.items[i] -> retention_item retention_item = product_definition product_definition
retention_period TO (ONE OR MANY UNIQUE) design_material (as is_applied_to)	PATH			retention <= action <- action_assignment.assigned_action action_assignment => retention_assignment retention_assignment.items[i] -> retention_item retention_item = product_definition_relationship product_definition_relationship => product_definition_usage => make_from_usage_option

Table 7 - Mapping table for effectivity UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
retention_period TO (ONE OR MANY UNIQUE) fe_analysis (as is_applied_to)	PATH			retention <= action <- action_assignment.assigned_action action_assignment => retention_assignment retention_assignment.items[i] -> retention_item retention_item = control control
retention_period TO (ONE OR MANY UNIQUE) fe_analysis_results (as is_applied_to)	PATH			retention <= action <- action_assignment.assigned_action action_assignment => retention_assignment retention_assignment.items[i] -> retention_item retention_item = result result
retention_period TO (ONE OR MANY UNIQUE) fea_model (as is_applied_to)	PATH			retention <= action <- action_assignment.assigned_action action_assignment => retention_assignment retention_assignment.items[i] -> retention_item retention_item = representation representation => fea_model
retention_period TO (ONE OR MANY UNIQUE) material_property (as is_applied_to)	PATH			retention <= action <- action_assignment.assigned_action action_assignment => retention_assignment retention_assignment.items[i] -> retention_item retention_item = material_property material_property

Table 7 - Mapping table for effectivity UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
retention_period TO (ONE OR MANY UNIQUE) material_specification from material UoF (as is_applied_to)	PATH			retention <= action <- action_assignment.assigned_action action_assignment => retention_assignment retention_assignment.items[i] -> retention_item retention_item = document document
retention_period TO (ONE OR MANY UNIQUE) part (as is_applied_to)	PATH			retention <= action <- action_assignment.assigned_action action_assignment => retention_assignment retention_assignment.items[i] -> retention_item retention_item = product product
retention_period TO (ONE OR MANY UNIQUE) part_version (as is_applied_to)	PATH			retention <= action <- action_assignment.assigned_action action_assignment => retention_assignment retention_assignment.items[i] -> retention_item retention_item = product_definition_formation product_definition_formation

Table 7 - Mapping table for effectivity UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
retention_period TO date (as latest_end_definition)	action.description	41	16, 17, 40	(retention date_time_item = retention date_time_item <- applied_date_and_time_assignment.items[i] applied_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment date_and_time_assignment.role -> date_time_role date_time_role.name = 'latest end date'} date_and_time_assignment.assigned_date_and_time -> date_and_time) (retention date_item = retention date_item <- applied_date_assignment.items[i] applied_date_assignment <= date_assignment {date_assignment date_assignment.role -> date_role date_role.name = 'latest end date'} date_assignment.assigned_date -> date)

Table 7 - Mapping table for effectivity UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
retention_period TO date (as start_definition)	action.description	41	16, 17, 40	(retention date_time_item = retention date_time_item <- applied_date_and_time_assignment.items[i] applied_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment date_and_time_assignment.role -> date_time_role date_time_role.name = 'start date'} date_and_time_assignment.assigned_date_and_time -> date_and_time) (retention date_item = retention date_item <- applied_date_assignment.items[i] applied_date_assignment <= date_assignment {date_assignment date_assignment.role -> date_role date_role.name = 'start date'} date_assignment.assigned_date -> date)

Table 7 - Mapping table for effectivity UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
SEQUENCE_EFFECTIVITY sequence_effectivity <= effectivity	serial_numbered_effectivity	41		serial_numbered_effectivity <= effectivity => product_definition_effectivity => configuration_effectivity
component_quantity	measure_with_unit.\ value_component	41	7	serial_numbered_effectivity <= effectivity => product_definition_effectivity product_definition_effectivity.usage -> product_definition_relationship => product_definition_usage => assembly_component_usage => quantified_assembly_component_usage => product_definition_usage => assembly_component_usage => quantified_assembly_component_usage.quantity -> measure_with_unit measure_with_unit.value_component
from_effectivity_identification	serial_numbered_effectivity.\ effectivity_start_id	41		
quantity_unit_of_measure	measure_with_unit.\ unit_component	41	19	serial_numbered_effectivity <= effectivity => product_definition_effectivity product_definition_effectivity.usage -> product_definition_relationship => product_definition_usage => assembly_component_usage => quantified_assembly_component_usage => product_definition_usage => assembly_component_usage => quantified_assembly_component_usage.quantity -> measure_with_unit measure_with_unit.unit_component
to_effectivity_identification	serial_numbered_effectivity.\ effectivity_end_id	41		

Table 8 - Mapping table for end_item_identification UoF

Application element	AIM element	Source	Rules	Reference Path
PRODUCT_CONFIGURATION	configuration_item	44	9, 30	
item_identification (UNIQUE)	configuration_item.id	44		
phase_of_product	configuration_item.purpose	44		
product_configuration TO product_model (as model_name)	PATH		30	configuration_item configuration_item.item_concept -> product_concept
product_configuration TO (OPTIONAL) approval FROM authorization UoF (as approved_by)	PATH			configuration_item approval_item = configuration_item approval_item <- applied_approval_assignment.items[i] applied_approval_assignment
product_configuration TO (ZERO, ONE OR MANY UNIQUE) part FROM part_identification UoF (as parts_configured)	PATH			configuration_item <- configuration_design.configuration configuration_design configuration_design.design -> product_definition_formation product_definition_formation.of_product -> product
PRODUCT_MODEL	product_concept	41	30	
model_name	product_concept.name	41		

Table 9 - Mapping table for faceted_boundary_representation UoF

Application element	AIM element	Source	Rules	Reference Path
FACETED_BOUNDARY_REPRESENTATION faceted_boundary_representation <= geometric_model_representation	faceted_brep_shape_representation	512	47	faceted_brep_shape_representation <= shape_representation
GEOMETRIC_MODEL_REPRESENTATION	shape_representation	41	21, 23, 25, 26, 47	
role	representation.name	43		shape_representation <= representation representation.name
geometric_model_representation TO (ONE OR MANY UNIQUE) geometry_element (as elements)	PATH			shape_representation <= representation representation.items -> representation_item => geometric_representation_item
GEOMETRY_ELEMENT	geometric_representation_item	43		

Table 10 - Mapping table for fe_analysis_control UoF

Application element	AIM element	Source	Rules	Reference Path
ANALYSIS_MESSAGE analysis_message <= fe_analysis_state_definition	analysis_message	104		analysis_message <= state_definition
message_level	analysis_message.message_level	104		
quality	analysis_message.message_text	104		
CALCULATED_STATE calculated_state <= fe_analysis_state	calculated_state	104		calculated_state <= state
CONSTRAINT	constraint_element	104		
identification	constraint_element.element_id	104		
constraint TO (ONE OR MANY UNIQUE) fe_analysis_control_step (as steps)	PATH			constraint_element constraint_element.steps[i] -> control_analysis_step
ELEMENT_FIELD_VARIABLE_DEFINITION element_field_variable_definition <= fe_analysis_state_definition	field_variable_definition	104		field_variable_definition <= state_definition
ELEMENT_NODAL_FREEDOM_ACTIONS element_nodal_freedom_actions <= fe_analysis_state_definition	element_nodal_freedom_actions	104		element_nodal_freedom_actions <= state_definition
FE_ANALYSIS	control	104		
intended_analysis_code	control.intended_analysis_code[i]	104		
fe_analysis TO (OPTIONAL) approval FROM authorization UoF (as approved_by)	PATH			control approval_item = control approval_item <- applied_approval_assignment.items[i] applied_approval_assignment
fe_analysis TO fea_model (as model_ref)	PATH			control control.model_ref -> fea_model

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
FE_ANALYSIS_CONTROL_STEP	control_analysis_step	104		
identification	control_analysis_step.step_id	104		
sequence	control_analysis_step.sequence	104		
fe_analysis_control_step TO fe_analysis (as analysis)	PATH			control_analysis_step <= analysis_step analysis_step.analysis_control -> control
fe_analysis_control_step TO fe_analysis_state (as initial_input_state) #1: as a linear static analysis step #2: as a linear modes and frequencies analysis step	PATH			#1 (control_analysis_step [control_analysis_step.initial_state -> state] [control_analysis_step => control_linear_static_analysis_step control_linear_static_analysis_step.process -> control_linear_static_load_increment_process control_linear_static_load_increment_process.\ final_input_state -> state]) #2 (control_analysis_step [control_analysis_step.initial_state -> state] [control_analysis_step => control_linear_modes_and_frequencies_analysis_step control_linear_modes_and_frequencies_analysis_step.\ process -> control_linear_modes_and_frequencies_process control_linear_modes_and_frequencies_process.\ final_input_state -> state])
FE_ANALYSIS_STATE	state	104		
fe_analysis_state TO (ONE OR MANY UNIQUE) fe_analysis_state_definition (as definitions)	PATH			state <- state_definition.defined_state state_definition
FE_ANALYSIS_STATE_DEFINITION	state_definition	104		

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO element FROM fea_model UoF (as ref) #1: to an element #2: to a 2d axisymmetric curve element #3: to a 2d plane curve element #4: to a 3d curve element #5: to a 2d axisymmetric surface element #6: to a 2d plane surface element #7: to a 3d surface element #8: to a 2d axisymmetric volume element #9: to a 2d plane volume element #10: to a 3d volume element	PATH			state_definition => #1 ((element_nodal_freedom_actions element_nodal_freedom_actions.element -> model_or_control_element model_or_control_element = element_representation) (analysis_message => element_analysis_message element_analysis_message.element ->) element_representation) #2 (field_variable_definition => field_variable_element_definition => curve_2d_element_field_variable_definition curve_2d_element_field_variable_definition.element -> curve_2d_element_output_reference (curve_2d_element_output_reference = \ curve_2d_element_representation curve_2d_element_representation = \ axisymmetric_curve_2d_element_representation axisymmetric_curve_2d_element_representation) (curve_2d_element_output_reference = \ curve_2d_element_descriptor curve_2d_element_descriptor = \ axisymmetric_curve_2d_element_descriptor axisymmetric_curve_2d_element_descriptor <- axisymmetric_curve_2d_element_representation.\ element_descriptor axisymmetric_curve_2d_element_representation) (curve_2d_element_output_reference = \ curve_2d_substructure_element_reference curve_2d_substructure_element_reference curve_2d_substructure_element_reference.element_ref -> curve_2d_element_representation curve_2d_element_representation = \ axisymmetric_curve_2d_element_representation axisymmetric_curve_2d_element_representation)) *(continued on the following page)*

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO element FROM fea_model UoF (as ref) (*continued*)	PATH			#3 (field_variable_definition => field_variable_element_definition => curve_2d_element_field_variable_definition curve_2d_element_field_variable_definition.element -> curve_2d_element_output_reference (curve_2d_element_output_reference = \\ curve_2d_element_representation curve_2d_element_representation = \\ plane_curve_2d_element_representation plane_curve_2d_element_representation) (curve_2d_element_output_reference = \\ curve_2d_element_descriptor curve_2d_element_descriptor = \\ plane_curve_2d_element_descriptor plane_curve_2d_element_descriptor <- plane_curve_2d_element_representation.element_descriptor plane_curve_2d_element_representation) (curve_2d_element_output_reference = \\ curve_2d_substructure_element_reference curve_2d_substructure_element_reference curve_2d_substructure_element_reference.element_ref -> curve_2d_element_representation curve_2d_element_representation = \\ plane_curve_2d_element_representation plane_curve_2d_element_representation)) #4 (field_variable_definition => field_variable_element_definition => curve_3d_element_field_variable_definition curve_3d_element_field_variable_definition.element -> curve_3d_element_output_reference

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO element FROM fea_model UoF (as ref) (*continued*)				(curve_3d_element_output_reference = \\ curve_3d_element_representation curve_3d_element_representation) (curve_3d_element_output_reference = \\ curve_3d_element_descriptor curve_3d_element_descriptor <- curve_3d_element_representation.element_descriptor curve_3d_element_representation) (curve_3d_element_output_reference = \\ curve_3d_substructure_element_reference curve_3d_substructure_element_reference curve_3d_substructure_element_reference.element_ref -> curve_3d_element_representation)) #5 (field_variable_definition => field_variable_element_definition => surface_2d_element_field_variable_definition surface_2d_element_field_variable_definition.element -> surface_2d_element_output_reference (surface_2d_element_output_reference = \\ surface_2d_element_representation surface_2d_element_representation = \\ axisymmetric_surface_2d_element_representation axisymmetric_surface_2d_element_representation) (surface_2d_element_output_reference = \\ surface_2d_element_descriptor surface_2d_element_descriptor = \\ axisymmetric_surface_2d_element_descriptor axisymmetric_surface_2d_element_descriptor <- axisymmetric_surface_2d_element_representation.\\ element_descriptor axisymmetric_surface_2d_element_representation)

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO element FROM fea_model UoF (as ref) (*continued*)				(surface_2d_element_output_reference = \\ surface_2d_substructure_element_reference surface_2d_substructure_element_reference surface_2d_substructure_element_reference.element_ref -> surface_2d_element_representation surface_2d_element_representation = \\ axisymmetric_surface_2d_element_representation axisymmetric_surface_2d_element_representation)) #6 (field_variable_definition => field_variable_element_definition => surface_2d_element_field_variable_definition surface_2d_element_field_variable_definition.element -> surface_2d_element_output_reference (surface_2d_element_output_reference = \\ surface_2d_element_representation surface_2d_element_representation = \\ plane_surface_2d_element_representation plane_surface_2d_element_representation) (surface_2d_element_output_reference = \\ surface_2d_element_descriptor surface_2d_element_descriptor = \\ plane_surface_2d_element_descriptor plane_surface_2d_element_descriptor <- plane_surface_2d_element_representation.element_descriptor plane_surface_2d_element_representation) (surface_2d_element_output_reference = \\ surface_2d_substructure_element_reference surface_2d_substructure_element_reference surface_2d_substructure_element_reference.element_ref -> surface_2d_element_representation surface_2d_element_representation = \\ plane_surface_2d_element_representation plane_surface_2d_element_representation))

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO element FROM fea_model UoF (as ref) (*continued*)				#7 (field_variable_definition => field_variable_element_definition => surface_3d_element_field_variable_definition surface_3d_element_field_variable_definition.element -> surface_3d_element_output_reference (surface_3d_element_output_reference = \\ surface_3d_element_representation surface_3d_element_representation) (surface_3d_element_output_reference = \\ surface_3d_element_descriptor surface_3d_element_descriptor <- surface_3d_element_representation.element_descriptor surface_3d_element_representation) (surface_3d_element_output_reference = \\ surface_3d_substructure_element_reference surface_3d_substructure_element_reference surface_3d_substructure_element_reference.element_ref -> surface_3d_element_representation)) #8 (field_variable_definition => field_variable_element_definition => volume_2d_element_field_variable_definition volume_2d_element_field_variable_definition.element -> volume_2d_element_output_reference (volume_2d_element_output_reference = \\ volume_2d_element_representation volume_2d_element_representation = \\ axisymmetric_volume_2d_element_representation axisymmetric_volume_2d_element_representation)

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO element FROM fea_model UoF (as ref) (*continued*)				(volume_2d_element_output_reference = \\ volume_2d_element_descriptor volume_2d_element_descriptor = \\ axisymmetric_volume_2d_element_descriptor axisymmetric_volume_2d_element_descriptor <- axisymmetric_volume_2d_element_representation.\\ element_descriptor axisymmetric_volume_2d_element_representation) (volume_2d_element_output_reference = \\ volume_2d_substructure_element_reference volume_2d_substructure_element_reference volume_2d_substructure_element_reference.element_ref -> volume_2d_element_representation volume_2d_element_representation = \\ axisymmetric_volume_2d_element_representation axisymmetric_volume_2d_element_representation)) #9 (field_variable_definition => field_variable_element_definition => volume_2d_element_field_variable_definition volume_2d_element_field_variable_definition.element -> volume_2d_element_output_reference (volume_2d_element_output_reference = \\ volume_2d_element_representation volume_2d_element_representation = \\ plane_volume_2d_element_representation plane_volume_2d_element_representation) (volume_2d_element_output_reference = \\ volume_2d_element_descriptor volume_2d_element_descriptor = \\ plane_volume_2d_element_descriptor plane_volume_2d_element_descriptor <- plane_volume_2d_element_representation.element_descriptor plane_volume_2d_element_representation)

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO element FROM fea_model UoF (as ref) (*continued*)				(volume_2d_element_output_reference = \ volume_2d_substructure_element_reference volume_2d_substructure_element_reference volume_2d_substructure_element_reference.element_ref -> volume_2d_element_representation volume_2d_element_representation = \ plane_volume_2d_element_representation plane_volume_2d_element_representation)) #10 (field_variable_definition => field_variable_element_definition => volume_3d_element_field_variable_definition volume_3d_element_field_variable_definition.element -> volume_3d_element_output_reference (volume_3d_element_output_reference = \ volume_3d_element_representation volume_3d_element_representation) (volume_3d_element_output_reference = \ volume_3d_element_descriptor volume_3d_element_descriptor <- volume_3d_element_representation.element_descriptor volume_3d_element_representation) (volume_3d_element_output_reference = \ volume_3d_substructure_element_reference volume_3d_substructure_element_reference volume_3d_substructure_element_reference.element_ref -> volume_3d_element_representation))
fe_analysis_state_definition TO node FROM fea_model UoF (as ref)	PATH			state_definition => (nodal_freedom_and_value_definition nodal_freedom_and_value_definition.node -> node_output_reference node_output_reference = node) (analysis_message => element_analysis_message element_analysis_message.node ->) node

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO group FROM fea_model UoF (as ref) #1: to a group of 2d curve elements #2: to a group of 2d surface elements #3: to a group of 2d volume elements #4: to a group of 3d curve elements #5: to a group of 3d surface elements #6: to a group of 3d volume elements #7: to a group of elements #8: to a group of one node and a group of elements #9: to a group of nodes #10: to a group that is the whole model	PATH			state_definition => #1 (field_variable_definition => field_variable_element_definition => curve_2d_element_field_variable_definition curve_2d_element_field_variable_definition.element -> curve_2d_element_output_reference curve_2d_element_output_reference = \ curve_2d_element_group curve_2d_element_group) #2 (field_variable_definition => field_variable_element_definition => surface_2d_element_field_variable_definition surface_2d_element_field_variable_definition.element -> surface_2d_element_output_reference surface_2d_element_output_reference = \ surface_2d_element_group surface_2d_element_group) #3 (field_variable_definition => field_variable_element_definition => volume_2d_element_field_variable_definition volume_2d_element_field_variable_definition.element -> volume_2d_element_output_reference volume_2d_element_output_reference = \ volume_2d_element_group volume_2d_element_group)

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO group FROM fea_model UoF (as ref) (*continued*)				#4 (field_variable_definition => field_variable_element_definition => curve_3d_element_field_variable_definition curve_3d_element_field_variable_definition.element -> curve_3d_element_output_reference curve_3d_element_output_reference =\ curve_3d_element_group curve_3d_element_group) #5 (field_variable_definition => field_variable_element_definition => surface_3d_element_field_variable_definition surface_3d_element_field_variable_definition.element -> surface_3d_element_output_reference surface_3d_element_output_reference = \ surface_3d_element_group surface_3d_element_group) #6 (field_variable_definition => field_variable_element_definition => volume_3d_element_field_variable_definition volume_3d_element_field_variable_definition.element -> volume_3d_element_output_reference volume_3d_element_output_reference = \ volume_3d_element_group volume_3d_element_group)

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO group FROM fea_model UoF (as ref) (*continued*)				#7 ((field_variable_definition => field_variable_element_group_value field_variable_element_group_value.group ->) (analysis_message => element_group_analysis_message element_group_analysis_message.group ->) element_group) #8 (field_variable_definition => field_variable_node_definition [field_variable_node_definition.node -> node_output_reference node_output_reference = node_representation => node_representation] [field_variable_node_definition.group -> element_group]) #9 (nodal_freedom_and_value_definition nodal_freedom_and_value_definition.node -> node_output_reference node_output_reference = node_group node_group) #10 ((field_variable_definition => field_variable_whole_model_value) (analysis_message => whole_model_analysis_message))

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO geometry_element FROM fea_model UoF (as ref) #1: geometry_element that is a 2d axisymmetric curve element #2: geometry_element that is a 2d plane curve element (as ref) #3: geometry_element that is a 3d curve element #4: geometry_element that is a 2d axisymmetric surface element #5: geometry_element that is a 2d plane surface element #6: geometry_element that is a 3d surface element #7: geometry_element that is a 2d axisymmetric volume element #8: geometry_element that is a 2d plane volume element #9: geometry_element that is a 3d volume element	PATH			state_definition => #1 (field_variable_definition => field_variable_element_definition => curve_2d_element_field_variable_definition curve_2d_element_field_variable_definition.element -> curve_2d_element_output_reference curve_2d_element_output_reference =\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item) #2 (field_variable_definition => field_variable_element_definition => curve_2d_element_field_variable_definition curve_2d_element_field_variable_definition.element -> curve_2d_element_output_reference curve_2d_element_output_reference =\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item) #3 (field_variable_definition => field_variable_element_definition => curve_3d_element_field_variable_definition curve_3d_element_field_variable_definition.element -> curve_3d_element_output_reference curve_3d_element_output_reference =\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item)

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO geometry_element FROM fea_model UoF (as ref) (*continued*)				#4 (field_variable_definition => field_variable_element_definition => surface_2d_element_field_variable_definition surface_2d_element_field_variable_definition.element -> surface_2d_element_output_reference surface_2d_element_output_reference =\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item) #5 (field_variable_definition => field_variable_element_definition => surface_2d_element_field_variable_definition surface_2d_element_field_variable_definition.element -> surface_2d_element_output_reference surface_2d_element_output_reference =\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item) #6 (field_variable_definition => field_variable_element_definition => surface_3d_element_field_variable_definition surface_3d_element_field_variable_definition.element -> surface_3d_element_output_reference surface_3d_element_output_reference =\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item)

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO geometry_element FROM fea_model UoF (as ref) (*continued*)				#7 (field_variable_definition => field_variable_element_definition => volume_2d_element_field_variable_definition volume_2d_element_field_variable_definition.element -> volume_2d_element_output_reference volume_2d_element_output_reference =\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item) #8 (field_variable_definition => field_variable_element_definition => volume_2d_element_field_variable_definition volume_2d_element_field_variable_definition.element -> volume_2d_element_output_reference volume_2d_element_output_reference =\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item) #9 (field_variable_definition => field_variable_element_definition => volume_3d_element_field_variable_definition volume_3d_element_field_variable_definition.element -> volume_3d_element_output_reference volume_3d_element_output_reference =\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item)
LINEAR_CONSTRAINT_EQUATION linear_constraint_equation <= constraint	linear_constraint_equation_element	104		linear_constraint_equation_element <= constraint_element

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
linear_constraint_equation TO (ONE OR MANY UNIQUE) geometry_element FROM fea_model UoF (as freedoms_and_coefficients_nodes)	PATH			linear_constraint_equation_element.\ freedoms_and_coefficients[i] -> linear_constraint_equation_nodal_term linear_constraint_equation_nodal_term.Node -> node_representation
linear_constraint_equation TO (ONE OR MANY UNIQUE) group FROM fea_model UoF (as freedoms_and_coefficients_nodes)	PATH			linear_constraint_equation_element.\ freedoms_and_coefficients[i] -> linear_constraint_equation_nodal_term linear_constraint_equation_nodal_term.Node -> node_representation
linear_constraint_equation TO (ONE OR MANY UNIQUE) node FROM fea_model UoF (as freedoms_and_coefficients_nodes)	PATH			linear_constraint_equation_element linear_constraint_equation_element.\ freedoms_and_coefficients[i] -> linear_constraint_equation_nodal_term linear_constraint_equation_nodal_term.node -> node_representation
LINEAR_CONSTRAINT_EQUATION_VALUE linear_constraint_equation_value <= fe_analysis_state_definition	linear_constraint_equation_\ element_value	104		linear_constraint_equation_element_value <= state_definition
linear_constraint_equation_value TO linear_constraint_equation (as equation)	PATH			linear_constraint_equation_element_value linear_constraint_equation_element_value.element -> linear_constraint_equation_element
LINEARLY_SUPERIMPOSED_STATE linearly_superimposed_state <= fe_analysis_state	linearly_superimposed_state	104		linearly_superimposed_state <= state
MODES_AND_FREQUENCIES_CONTROL_STEP modes_and_frequencies_control_step <= fe_analysis_control_step	control_linear_modes_and_\ frequencies_analysis_step	104		control_linear_modes_and_ frequencies_analysis_step <= control_analysis_step
frequency_range (EXACTLY TWO ORDERED)	control_linear_modes_and_\ frequencies_analysis_step.\ frequency_range[i]	104		

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
number_of_modes	control_linear_modes_and_\ frequencies_analysis_step.\ number_of_modes	104		
modes_and_frequencies_control_step TO fe_analysis_state (as final_input_state)	PATH			control_linear_modes_and_frequencies_analysis_step <= control_linear_modes_and_frequencies_analysis_step.\ process -> control_linear_modes_and_frequencies_process control_linear_modes_and_frequencies_process.\ final_input_state -> state
NODAL_DEGREE_OF_FREEDOM_REDUCTION nodal_degree_of_freedom_reduction <= constraint	nodal_dof_reduction	104		nodal_dof_reduction <= constraint_element
nodal_degree_of_freedom_reduction TO geometry_element FROM fea_model UoF (as node)	PATH			nodal_dof_reduction nodal_dof_reduction_required_node -> node_output_reference node_output_reference = \ analysis_item_within_representation analysis_item_within_representation.item representation_item => geometric_representation_item
nodal_degree_of_freedom_reduction TO group FROM fea_model UoF (as node)	PATH			nodal_dof_reduction nodal_dof_reduction_required_node -> node_output_reference node_output_reference = node_group node_group
nodal_degree_of_freedom_reduction TO node FROM fea_model UoF (as node)	PATH			nodal_dof_reduction nodal_dof_reduction_required_node -> node_output_reference (node_output_reference = node_representation node_representation) (node_output_reference = substructure_node_reference substructure_node_reference substructure_node_reference.node_ref -> node_representation)

Table 10 - Mapping table for fe_analysis_control UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
NODAL_FREEDOM_DEFINITIONS nodal_freedom_definitions <= fe_analysis_state_definition	nodal_freedom_and_value_definition	104		nodal_freedom_and_value_definition <= state_definition
OUTPUT_REQUEST_STATE output_request_state <= fe_analysis_state	output_request_state	104		output_request_state <= state
output_request_state TO (ONE OR MANY UNIQUE) fe_analysis_control_step (as steps)	PATH			output_request_state <= output_request_state.steps[i] -> control_analysis_step
SINGLE_POINT_CONSTRAINT single_point_constraint <= constraint	single_point_constraint_element	104		single_point_constraint_element <= constraint_element
single_point_constraint TO geometry_element FROM fea_model UoF (as required_node)	PATH			single_point_constraint_element single_point_constraint_element.required_node -> node_output_reference node_output_reference = \\ analysis_item_within_representation analysis_item_within_representation.item representation_item => geometric_representation_item
single_point_constraint TO group FROM fea_model UoF (as required_node)	PATH			single_point_constraint_element single_point_constraint_element.required_node -> node_output_reference node_output_reference = node_group node_group
single_point_constraint TO node FROM fea_model UoF (as required_node)	PATH			single_point_constraint_element single_point_constraint_element.required_node -> node_output_reference (node_output_reference = node_representation node_representation) (node_output_reference = substructure_node_reference substructure_node_reference substructure_node_reference.node_ref -> node_representation)

Table 10 - Mapping table for fe_analysis_control UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
SINGLE_POINT_CONSTRAINT_VALUES single_point_constraint_values <= fe_analysis_state_definition	single_point_constraint_\ element_values	104		single_point_constraint_element_values <= state_definition
single_point_constraint_values TO single_point_constraint (as element)	PATH			single_point_constraint_element_values single_point_constraint_element_values.element -> single_point_constraint_element
SPECIFIED_STATE specified_state <= fe_analysis_state	specified_state	104		specified_state <= state
STATIC_CONTROL_STEP static_control_step <= fe_analysis_control_step	control_linear_static_analysis_step	104		control_linear_static_analysis_step <= control_analysis_step
static_control_step TO fe_analysis_state (as final_input_state)	PATH			control_linear_static_analysis_step <= control_linear_static_analysis_step.process -> control_linear_static_load_increment_process control_linear_static_load_increment_process.\ final_input_state -> state

Table 11 - Mapping table for fe_analysis_results UoF

Application element	AIM element	Source	Rules	Reference Path
ANALYSIS_MESSAGE analysis_message <= fe_analysis_state_definition	analysis_message	104		analysis_message <= state_definition
message_level	analysis_message.message_level	104		
quality	analysis_message.message_text	104		
CALCULATED_STATE calculated_state <= fe_analysis_state	calculated_state	104		calculated_state <= state
CONSTRAINT	constraint_element	104		
identification	constraint_element.element_id	104		
constraint TO (ONE OR MANY UNIQUE) fe_analysis_control_step (as steps)	PATH			constraint_element constraint_element.steps[i] -> control_analysis_step
ELEMENT_FIELD_VARIABLE_DEFINITION element_field_variable_definition <= fe_analysis_state_definition	field_variable_definition	104		field_variable_definition <= state_definition
ELEMENT_NODAL_FREEDOM_ACTIONS element_nodal_freedom_actions <= fe_analysis_state_definition	element_nodal_freedom_actions	104		element_nodal_freedom_actions <= state_definition

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
FE_ANALYSIS_RESULTS	result	104		
fe_analysis_results TO (OPTIONAL) approval FROM authorization UoF (as approved_by)	PATH			result approval_item = result approval_item <- applied_approval_assignment.items[i] applied_approval_assignment
FE_ANALYSIS_RESULTS_STEP	result_analysis_step	104		
fe_analysis_results_step TO fe_analysis_control_step (as control)	PATH			result_analysis_step <- control_result_relationship.result control_result_relationship control_result_relationship.control -> control_analysis_step
fe_analysis_results_step TO fe_analysis_results (as result)	PATH			result_analysis_step result_analysis_step.analysis_result -> result
FE_ANALYSIS_STATE	state	104		
fe_analysis_state TO (ONE OR MANY UNIQUE) fe_analysis_state_definition (as definitions)	PATH			state <- state_definition.defined_state state_definition
FE_ANALYSIS_STATE_DEFINITION	state_definition	104		

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO element FROM fea_model UoF (as ref) #1: to an element #2: to a 2d axisymmetric curve element #3: to a 2d plane curve element #4: to a 3d curve element #5: to a 2d axisymmetric surface element #6: to a 2d plane surface element #7: to a 3d surface element #8: to a 2d axisymmetric volume element #9: to a 2d plane volume element #10: to a 3d volume element	PATH			state_definition => #1 ((element_nodal_freedom_actions element_nodal_freedom_actions.element -> model_or_control_element model_or_control_element = element_representation) (analysis_message => element_analysis_message element_analysis_message.element ->) element_representation) #2 (field_variable_definition => field_variable_element_definition => curve_2d_element_field_variable_definition curve_2d_element_field_variable_definition.element -> curve_2d_element_output_reference (curve_2d_element_output_reference = \ curve_2d_element_representation curve_2d_element_representation = \ axisymmetric_curve_2d_element_representation axisymmetric_curve_2d_element_representation) (curve_2d_element_output_reference = \ curve_2d_element_descriptor curve_2d_element_descriptor = \ axisymmetric_curve_2d_element_descriptor axisymmetric_curve_2d_element_descriptor <- axisymmetric_curve_2d_element_representation.\ element_descriptor axisymmetric_curve_2d_element_representation) (curve_2d_element_output_reference = \ curve_2d_substructure_element_reference curve_2d_substructure_element_reference curve_2d_substructure_element_reference.element_ref -> curve_2d_element_representation curve_2d_element_representation = \ axisymmetric_curve_2d_element_representation axisymmetric_curve_2d_element_representation)) *(continued on the following page)*

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO element FROM fea_model UoF (as ref) (*continued*)	PATH			#3 (field_variable_definition => field_variable_element_definition => curve_2d_element_field_variable_definition curve_2d_element_field_variable_definition.element -> curve_2d_element_output_reference (curve_2d_element_output_reference = \\ curve_2d_element_representation curve_2d_element_representation = \\ plane_curve_2d_element_representation plane_curve_2d_element_representation) (curve_2d_element_output_reference = \\ curve_2d_element_descriptor curve_2d_element_descriptor = \\ plane_curve_2d_element_descriptor plane_curve_2d_element_descriptor <- plane_curve_2d_element_representation.element_descriptor plane_curve_2d_element_representation) (curve_2d_element_output_reference = \\ curve_2d_substructure_element_reference curve_2d_substructure_element_reference curve_2d_substructure_element_reference.element_ref -> curve_2d_element_representation curve_2d_element_representation = \\ plane_curve_2d_element_representation plane_curve_2d_element_representation)) #4 (field_variable_definition => field_variable_element_definition => curve_3d_element_field_variable_definition curve_3d_element_field_variable_definition.element -> curve_3d_element_output_reference

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO element FROM fea_model UoF (as ref) (*continued*)				(curve_3d_element_output_reference = \ curve_3d_element_representation curve_3d_element_representation) (curve_3d_element_output_reference = \ curve_3d_element_descriptor curve_3d_element_descriptor <- curve_3d_element_representation.element_descriptor curve_3d_element_representation) (curve_3d_element_output_reference = \ curve_3d_substructure_element_reference curve_3d_substructure_element_reference curve_3d_substructure_element_reference.element_ref -> curve_3d_element_representation)) #5 (field_variable_definition => field_variable_element_definition => surface_2d_element_field_variable_definition surface_2d_element_field_variable_definition.element -> surface_2d_element_output_reference (surface_2d_element_output_reference = \ surface_2d_element_representation surface_2d_element_representation = \ axisymmetric_surface_2d_element_representation axisymmetric_surface_2d_element_representation) (surface_2d_element_output_reference = \ surface_2d_element_descriptor surface_2d_element_descriptor = \ axisymmetric_surface_2d_element_descriptor axisymmetric_surface_2d_element_descriptor <- axisymmetric_surface_2d_element_representation.\ element_descriptor axisymmetric_surface_2d_element_representation)

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO element FROM fea_model UoF (as ref) (*continued*)				(surface_2d_element_output_reference = \ surface_2d_substructure_element_reference surface_2d_substructure_element_reference surface_2d_substructure_element_reference.element_ref -> surface_2d_element_representation surface_2d_element_representation = \ axisymmetric_surface_2d_element_representation axisymmetric_surface_2d_element_representation)) #6 (field_variable_definition => field_variable_element_definition => surface_2d_element_field_variable_definition surface_2d_element_field_variable_definition.element -> surface_2d_element_output_reference (surface_2d_element_output_reference = \ surface_2d_element_representation surface_2d_element_representation = \ plane_surface_2d_element_representation plane_surface_2d_element_representation) (surface_2d_element_output_reference = \ surface_2d_element_descriptor surface_2d_element_descriptor = \ plane_surface_2d_element_descriptor plane_surface_2d_element_descriptor <- plane_surface_2d_element_representation.element_descriptor plane_surface_2d_element_representation) (surface_2d_element_output_reference = \ surface_2d_substructure_element_reference surface_2d_substructure_element_reference surface_2d_substructure_element_reference.element_ref -> surface_2d_element_representation surface_2d_element_representation = \ plane_surface_2d_element_representation plane_surface_2d_element_representation))

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO element FROM fea_model UoF (as ref) (*continued*)				#7 (field_variable_definition => field_variable_element_definition => surface_3d_element_field_variable_definition surface_3d_element_field_variable_definition.element -> surface_3d_element_output_reference (surface_3d_element_output_reference = \\ surface_3d_element_representation surface_3d_element_representation) (surface_3d_element_output_reference = \\ surface_3d_element_descriptor surface_3d_element_descriptor <- surface_3d_element_representation.element_descriptor surface_3d_element_representation) (surface_3d_element_output_reference = \\ surface_3d_substructure_element_reference surface_3d_substructure_element_reference surface_3d_substructure_element_reference.element_ref -> surface_3d_element_representation)) #8 (field_variable_definition => field_variable_element_definition => volume_2d_element_field_variable_definition volume_2d_element_field_variable_definition.element -> volume_2d_element_output_reference (volume_2d_element_output_reference = \\ volume_2d_element_representation volume_2d_element_representation = \\ axisymmetric_volume_2d_element_representation axisymmetric_volume_2d_element_representation)

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO element FROM fea_model UoF (as ref) (*continued*)				(volume_2d_element_output_reference = \\ volume_2d_element_descriptor volume_2d_element_descriptor = \\ axisymmetric_volume_2d_element_descriptor axisymmetric_volume_2d_element_descriptor <- axisymmetric_volume_2d_element_representation.\\ element_descriptor axisymmetric_volume_2d_element_representation) (volume_2d_element_output_reference = \\ volume_2d_substructure_element_reference volume_2d_substructure_element_reference volume_2d_substructure_element_reference.element_ref -> volume_2d_element_representation volume_2d_element_representation = \\ axisymmetric_volume_2d_element_representation axisymmetric_volume_2d_element_representation)) #9 (field_variable_definition => field_variable_element_definition => volume_2d_element_field_variable_definition volume_2d_element_field_variable_definition.element -> volume_2d_element_output_reference (volume_2d_element_output_reference = \\ volume_2d_element_representation volume_2d_element_representation = \\ plane_volume_2d_element_representation plane_volume_2d_element_representation) (volume_2d_element_output_reference = \\ volume_2d_element_descriptor volume_2d_element_descriptor = \\ plane_volume_2d_element_descriptor plane_volume_2d_element_descriptor <- plane_volume_2d_element_representation.element_descriptor plane_volume_2d_element_representation)

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO element FROM fea_model UoF (as ref) (*continued*)				(volume_2d_element_output_reference = \ volume_2d_substructure_element_reference volume_2d_substructure_element_reference volume_2d_substructure_element_reference.element_ref -> volume_2d_element_representation volume_2d_element_representation = \ plane_volume_2d_element_representation plane_volume_2d_element_representation)) #10 (field_variable_definition => field_variable_element_definition => volume_3d_element_field_variable_definition volume_3d_element_field_variable_definition.element -> volume_3d_element_output_reference (volume_3d_element_output_reference = \ volume_3d_element_representation volume_3d_element_representation) (volume_3d_element_output_reference = \ volume_3d_element_descriptor volume_3d_element_descriptor <- volume_3d_element_representation.element_descriptor volume_3d_element_representation) (volume_3d_element_output_reference = \ volume_3d_substructure_element_reference volume_3d_substructure_element_reference volume_3d_substructure_element_reference.element_ref -> volume_3d_element_representation))
fe_analysis_state_definition TO node FROM fea_model UoF (as ref)	PATH			state_definition => (nodal_freedom_and_value_definition nodal_freedom_and_value_definition.node -> node_output_reference node_output_reference = node) (analysis_message => element_analysis_message element_analysis_message.node ->) node

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO group FROM fea_model UoF (as ref) #1: to a group of 2d curve elements #2: to a group of 2d surface elements #3: to a group of 2d volume elements #4: to a group of 3d curve elements #5: to a group of 3d surface elements #6: to a group of 3d volume elements #7: to a group of elements #8: to a group of one node and a group of elements #9: to a group of nodes #10: to a group that is the whole model	PATH			state_definition => #1 (field_variable_definition => field_variable_element_definition => curve_2d_element_field_variable_definition curve_2d_element_field_variable_definition.element -> curve_2d_element_output_reference curve_2d_element_output_reference = \\ curve_2d_element_group curve_2d_element_group) #2 (field_variable_definition => field_variable_element_definition => surface_2d_element_field_variable_definition surface_2d_element_field_variable_definition.element -> surface_2d_element_output_reference surface_2d_element_output_reference = \\ surface_2d_element_group surface_2d_element_group) #3 (field_variable_definition => field_variable_element_definition => volume_2d_element_field_variable_definition volume_2d_element_field_variable_definition.element -> volume_2d_element_output_reference volume_2d_element_output_reference = \\ volume_2d_element_group volume_2d_element_group)

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO group FROM fea_model UoF (as ref) (*continued*)				#4 (field_variable_definition => field_variable_element_definition => curve_3d_element_field_variable_definition curve_3d_element_field_variable_definition.element -> curve_3d_element_output_reference curve_3d_element_output_reference =\ curve_3d_element_group curve_3d_element_group) #5 (field_variable_definition => field_variable_element_definition => surface_3d_element_field_variable_definition surface_3d_element_field_variable_definition.element -> surface_3d_element_output_reference surface_3d_element_output_reference = \ surface_3d_element_group surface_3d_element_group) #6 (field_variable_definition => field_variable_element_definition => volume_3d_element_field_variable_definition volume_3d_element_field_variable_definition.element -> volume_3d_element_output_reference volume_3d_element_output_reference = \ volume_3d_element_group volume_3d_element_group)

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO group FROM fea_model UoF (as ref) (*continued*)				#7 ((field_variable_definition => field_variable_element_group_value field_variable_element_group_value.group ->) (analysis_message => element_group_analysis_message element_group_analysis_message.group ->) element_group) #8 (field_variable_definition => field_variable_node_definition [field_variable_node_definition.node -> node_output_reference node_output_reference = node_representation => node_representation] [field_variable_node_definition.group -> element_group]) #9 (nodal_freedom_and_value_definition nodal_freedom_and_value_definition.node -> node_output_reference node_output_reference = node_group node_group) #10 ((field_variable_definition => field_variable_whole_model_value) (analysis_message => whole_model_analysis_message))

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO geometry_element FROM fea_model UoF (as ref) #1: geometry_element that is a 2d axisymmetric curve element #2: geometry_element that is a 2d plane curve element (as ref) #3: geometry_element that is a 3d curve element #4: geometry_element that is a 2d axisymmetric surface element #5: geometry_element that is a 2d plane surface element #6: geometry_element that is a 3d surface element #7: geometry_element that is a 2d axisymmetric volume element #8: geometry_element that is a 2d plane volume element #9: geometry_element that is a 3d volume element	PATH			state_definition => #1 (field_variable_definition => field_variable_element_definition => curve_2d_element_field_variable_definition curve_2d_element_field_variable_definition.element -> curve_2d_element_output_reference curve_2d_element_output_reference =\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item) #2 (field_variable_definition => field_variable_element_definition => curve_2d_element_field_variable_definition curve_2d_element_field_variable_definition.element -> curve_2d_element_output_reference curve_2d_element_output_reference =\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item) #3 (field_variable_definition => field_variable_element_definition => curve_3d_element_field_variable_definition curve_3d_element_field_variable_definition.element -> curve_3d_element_output_reference curve_3d_element_output_reference =\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item) (*continued on the following page*)

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO geometry_element FROM fea_model UoF (as ref) (*continued*)				#4 (field_variable_definition => field_variable_element_definition => surface_2d_element_field_variable_definition surface_2d_element_field_variable_definition.element -> surface_2d_element_output_reference surface_2d_element_output_reference =\\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item) #5 (field_variable_definition => field_variable_element_definition => surface_2d_element_field_variable_definition surface_2d_element_field_variable_definition.element -> surface_2d_element_output_reference surface_2d_element_output_reference =\\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item) #6 (field_variable_definition => field_variable_element_definition => surface_3d_element_field_variable_definition surface_3d_element_field_variable_definition.element -> surface_3d_element_output_reference surface_3d_element_output_reference =\\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item)

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fe_analysis_state_definition TO geometry_element FROM fea_model UoF (as ref) (*continued*)				#7 (field_variable_definition => field_variable_element_definition => volume_2d_element_field_variable_definition volume_2d_element_field_variable_definition.element -> volume_2d_element_output_reference volume_2d_element_output_reference =\\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item) #8 (field_variable_definition => field_variable_element_definition => volume_2d_element_field_variable_definition volume_2d_element_field_variable_definition.element -> volume_2d_element_output_reference volume_2d_element_output_reference =\\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item) #9 (field_variable_definition => field_variable_element_definition => volume_3d_element_field_variable_definition volume_3d_element_field_variable_definition.element -> volume_3d_element_output_reference volume_3d_element_output_reference =\\ analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item)
LINEAR_CONSTRAINT_EQUATION linear_constraint_equation <= constraint	linear_constraint_equation_element	104		linear_constraint_equation_element <= constraint_element

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
LINEAR_CONSTRAINT_EQUATION linear_constraint_equation <= constraint	linear_constraint_equation_element	104		linear_constraint_equation_element <= constraint_element
linear_constraint_equation TO (ONE OR MANY UNIQUE) geometry_element FROM fea_model UoF (as freedoms_and_coefficients_nodes)	PATH			linear_constraint_equation_element.\ freedoms_and_coefficients[i] -> linear_constraint_equation_nodal_term linear_constraint_equation_nodal_term.Node -> node_representation
linear_constraint_equation TO (ONE OR MANY UNIQUE‘group FROM fea_model UoF (as freedoms_and_coefficients_nodes)	PATH			linear_constraint_equation_element.\ freedoms_and_coefficients[i] -> linear_constraint_equation_nodal_term linear_constraint_equation_nodal_term.Node -> node_representation
linear_constraint_equation TO (ONE OR MANY UNIQUE) node FROM fea_model UoF (as freedoms_and_coefficients_nodes)	PATH			linear_constraint_equation_element linear_constraint_equation_element.\ freedoms_and_coefficients[i] -> linear_constraint_equation_nodal_term linear_constraint_equation_nodal_term.node -> node_representation
LINEAR_CONSTRAINT_EQUATION_VALUE linear_constraint_equation_value <= fe_analysis_state_definition	linear_constraint_equation_\ element_value	104		linear_constraint_equation_element_value <= state_definition
linear_constraint_equation_value TO linear_constraint_equation (as equation)	PATH			linear_constraint_equation_element_value linear_constraint_equation_element_value.element -> linear_constraint_equation_element
LINEARLY_SUPERIMPOSED_STATE linearly_superimposed_state <= fe_analysis_state	linearly_superimposed_state	104		linearly_superimposed_state <= state

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
MODES_AND_FREQUENCIES_RESULTS_STEP modes_and_frequencies_results_step <= fe_analysis_results_step	result_linear_modes_and_\ frequencies_analysis_sub_step	104		result_linear_modes_and_frequencies_\ analysis_sub_step <= result_analysis_step
modes_and_frequencies_results_step to (ONE OR MANY) calculated_state (as resulting_states)	PATH			result_linear_modes_and_frequencies_\ analysis_sub_step result_linear_modes_and_frequencies_\ analysis_sub_step.states[i] calculated_state
NODAL_DEGREE_OF_FREEDOM_REDUCTION nodal_degree_of_freedom_reduction <= constraint	nodal_dof_reduction	104		nodal_dof_reduction <= constraint_element
nodal_degree_of_freedom_reduction TO geometry_element FROM fea_model UoF (as node)	PATH			nodal_dof_reduction nodal_dof_reduction_required_node -> node_output_reference node_output_reference = \ analysis_item_within_representation analysis_item_within_representation.item representation_item => geometric_representation_item
nodal_degree_of_freedom_reduction TO group FROM fea_model UoF (as node)	PATH			nodal_dof_reduction nodal_dof_reduction_required_node -> node_output_reference node_output_reference = node_group node_group
nodal_degree_of_freedom_reduction TO node FROM fea_model UoF (as node)	PATH			nodal_dof_reduction nodal_dof_reduction_required_node -> node_output_reference (node_output_reference = node_representation node_representation) (node_output_reference = substructure_node_reference substructure_node_reference substructure_node_reference.node_ref -> node_representation)

Table 11 - Mapping table for fe_analysis_results UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
NODAL_FREEDOM_DEFINITIONS nodal_freedom_definitions <= fe_analysis_state_definition	nodal_freedom_and_value_definition	104		nodal_freedom_and_value_definition <= state_definition
OUTPUT_REQUEST_STATE output_request_state <= fe_analysis_state	output_request_state	104		output_request_state <= state
output_request_state TO (ONE OR MANY UNIQUE) fe_analysis_control_step (as steps)	PATH			output_request_state <= output_request_state.steps[i] -> control_analysis_step
SINGLE_POINT_CONSTRAINT single_point_constraint <= constraint	single_point_constraint_element	104		single_point_constraint_element <= constraint_element
single_point_constraint TO geometry_element FROM fea_model UoF (as required_node)	PATH			single_point_constraint_element single_point_constraint_element.required_node -> node_output_reference node_output_reference = \ analysis_item_within_representation analysis_item_within_representation.item representation_item => geometric_representation_item
single_point_constraint TO group FROM fea_model UoF (as required_node)	PATH			single_point_constraint_element single_point_constraint_element.required_node -> node_output_reference node_output_reference = node_group node_group
single_point_constraint TO node FROM fea_model UoF (as required_node)	PATH			single_point_constraint_element single_point_constraint_element.required_node -> node_output_reference (node_output_reference = node_representation node_representation) (node_output_reference = substructure_node_reference substructure_node_reference substructure_node_reference.node_ref -> node_representation)

Table 11 - Mapping table for FE_analysis_results UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
SINGLE_POINT_CONSTRAINT_VALUES single_point_constraint_values <= fe_analysis_state_definition	single_point_constraint_\ element_values	104		(single_point_constraint_element_values <= state_definition
single_point_constraint_values TO single_point_constraint (as element)	PATH			single_point_constraint_element_values single_point_constraint_element_values.element -> single_point_constraint_element
SPECIFIED_STATE specified_state <= fe_analysis_state	specified_state	104		specified_state <= state
STATIC_RESULTS_STEP static_results_step <= fe_analysis_results_step	result_linear_modes_and_\ frequencies_analysis_sub_step	104		result_linear_static_analysis_sub_step <= result_analysis_step
static_results_step to calculated_state (as resulting_state)	PATH			result_linear_static_analysis_sub_step result_linear_static_analysis_sub_step.state calculated_state

Table 12 - Mapping table for fea_model UoF

Application element	AIM element	Source	Rules	Reference Path
CURVE_CROSS_SECTION	shape_representation	41		
curve_cross_section TO curve_property (as property) #1: Property of 2d axisymmetric curve element #2: Property of 2d plane curve element #3: Property of 3d curve element	PATH			shape_representation <= representation <- representation_relationship.rep_1 representation_relationship representation_relationship.rep_2 - > representation => element_representation => #1 (axisymmetric_curve_2d_element_representation axisymmetric_curve_2d_element_representation.property -> curve_2d_element_property curve_2d_element_property.section -> curve_element_section_definition) #2 (plane_curve_2d_element_representation plane_curve_2d_element_representation.property -> curve_2d_element_property curve_2d_element_property.section -> curve_element_section_definition) #3 (curve_3d_element_representation curve_3d_element_representation.property -> curve_3d_element_property curve_3d_element_property.interval_definitions[i] -> curve_element_interval => {curve_element_interval_constant curve_element_interval_constant.section -> curve_element_section_definition} {curve_element_interval_linearly_varying curve_element_interval_linearly_varying.sections[i] -> curve_element_section_definition})

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
curve_cross_section TO (OPTIONAL) idealized_analysis_shape (as section)	PATH			shape_representation <= representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation {representation => shape_representation} representation <- property_definition_representation.used_representation property_definition_representation => shape_definition_representation
curve_cross_section TO (OPTIONAL) nominal_design_shape (as section)	PATH			shape_representation <= representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation {representation => shape_representation} representation <- property_definition_representation.used_representation property_definition_representation => shape_definition_representation

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
CURVE_ELEMENT				
#1: curve_element <= element as 2d axisymmetric	#1 (axisymmetric_curve_2d_element_\ representation)	104		#1 (axisymmetric_curve_2d_element_representation <= element_representation)
#2: curve_element <= element as 2d plane	#2 (plane_curve_2d_element_\ representation)	104		#2 (plane_curve_2d_element_representation <= element_representation)
#3: curve_element <= element as 3d	#3 (curve_3d_element_representation)	104		#3 (curve_3d_element_representation <= element_representation)
CURVE_PROPERTY				
#1: Property for a 2d curve element	#1 (curve_2d_element_property)	104		
#2: Property for a 3d curve element	#2 (curve_3d_element_property)	104		
#3: Angle property for a 2d axisymmetric curve element	#3 (axisymmetric_2d_element_\ property)	104		
#4: Depth property for a 2d plane curve element	#4 (plane_2d_element_property)	104		
curve_property TO (ONE OR MANY UNIQUE) curve_element (as defined_elements)	PATH			
#1: curve_element that is a 2d axisymmetric curve element				#1 (curve_2d_element_property < - axisymmetric_curve_2d_element_representation.property axisymmetric_curve_2d_element_representation)
#2: curve_element that is a 2d plane curve element				#2 (curve_2d_element_property < - plane_curve_2d_element_representation.property plane_curve_2d_element_representation)
#3: curve_element that is a 3d curve element				#3 (curve_3d_element_property < - curve_3d_element_representation.property curve_3d_element_representation)
#4: as an angle property to a curve_element that is a 2d axisymmetric curve element				#4 (axisymmetric_2d_element_property < - axisymmetric_curve_2d_element_representation.\ angle_property axisymmetric_curve_2d_element_representation)
#5: as a depth property to a curve_element that is a 2d plane curve element				#5 (plane_2d_element_property < - plane_curve_2d_element_representation.depth_property plane_curve_2d_element_representation)

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
CURVE_SECTION_PROPERTIES	curve_element_section_derived_\ definitions	104		
curve_section_properties TO curve_property (as property) #1: Section properties for a 2d curve element #2: Constant interval properties for a 3d curve element #3: Linearly varying properties for a 3d curve element	PATH			curve_element_section_derived_definitions <= curve_element_section_definition < - #1 (curve_2d_element_property.section curve_2d_element_property) #2 (curve_element_interval_constant.section curve_element_interval_constant <= curve_element_interval < - curve_3d_element_property.interval_definitions[i] curve_3d_element_property) #3 (curve_element_interval_linearly_varying.sections[i] curve_element_interval_linearly_varying <= curve_element_interval < - curve_3d_element_property.interval_definitions[i] curve_3d_element_property)
DAMPING_MATRIX #1: damping_matrix <= matrix as a point element nodal response matrix #2: damping_matrix <= matrix as a coefficient #3: damping_matrix <= matrix directly specified	#1 (grounded_damper) #2 (directionally_explicit_element_\ coefficient) #3 (explicit_element_matrix)	104 104 104		#1 (grounded_damper <= point_element_matrix) #2 (directionally_explicit_element_coefficient {directionally_explicit_element_coefficient directionally_explicit_element_coefficient.\ property_type = damping}) #3 (explicit_element_matrix {explicit_element_matrix explicit_element_matrix.property_type = damping})

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
DIRECTIONALLY_EXPLICIT_ELEMENT directionally_explicit_element <= element	directionally_explicit_element_\ representation	104		directionally_explicit_element_representation <= element_representation
directionally_explicit_element TO matrix (as associated_matrix)	PATH			directionally_explicit_element_representation directionally_explicit_element_representation.\ coefficient -> directionally_explicit_element_coefficient
ELEMENT	element_representation	104		
element_identification	representation.name	43		element_representation <= representation representation.name
element TO (OPTIONAL) element_description (as description)	PATH			element_representation <= representation < - property_definition_representation.used_representation property_definition_representation {property_definition_representation => structural_response_property_definition_representation} property_definition_representation.definition -> property_definition {property_definition => structural_response_property} property_definition.definition -> characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect => element_definition

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
element TO fea_model (as model_ref) #1: as curve_element that is 2d axisymmetric #2: as curve_element that is 2d plane #3: as curve_element that is 3d #4: as directionally_explicit_element #5: as explicit_element #6: as point_element #7: as surface_element that is 2d axisymmetric #8: as surface_element that is 2d plane #9: as surface_element that is 3d #10: as volume_element that is 2d axisymmetric #11: as volume_element that is 2d plane #12: as volume_element that is 3d #13: as substructure_element	PATH			element_representation => #1 (axisymmetric_curve_2d_element_representation axisymmetric_curve_2d_element_representation.\ model_ref -> fea_model_2d <=) #2 (plane_curve_2d_element_representation plane_curve_2d_element_representation.model_ref -> fea_model_2d <=) #3 (curve_3d_element_representation curve_3d_element_representation.model_ref -> fea_model_3d <=) #4 (directionally_explicit_element_representation directionally_explicit_element_representation.model_ref -> fea_model_3d <=) #5 (explicit_element_representation explicit_element_representation.model_ref -> fea_model_3d <=) #6 (point_element_representation point_element_representation.model_ref -> fea_model_3d <=) #7 (axisymmetric_surface_2d_element_representation axisymmetric_surface_2d_element_representation.\ model_ref -> fea_model_2d <=) #8 (plane_surface_2d_element_representation plane_surface_2d_element_representation.model_ref -> fea_model_2d <=) (*continued on the following page*)

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
element TO fea_model (as model_ref) (*continued*)	PATH			#9 (surface_3d_element_representation surface_3d_element_representation.model_ref -> fea_model_3d <=) #10 (axisymmetric_volume_2d_element_representation axisymmetric_volume_2d_element_representation.\ model_ref -> fea_model_2d <=) #11 (plane_volume_2d_element_representation plane_volume_2d_element_representation.model_ref -> fea_model_2d <=) #12 (volume_3d_element_representation volume_3d_element_representation.model_ref -> fea_model_3d <=) #13 (substructure_element_representation substructure_element_representation.model_ref -> fea_model_3d <=) fea_model
element TO (ONE OR MANY ORDERED) node (as node_list)	PATH			element_representation element_representation.node_list[i] -> node_representation => node
ELEMENT_DESCRIPTION	element_definition	104		
description	shape_aspect.description	41		element_definition <= shape_aspect shape_aspect.description
ELEMENT-PROPERTY_GEOMETRIC_RELATIONSHIP	#1 (fea_curve_section_geometric_\ relationship) #2 (fea_surface_section_geometric_\ relationship)	104 104		

Table 12 - Mapping table for fea_model UoF (continued)

element_property_geometric_relationship TO geometry_element	PATH			#1 fea_curve_section_geometric_relationship fea_curve_section_geometric_relationship_item -> analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item) #2 (fea_surface_section_geometric_relationship fea_surface_section_geometric_relationship.item -> analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item)
element_property_geometric_relationship TO curve_property (as property_ref)	PATH			fea_curve_section_geometric_relationship fea_curve_section_geometric_relationship.section_ref -> curve_element_section_definition
element_property_geometric_relationship TO surface_ property (as property_ref)	PATH			fea_surface_section_geometric_relationship fea_surface_section_geometric_relationship.section_ref -> surface_section
ELEMENT_SHAPE_ASPECT	element_aspect	104		
ELEMENT_SHAPE_RELATIONSHIP	element_geometric_relationship	104		
role	analysis_item_within_\ representation.description	104		element_geometric_relationship element_geometric_relationship.item -> analysis_item_within_representation analysis_item_within_representation.description

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
element_shape_relationship TO element_shape_aspect (as element_aspect)	PATH			element_geometric_relationship element_geometric_relationship.aspect -> element_aspect
element_shape_relationship TO element (as element_reference)	PATH			element_geometric_relationship element_geometric_relationship.element_ref -> element_representation
element_shape_relationship TO geometry_element FROM advanced_boundary_representation UoF OR FROM composite_constituent_representation UoF OR FROM faceted_boundary_representation UoF OR FROM fea_model UoF OR FROM manifold_surface_with_topology UoF OR FROM non_topological_surface_and_wireframe UoF OR FROM wireframe_with_topology UoF (as shape)	PATH			element_geometric_relationship element_geometric_relationship.item -> analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item
ENVIRONMENT	data_environment	45		
EXPLICIT_ELEMENT explicit_element <= element	explicit_element_representation	104		explicit_element_representation <= element_representation
explicit_element TO matrix (as associated_matrix)	PATH			explicit_element_representation explicit_element_representation.matrix -> explicit_element_matrix
FEA_MATERIAL_DEFINITION	fea_material_property_representation	104		
description	property_definition.description	104		fea_material_property_representation <= material_property_representation <= property_definition_representation property_definition_representation.definition -> property_definition {property_definition => material_property_definition} property_definition.description

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
material_identification	representation.name	104		fea_material_property_representation <= material_property_representation <= property_definition_representation property_definition_representation.used_representation -> representation representation.name
fea_material_definition TO (ZERO, ONE OR MANY UNIQUE) curve_element (as elements) #1: curve_element that is 2d axisymmetric #2: curve_element that is 2d plane #3: curve_element that is 3d	PATH			#1 (fea_material_property_representation <= material_property_representation <- element_material.properties[i] element_material <- axisymmetric_curve_2d_element_representation.material axisymmetric_curve_2d_element_representation) #2 (fea_material_property_representation <= material_property_representation <- element_material.properties[i] element_material <- plane_curve_2d_element_representation.material plane_curve_2d_element_representation) #3 (fea_material_property_representation <= material_property_representation <- element_material.properties[i] element_material <- curve_3d_element_representation.material curve_3d_element_representation)

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fea_material_definition TO (ZERO, ONE OR MANY UNIQUE) surface_element (as elements) #1: surface_element that is 2d axisymmetric #2: surface_element that is 2d plane #3: surface_element that is 3d	PATH			#1 (fea_material_property_representation <= material_property_representation <- element_material.properties[i] element_material <- axisymmetric_surface_2d_element_representation.material axisymmetric_surface_2d_element_representation) #2 (fea_material_property_representation <= material_property_representation <- element_material.properties[i] element_material <- plane_surface_2d_element_representation.material plane_surface_2d_element_representation) #3 (fea_material_property_representation <= material_property_representation <- element_material.properties[i] element_material <- surface_3d_element_representation.material surface_3d_element_representation)

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fea_material_definition TO (ZERO, ONE OR MANY UNIQUE) volume_element (as elements) #1: volume_element that is 2d axisymmetric #2: volume_element that is 2d plane #3: volume_element that is 3d	PATH			#1 (fea_material_property_representation <= material_property_representation <- element_material.properties[i] element_material <- axisymmetric_volume_2d_element_representation.material axisymmetric_volume_2d_element_representation) #2 (fea_material_property_representation <= material_property_representation <- element_material.properties[i] element_material <- plane_volume_2d_element_representation.material plane_volume_2d_element_representation) #3 (fea_material_property_representation <= material_property_representation <- element_material.properties[i] element_material <- volume_3d_element_representation.material volume_3d_element_representation)

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fea_material_definition TO (ZERO, ONE OR MANY UNIQUE) shape_aspect (as elements)	PATH			fea_material_property_representation <- fea_material_property_geometric_relationship.material_ref fea_material_property_geometric_relationship fea_material_property_geometric_relationship.item -> analysis_item_within_representation [analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item] [analysis_item_within_representation analysis_item_within_representation.rep -> representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition -> property_definition property_definition.definition -> characterized_definition characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect]

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
FEA_MATERIAL_PROPERTY fea_material_property <= material_property #1: as a specialized fea material property representation #2: as a general material property representation	material_property	45		material_property {material_property <= property_definition <- property_definition_representation.definition property_definition_representation #1 ({property_definition_representation => material_property_representation => fea_material_property_representation} property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => fea_material_property_representation_item) #2 ({property_definition_representation => material_property_representation} property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => measure_representation_item)}
fea_material_property TO fea_material_definition (as property_use)	PATH			material_property <= property_definition <- property_definition_representation.definition property_definition_representation => material_property_representation => fea_material_property_representation

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
FEA_MATERIAL_SPECIFICATION fea_material_specification <= material_specification	document	41	18	document {document document.kind -> document_type document_type.product_data_type = \\ 'fea material specification'} document <- document_reference.assigned_document document_reference => applied_document_reference
fea_material_specification TO fea_material_definition (as specification_use)	PATH			document <- document_reference.assigned_document document_reference => applied_document_reference applied_document_reference.items[i] -> document_reference_item document_reference_item = product_definition product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition < - property_definition.definition property_definition {property_definition => material_property} property_definition <-property_definition_representation.definition property_definition_representation => material_property_representation => fea_material_property_representation

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
FEA_MODEL	fea_model	104		
analysis_type	fea_model.analysis_type	104		
fea_model TO (OPTIONAL) file	PATH		18	fea_model <= representation < - property_definition_representation.used_representation property_definition_representation {property_definition_representation => structural_response_property_definition_representation} property_definition_representation.definition -> property_definition {property_definition => structural_response_property} property_definition.definition -> characterized_definition characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect {shape_aspect => fea_model_definition} shape_aspect.of_shape -> product_definition_shape <= property_definition property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition = product_definition product_definition => product_definition_with_associated_documents product_definition_with_associated_documents.\ documentation_ids[i] -> document {document.kind -> document_type document_type.product_data_type = 'fea file'}
creating_software	fea_model.creating_software	104		
identification	representation.name	43		fea_model <= representation representation.name
intended_analysis_code	fea_model.intended_analysis_code[i]	104		

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fea_model TO (OPTIONAL) approval FROM authorization UoF (as approved_by)	PATH			fea_model approval_item = fea_model approval_item < - applied_approval_assignment.items[i] applied_approval_assignment
fea_model TO analysis_discipline_product_definition (as definition)	PATH			fea_model <= representation < - property_definition_representation.used_representation property_definition_representation {property_definition_representation => structural_response_property_definition_representation} property_definition_representation.definition -> property_definition {property_definition => structural_response_property} property_definition.definition -> characterized_definition characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect {shape_aspect => fea_model_definition} shape_aspect.of_shape -> product_definition_shape <= property_definition property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition = product_definition product_definition

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
fea_model TO fea_model_description (as description)	PATH			fea_model <= representation < - property_definition_representation.used_representation property_definition_representation {property_definition_representation => structural_response_property_definition_representation} property_definition_representation.definition -> property_definition {property_definition => structural_response_property} property_definition.definition -> characterized_definition characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect => fea_model_definition
FEA_MODEL_DESCRIPTION	fea_model_definition	104		
description	shape_aspect.description	41		fea_model_definition <= shape_aspect shape_aspect.description
GEOMETRIC_MODEL_REPRESENTATION	shape_representation	41	21, 23, 25, 26, 47	
role	representation.name	43		shape_representation <= representation representation.name
geometric_model_representation TO (ONE OR MANY UNIQUE) geometry_element (as elements)	PATH	41		shape_representation <= representation representation.items[i] -> representation_item => geometric_representation_item
GEOMETRY_ELEMENT	geometric_representation_item	43		

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
GROUP	fea_group	104		
description	group.description	41		fea_group <= group group.description
group_identification	group.name	41		fea_group <= group group.name
group TO (ZERO, ONE OR MANY UNIQUE) element (as elements)	PATH			fea_group => element_group element_group.elements[i] -> element_representation
group TO (ZERO, ONE OR MANY UNIQUE) node (as nodes)	PATH			fea_group => node_group node_group.nodes[i] -> node_representation => node
GROUP_RELATIONSHIP	fea_group_relation	104		
group_relationship TO group (as related_group)	PATH			fea_group_relation <= group_relationship group_relationship.related_group -> group => fea_group
group_relationship TO group (as relating_group)	PATH			fea_group_relation <= group_relationship group_relationship.relating_group -> group => fea_group

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
MASS_MATRIX #1: mass_matrix <= matrix as a point element nodal response matrix #2: mass_matrix <= matrix as a coefficient #3: mass_matrix <= matrix directly specified	#1 (stationary_mass) #2 (directionally_explicit_element_\ coefficient) #3 (explicit_element_matrix)	104 104 104		#1 (stationary_mass <= point_element_matrix) #2 (directionally_explicit_element_coefficient {directionally_explicit_element_coefficient directionally_explicit_element_coefficient.\ property_type = mass}) #3 (explicit_element_matrix {explicit_element_matrix explicit_element_matrix.property_type = mass})
MATERIAL_PROPERTY	material_property	45		
property_name	property_definition.name	41		material_property <= property_definition property_definition.name
material_property TO environment (as ambient)	PATH			material_property <= property_definition < - property_definition_representation.definition property_definition_representation => material_property_representation material_property_representation.dependent_env -> data_environment
material_property TO (OPTIONAL) measure_value (as property_value)	PATH			material_property <= property_definition <- property_definition_representation.definition property_definition_representation {property_definition_representation => material_property_representation} property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => measure_representation_item

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
MATERIAL_SPECIFICATION	document	41	18	document
material_specification <= specification				{document document.kind -> document_type document_type.product_data_type = 'material specification'} document <- document_reference.assigned_document document_reference => applied_document_reference
MATRIX	#1 (point_element_matrix)	104		
#1: Represent matrix as a point element nodal response matrix	#2 (directionally_explicit_element_\ coefficient)	104		
#2: Represent matrix as a coefficient	#3 (explicit_element_matrix)	104		
#3: Directly specify the matrix				
MEASURE_VALUE	measure_representation_item	45		
NODAL_RESULTS_COORDINATE_SYSTEM	node_with_solution_coordinate_\ system	104		
NODE	node	104		
node_identification	representation.name	43		node <= node_representation <= representation representation.name

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
node TO (OPTIONAL) node_description (as description)	PATH			node <= node_representation <= representation < - property_definition_representation.used_representation property_definition_representation {property_definition_representation => structural_response_property_definition_representation} property_definition_representation.definition -> property_definition {property_definition => structural_response_property} property_definition.definition -> characterized_definition characterized_definition = shape_definition shape_definition = shape_aspect shape_aspect => node_definition
node TO geometry_element (as location)	PATH			node <= node_representation <= representation representation.items[i] -> representation_item => geometric_representation_item => point
node TO fea_model (as model_ref)	PATH			node <= node_representation node_representation.model_ref -> fea_model
node TO (OPTIONAL) nodal_results_coordinate_system (as reults_coordinate_space)	PATH			node => node_with_solution_coordinate_system
NODE_DESCRIPTION	node_definition	104		
description	shape_aspect.description	41		node_definition <= shape_aspect shape_aspect.description

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
NODE_SHAPE_RELATIONSHIP	node_geometric_relationship	104		
role	analysis_item_within_\ representation.description	43		node_geometric_relationship node_geometric_relationship.item -> analysis_item_within_representation analysis_item_within_representation.description
node_shape_relationship TO node (as node_reference)	PATH			node_geometric_relationship node_geometric_relationship.node_ref -> node_representation
node_shape_relationship TO geometry_element FROM advanced_boundary_representation UoF OR FROM composite_constituent_representation UoF OR FROM faceted_boundary_representation UoF OR FROM fea_model UoF OR FROM manifold_surface_with_topology UoF OR FROM non_topological_surface_and_wireframe UoF OR FROM wireframe_with_topology UoF (as shape)	PATH			node_geometric_relationship node_geometric_relationship.item -> analysis_item_within_representation analysis_item_within_representation.item -> representation_item => geometric_representation_item
POINT_ELEMENT point_element <= element	point_element_representation	104		point_element_representation <= element_representation
point_element TO matrix (as associated_matrix)	PATH			point_element_representation point_element_representation.matrix_set[i] -> point_element_matrix
POINT_MODEL point_model <= geometric_model_representation	point_representation	209		point_representation <= shape_representation
SPECIFICATION	applied_document_reference	209		
specification_code	document.id	41		applied_document_reference <= document_reference document_reference.assigned_document -> document document.id
specification_source	document_reference.source	41		applied_document_reference <= document_reference document_reference.source

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
STIFFNESS_MATRIX #1: stiffness_matrix <= matrix as a point element nodal response matrix #2: stiffness_matrix <= matrix as a coefficient #3: stiffness_matrix <= matrix directly specified	#1 (grounded_spring) #2 (directionally_explicit_element_\ coefficient) #3 (explicit_element_matrix)	104 104 104		#1 (grounded_spring <= point_element_matrix) #2 (directionally_explicit_element_coefficient {directionally_explicit_element_coefficient directionally_explicit_element_coefficient.\ property_type = stiffness}) #3 (explicit_element_matrix {explicit_element_matrix explicit_element_matrix.property_type = stiffness})
SUBSTRUCTURE_ELEMENT substructure_element <= element	substructure_element_representation	104		substructure_element_representation <= element_representation
substructure_element TO fea_model (as substructure_model_ref)	PATH			substructure_element_representation substructure_element_representation.\ substructure_model_ref -> fea_model
SUBSTRUCTURE_NODE_RELATIONSHIP	substructure_node_relationship	43		
substructure_node_relationship TO node (as related_node)	PATH			substructure_node_relationship <= representation_relationship representation_relationship.rep_2 -> representation => node_representation
substructure_node_relationship TO node (as relating_node)	PATH			substructure_node_relationship <= representation_relationship representation_relationship.rep_1 -> representation => node_representation

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
SURFACE_ELEMENT #1: surface_element <= element as 2d axisymmetric #2: surface_element <= element as 2d plane #3: surface_element <= element as 3d	#1 (axisymmetric_surface_2d_\ element_representation) #2 (plane_surface_2d_element_\ representation) #3 (surface_3d_element_representation)	104 104 104		#1 (axisymmetric_surface_2d_element_representation <= element_representation) #2 (plane_surface_2d_element_representation <= element_representation) #3 (surface_3d_element_representation <= element_representation)
SURFACE_PROPERTY #1: angle property of a 2d axisymmetric element #2: depth property of a 2d plane element #3: surface property of an element	#1 (axisymmetric_2d_element_\ property) #2 (plane_2d_element_property) #3 (surface_element_property)	104 104 104		
surface_property TO (ONE OR MANY UNIQUE) surface_element (as defined_elements) #1: surface_element that is a 2d axisymmetric surface element #2: surface_element that is a 2d plane surface element #3: surface_element that is a 3d surface element #4: as an angle property to a surface_element that is a 2d axisymmetric surface element #5: as a depth property to a surface_element that is a 2d plane surface element	PATH			#1 (surface_element_property < - axisymmetric_surface_2d_element_representation.property axisymmetric_surface_2d_element_representation) #2 (surface_element_property < - plane_surface_2d_element_representation.property plane_surface_2d_element_representation) #3 (surface_element_property < - surface_3d_element_representation.property surface_3d_element_representation) #4 (axisymmetric_2d_element_property < - axisymmetric_surface_2d_element_representation.\ angle_property axisymmetric_surface_2d_element_representation) #5 (plane_2d_element_property < - plane_surface_2d_element_representation.depth_property plane_surface_2d_element_representation)

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
SURFACE_THICKNESS #1: Uniform surface thickness #2: Layered thickness	#1 (uniform_surface_section) #2 (uniform_surface_section_layered)	104 104		
surface_thickness TO length_measure (as thickness)	PATH			uniform_surface_section uniform_surface_section.thickness -> context_dependent_measure
surface_thickness TO surface_property (as property) #1: Uniform thickness as a constant field on a 3d surface element #2: Uniform thickness as a constant field on an axisymmetric 2d element #3: Uniform thickness as a constant field on an plane 2d element #4: Uniform thickness as a varying field on a 3d surface element #5: Uniform thickness as a varying field on an axisymmetric 2d element	PATH			#1 (uniform_surface_section <= surface_section < - surface_section_field_constant.definition surface_section_field_constant <= surface_section_field < - surface_element_property.section surface_element_property) #2 (uniform_surface_section <= surface_section < - surface_section_field_constant.definition surface_section_field_constant <= surface_section_field < - surface_element_property.section surface_element_property < - axisymmetric_surface_2d_element_representation.property axisymmetric_surface_2d_element_representation axisymmetric_surface_2d_element_representation.\ angle_property -> axisymmetric_2d_element_property) (*continued on the following page*)

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
surface_thickness TO surface_property (as property) (*continued*) #6: Uniform thickness as a varying field on an plane 2d element #7: Layered thickness as a constant field on a 3d surface element #8: Layered thickness as a constant field on an axisymmetric 2d element #9:Layered thickness as a constant field on an plane 2d element #10: Layered thickness as a varying field on a 3d surface element #11:Layered thickness as a varying field on an axisymmetric 2d element #12:Layered thickness as a varying field on an plane 2d element	PATH			#3 (uniform_surface_section <= surface_section < - surface_section_field_constant.definition surface_section_field_constant <= surface_section_field < - surface_element_property.section surface_element_property < - plane_surface_2d_element_representation.property plane_surface_2d_element_representation plane_surface_2d_element_representation.depth_property -> plane_2d_element_property) #4 (uniform_surface_section <= surface_section < - surface_section_field_varying.definitions[i] surface_section_field_varying <= surface_section_field < - surface_element_property.section surface_element_property) #5 (uniform_surface_section <= surface_section < - surface_section_field_varying.definitions[i] surface_section_field_varying <= surface_section_field < - surface_element_property.section surface_element_property < - axisymmetric_surface_2d_element_representation.property axisymmetric_surface_2d_element_representation axisymmetric_surface_2d_element_representation.\ angle_property -> axisymmetric_2d_element_property) (*continued on the following page*)

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
surface_thickness TO surface_property (as property) (*continued*)	PATH			#6 (uniform_surface_section <= surface_section < - surface_section_field_varying.definitions[i] surface_section_field_varying <= surface_section_field < - surface_element_property.section surface_element_property < - plane_surface_2d_element_representation.property plane_surface_2d_element_representation plane_surface_2d_element_representation.depth_property -> plane_2d_element_property) #7 (uniform_surface_section_layered <= surface_section < - surface_section_field_constant.definition surface_section_field_constant <= surface_section_field < - surface_element_property.section surface_element_property) #8 (uniform_surface_section_layered <= surface_section < - surface_section_field_constant.definition surface_section_field_constant <= surface_section_field < - surface_element_property.section surface_element_property < - axisymmetric_surface_2d_element_representation.property axisymmetric_surface_2d_element_representation axisymmetric_surface_2d_element_representation.\ angle_property -> axisymmetric_2d_element_property) #9 (uniform_surface_section_layered <= surface_section < - surface_section_field_constant.definition surface_section_field_constant <= surface_section_field < - surface_element_property.section surface_element_property < - plane_surface_2d_element_representation.property plane_surface_2d_element_representation plane_surface_2d_element_representation.depth_property -> plane_2d_element_property)

Table 12 - Mapping table for fea_model UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
surface_thickness TO surface_property (as property) (*continued*)	PATH			#10 (uniform_surface_section_layered <= surface_section < - surface_section_field_varying.definitions[i] surface_section_field_varying <= surface_section_field < - surface_element_property.section surface_element_property) #11 (uniform_surface_section_layered <= surface_section < - surface_section_field_varying.definitions[i] surface_section_field_varying <= surface_section_field < - surface_element_property.section surface_element_property < - axisymmetric_surface_2d_element_representation.property axisymmetric_surface_2d_element_representation axisymmetric_surface_2d_element_representation.\\ angle_property -> axisymmetric_2d_element_property) #12 (uniform_surface_section_layered <= surface_section < - surface_section_field_varying.definitions[i] surface_section_field_varying <= surface_section_field < - surface_element_property.section surface_element_property < - plane_surface_2d_element_representation.property plane_surface_2d_element_representation plane_surface_2d_element_representation.depth_property -> plane_2d_element_property)

Table 12 - Mapping table for fea_model UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
VOLUME_ELEMENT				
#1: volume_element <= element as 2d axisymmetric	#1 (axisymmetric_volume_2d_\ element_representation)	104		#1 (axisymmetric_volume_2d_element_representation <= element_representation)
#2: volume_element <= element as 2d plane	#2 (plane_volume_2d_element_\ representation)	104		#2 (plane_volume_2d_element_representation <= element_representation)
#3: volume_element <= element as 3d	#3 (volume_3d_element_\ representation)	104		#3 (volume_3d_element_representation <= element_representation)

Table 13 - Mapping table for manifold_surface_with_topology UoF

Application element	AIM element	Source	Rules	Reference Path
GEOMETRIC_MODEL_REPRESENTATION	shape_representation	41	21, 23, 25, 26, 47	
role	representation.name	43		shape_representation <= representation representation.name
geometric_model_representation TO (ONE OR MANY UNIQUE) geometry_element (as elements)	PATH			shape_representation <= representation representation.items -> representation_item => geometric_representation_item
GEOMETRY_ELEMENT	geometric_representation_item	43		
MANIFOLD_SURFACE_WITH_TOPOLOGY manifold_surface_with_topology <= geometric_model_representation	manifold_surface_shape_\ representation	509	47	manifold_surface_shape_representation <= shape_representation

Table 14 - Mapping table for material UoF

Application element	AIM element	Source	Rules	Reference Path
ADDITIONAL_DESIGN_INFORMATION	document	41	18	
additional_design_information TO (ONE OR MANY UNIQUE) specification (as additional_information)	PATH			document <- document_relationship.relating_document document_relationship document_relationship.related_document -> document <- document_reference.assigned_document document_reference => applied_document_reference
additional_design_information TO (ONE OR MANY UNIQUE) design_discipline_product_definition (as design)	PATH			document <- document_reference.assigned_document document_reference => applied_document_reference applied_document_reference.items[i] -> document_reference_item document_reference_item = product_definition product_definition
ANISOTROPIC_MATERIAL anisotropic_material <= stock_material	product_definition	41	45	product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category {product_category product_category.name = 'anisotropic material'}

Table 14 - Mapping table for material UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
DESIGN_SPECIFICATION design_specification <= specification	document	41	18	document {document document.kind -> document_type document_type.product_data_type = 'design specification'} document <- document_reference.assigned_document document_reference => applied_document_reference
DIRECTION	direction	42		
DISCONTINUOUS_FIBER_ASSEMBLY discontinuous_fiber_assembly <= stock_material	product_definition	41	29, 45	product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category {product_category product_category.name = 'discontinuous fiber assembly'}
ENVIRONMENT	data_environment	45		
FILAMENT_ASSEMBLY filament_assembly <= stock_material	product_definition	41	29, 45	product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category {product_category product_category.name = 'filament assembly'}

Table 14 - Mapping table for material UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
ISOTROPIC_MATERIAL isotropic_material <= stock_material	product_definition	41	29, 45	product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category {product_category product_category.name = 'isotropic material'}
MATERIAL_DIRECTION	material_property	45		
material_direction TO direction (as material_orientation)	PATH			material_property <= property_definition <- property_definition_representation.definition property_definition_representation {property_definition_representation => material_property_representation} property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => geometric_representation_item => direction
MATERIAL_PROPERTY	material_property	45		
property_name	property_definition.name	41		material_property <= property_definition property_definition.name
material_property TO environment (as ambient)	PATH			material_property <= property_definition < - property_definition_representation.definition property_definition_representation => material_property_representation material_property_representation.dependent_env -> data_environment

Table 14 - Mapping table for material UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
material_property TO (OPTIONAL) measure_value (as property_value)	PATH			material_property <= property_definition < - property_definition_representation.definition property_definition_representation {property_definition_representation => material_property_representation} property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => measure_representation_item
MATERIAL_SPECIFICATION material_specification <= specification	document	41	18	document {document document.kind -> document_type document_type.product_data_type = 'material specification'} document <- document_reference.assigned_document document_reference => applied_document_reference
PROCESS_SPECIFICATION process_specification <= specification	document	41	18	document {document document.kind -> document_type document_type.product_data_type = 'process specification'} document <- document_reference.assigned_document document_reference => applied_document_reference

Table 14 - Mapping table for material UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
SPECIFICATION	applied_document_reference	209		
specification_code	document.id	41		applied_document_reference <= document_reference document_reference.assigned_document -> document document.id
specification_source	document_reference.source	41		applied_document_reference <= document_reference document_reference.source
STOCK_CORE stock_core <= stock_material	product_definition	41	45	product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category {product_category product_category.name = 'stock core'}
STOCK_MATERIAL	product_definition	41		
stock_material TO (OPTIONAL) approval FROM authorization UoF (as approved_by)	PATH			product_definition approval_item = product_definition approval_item <- applied_approval_assignment.items[i] applied_approval_assignment
stock_material TO material_property (as property)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => material_property

Table 14 - Mapping table for material UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
stock_material TO material_direction (as reference_direction)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => material_property
stock_material TO part_version FROM part_identification UoF (as of_part)	PATH			product_definition product_definition.formation -> product_definition_formation
stock_material TO material_specification (as specified_material)	PATH			product_definition document_reference_item = product_definition document_reference_item <- applied_document_reference.items[i] applied_document_reference
SURFACE_FINISH_SPECIFICATION surface_finish_specification <= specification	document	41	18	document {document document.kind -> document_type document_type.product_data_type = \\ 'surface finish specification'} document <- document_reference.assigned_document document_reference => applied_document_reference

Table 14 - Mapping table for material UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
USAGE_CONSTRAINT	document_usage_constraint	41		
usage_element	document_usage_constraint.\ subject_element	41		
usage_value	document_usage_constraint.\ subject_element_value	41		
usage_constraint TO specification (as constrains)	PATH			document_usage_constraint document_usage_constraint.source -> document <- document_reference.assigned_document document_reference => applied_document_reference

Table 15 - Mapping table for non_topological_surface_and_wireframe UoF

Application element	AIM element	Source	Rules	Reference Path
GEOMETRIC_MODEL_REPRESENTATION	shape_representation	41	21, 23, 25, 26, 47	
role	representation.name	43		shape_representation <= representation representation.name
geometric_model_representation TO (ONE OR MANY UNIQUE) geometry_element (as elements)	PATH			shape_representation <= representation representation.items -> representation_item => geometric_representation_item
GEOMETRY_ELEMENT	geometric_representation_item	43		
NON_TOPOLOGICAL_SURFACE_AND_WIREFRAME #1:non_topological_surface_and_wireframe <= geometric_model_representation as surface model #2:non_topological_surface_and_wireframe <= geometric_model_representation as wireframe model	#1 (geometrically_bounded_surface_\ shape_representation) #2 geometrically_bounded_wireframe_\ shape_representation)	507 510	47 47	#1 (geometrically_bounded_surface_shape_\ representation <= shape_representation) #2 (geometrically_bounded_wireframe_shape_\ representation <= shape_representation)

Table 16 - Mapping table for part_composite_constituents UoF

Application element	AIM element	Source	Rules	Reference Path
COMPOSITE_ASSEMBLY composite_assembly <= constituent_part	product_definition	41	8	product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category {product_category product_category.name = 'composite assembly'}
composite_assembly TO composite_assembly_table FROM part_laminate_table UoF (as layup_part)	PATH			product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => make_from_usage_option} product_definition_relationship.related_product_definition -> product_definition => composite_assembly_definition

Table 16 - Mapping table for part_composite_constituent UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
composite_assembly TO (OPTIONAL) advanced_boundary_representation FROM advanced_boundary_representation UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation => advanced_brep_shape_representation

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
composite_assembly TO (OPTIONAL) faceted_boundary_representation FROM faceted_boundary_representation UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation => faceted_brep_shape_representation

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
composite_assembly TO (OPTIONAL) manifold_surface_with_topology FROM manifold_surface_with_topology UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation => manifold_surface_shape_representation

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
composite_assembly TO (OPTIONAL) non_topological_surface_and_wireframe as a surface model FROM non_topological_surface_and_wireframe UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation => geometrically_bounded_surface_shape_representation

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
composite_assembly TO (OPTIONAL) non_topological_surface_and_wireframe as a wireframe model FROM non_topological_surface_and_wireframe UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation => geometrically_bounded_wireframe_shape_representation

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
composite_assembly TO (OPTIONAL) three_d_geometry_set FROM composite_constituent_representation UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation => geometrically_bounded_surface_shape_representation

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
composite_assembly TO (OPTIONAL) wireframe_with_topology as edge based wireframe FROM wireframe_with_topology UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation => edge_based_wireframe_shape_representation

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
composite_assembly TO (OPTIONAL) wireframe_with_topology as shell based wireframe FROM wireframe_with_topology UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation => shell_based_wireframe_shape_representation

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
CONSTITUENT_PART	product_definition	41		
constituent_part_identification	product_definition.id	41		
constituent_part TO part_version FROM part_identification UoF (as of_part)	PATH			product_definition product_definition.formation -> product_definition_formation
constituent_part TO (OPTIONAL) mass_measure (as weight)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition {property_definition => material_property} property_definition <- property_definition_representation.definition property_definition_representation {property_definition_representation => material_property_representation} property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => measure_representation_item <= measure_with_unit => mass_measure_with_unit

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
CURVE #1: as a parametric curve #2: as curve on a surface #3: as a composite curve on a surface	#1 (pcurve) #2 (surface_curve) #3 (composite_curve_on_surface)	42 42 42		
DIRECTION	direction	42		
FILAMENT_LAMINATE filament_laminate <= constituent_part	product_definition	41	8, 48	product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category {product_category product_category.name = 'filament laminate'}
filament_laminate TO filament_assembly FROM material UoF (as made_from)	PATH			product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => make_from_usage_option} product_definition_relationship.related_product_definition -> product_definition

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
filament_laminate TO (OPTIONAL) filament_laminate_shape FROM composite_constituent_representation UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect
LENGTH_MEASURE	length_measure_with_unit	41		
MASS_MEASURE	mass_measure_with_unit	41		
PLY ply <= constituent_part	product_definition	41	8, 28, 29, 48	product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category {product_category product_category.name = 'ply'}

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
ply TO (ZERO, ONE OR MANY ORDERED) ply_piece (as constituents)	PATH			product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition -> product_definition
ply TO ply_orientation_angle (as material_orientation)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_product_definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation <- representation_relationship.rep_1 representation_relationship => representation_relationship_with_transformation representation_relationship_with_transformation.\ transformation_operator -> transformation transformation = item_defined_transformation item_defined_transformation => transformation_with_derived_angle

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
ply TO discontinuous_fiber_assembly FROM material UoF (as material_type)	PATH		29	product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => make_from_usage_option} product_definition_relationship.related_product_definition -> product_definition
ply TO filament_assembly FROM material UoF (as material_type)	PATH		29	product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => make_from_usage_option} product_definition_relationship.related_product_definition -> product_definition
ply TO isotropic_material FROM material UoF (as material_type)	PATH		29	product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => make_from_usage_option} product_definition_relationship.related_product_definition -> product_definition

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
ply TO length_measure (as ply_thickness)	PATH		19	product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_product_definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => measure_representation_item <= measure_with_unit {measure_with_unit measure_with_unit.value_component -> measure_value measure_value = positive_length_measure} {measure_with_unit measure_with_unit.unit_component -> unit unit = named_unit named_unit => length_unit} measure_with_unit => length_measure_with_unit
ply TO (OPTIONAL) ply_shape FROM compostie_consituent_representation UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
PLY_LAMINATE ply_laminate <= constituent_part	product_definition	41	8	product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category {product_category product_category.name = 'ply laminate'}
ply_laminate TO ply_laminate_table FROM part_laminate_table UoF (as ply_table)	PATH			product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => make_from_usage_option} product_definition_relationship.related_product_definition -> product_definition => ply_laminate_definition

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
ply_laminate TO (OPTIONAL) composite_sheet_representation FROM composite_constituent_representation UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation => composite_sheet_representation

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
ply_laminate TO (OPTIONAL) three_d_geometry_set FROM composite_constituent_representation UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation => geometrically_bounded_surface_shape_representation
PLY_ORIENTATION_ANGLE	transformation_with_derived_angle	209		
ply_orientation_angle to direction (as angle_reference)	PATH			transformation_with_derived_angle <= item_defined_transformation item_defined_transformation.transform_item_2 -> representation_item => geometric_representation_item => direction
ply_orientation_angle to curve (as angle_reference)	PATH			transformation_with_derived_angle <= item_defined_transformation item_defined_transformation.transform_item_2 -> representation_item => geometric_representation_item => curve

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
ply_orientation_angle to point_path (as angle_reference)	PATH			transformation_with_derived_angle <= item_defined_transformation item_defined_transformation.transform_item_2 -> representation_item => geometric_representation_item => direction {direction <= geometric_representation_item <= representation_item <- representation.items[i] representation {representation => shape_representation => point_and_vector} representation <- representation_relationship.rep_2 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_1 -> representation => shape_representation => point_path}
ply_orientation_angle to reinforcement_orientation_basis (as basis)	PATH			transformation_with_derived_angle <= item_defined_transformation item_defined_transformation.transform_item_1 -> representation_item => geometric_representation_item => placement => axis2_placement_3d

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
PLY_PIECE ply_piece <= constituent_part	product_definition	41		product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category {product_category product_category.name = 'ply piece'}
warp_surface #1: Direction of the warp_surface away from the basis surface #2: Direction of the warp_surface is towards the basis surface #3: Direction of the warp_surface is unknown	property_definition.name	41		product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition #1 ([property_definition.name = 'true'] [property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect]) #2 ([property_definition.name = 'false'] [property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect]) #3 ([property_definition.name = 'unknown'] [property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect])
ply_piece TO (OPTIONAL) ply_shape FROM composite_constituent_representation UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
POINT	cartesian_point	42		
POINT_AND_VECTOR	point_and_vector	209		point_and_vector <= shape_representation <= representation
point_and_vector TO point (as location)	PATH			point_and_vector <= shape_representation <= representation representation.items[1] -> representation_item => geometric_representation_item => point
point_and_vector TO (EXACTLY TWO ORDERED) direction (as vector) #1: first (major) direction #2: second (minor) direction	PATH			point_and_vector <= shape_representation <= representation #1 [representation.items[2] -> representation_item => geometric_representation_item => direction] #2 [representation.items[3] -> representation_item => geometric_representation_item => direction]

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
POINT_PATH	point_path	209		point_path <= shape_representation <= representation
point_path TO (ONE OR MANY ORDERED) point_and_vector (as directions)	PATH			point_path <= shape_representation <= representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation => shape_representation => point_and_vector
PROCESSED_CORE processed_core <= constituent_part	product_definition	41	8, 48	product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category {product_category product_category.name = 'processed core'}
processed_core TO (ZERO, ONE OR MANY ORDERED) ply (as added_material)	PATH			product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition -> product_definition

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
processed_core TO ply_orientation_angle (as cell_direction)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_product_definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation <- representation_relationship.rep_1 representation_relationship => representation_relationship_with_transformation representation_relationship_with_transformation.\ transformation_operator -> transformation transformation = item_defined_transformation item_defined_transformation => transformation_with_derived_angle
processed_core TO (ZERO, ONE OR MANY UNIQUE) processed_core (as made_from_processed_core)	PATH			product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition -> product_definition
processed_core TO stock_core FROM material UoF (as made_from_stock)	PATH			product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => make_from_usage_option} product_definition_relationship.related_product_definition -> product_definition

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
processed_core TO (OPTIONAL) advanced_boundary_representation FROM advanced_boundary_representation UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation => advanced_brep_shape_representation

Table 16 - Mapping table for part_composite_constituents UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
processed_core TO (OPTIONAL) beveled_sheet_representation FROM composite_constituent_representation UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation => beveled_sheet_representation

Table 16 - Mapping table for part_composite_constituents UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
processed_core TO (OPTIONAL) faceted_boundary_representation FROM faceted_boundary_representation UoF (as shape)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation => faceted_brep_shape_representation

Table 17 - Mapping table for part_identification UoF

Application element	AIM element	Source	Rules	Reference Path
ALTERNATE_PART	product	41		
alternate_part to part (as alternate)	PATH			product <- alternate_product_relationship.alternate alternate_product_relationship alternate_product_relationship.base -> product
ANALYSIS	product	41		
analysis_type	product_category.name	41		product <- product_related_product_category.products[i] product_related_product_category <= product_category product_category.name
analysis TO person_organization (as owner)	PATH			(product person_organization_item = product person_organization_item <- applied_person_and_organization_assignment.items[i] applied_person_and_organization_assignment) (product organization_item = product organization_item <- applied_organization_assignment.items[i] applied_organization_assignment)
ANALYSIS_DESIGN_VERSION_RELATIONSHIP	product_definition_formation_\ relationship	41		
analysis_design_version_relationship TO analysis_version (as analysis)	PATH			product_definition_formation_relationship product_definition_formation_relationship.\ related_product_definition_formation -> product_definition_formation
analysis_design_version_relationship TO part_version (as design)	PATH			product_definition_formation_relationship product_definition_formation_relationship.\ relating_product_definition_formation -> product_definition_formation
ANALYSIS_DISCIPLINE_PRODUCT_DEFINITION	product_definition	41	31, 32, 43	
description	product_definition.description	41		

Table 17 - Mapping table for part_identification UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
discipline_identification	application_context_element.name	41	43	product_definition product_definition.frame_of_reference -> product_definition_context {product_definition_context product_definition_context.life_cycle_stage = 'analysis'} product_definition_context <= application_context_element application_context_element.name
analysis_discipline_product_definition TO (OPTIONAL) approval FROM authorization UoF (as approved_by)	PATH			product_definition approval_item = product_definition approval_item <- applied_approval_assignment.items[i] applied_approval_assignment
analysis_discipline_product_definition TO (OPTIONAL) file (as cae_filename)	PATH		18	product_definition => product_definition_with_associated_documents product_definition_with_associated_documents.\ documentation_ids[i] -> document {document document.kind -> document_type document_type.product_data_type = 'cae file'}

Table 17 - Mapping table for part_identification UoF

Application element	AIM element	Source	Rules	Reference Path
analysis_discipline_product_definition to date (as creation_date)	PATH		16, 17, 31, 40	(product_definition date_and_time_item = product_definition date_and_time_item <- applied_date_and_time_assignment.items[i] applied_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment date_and_time_assignment.role -> date_time_role date_time_role.name = 'creation date'} date_and_time_assignment.assigned_date_and_time -> date_and_time) (product_definition date_item = product_definition date_item <- applied_date_assignment.items[i] applied_date_assignment <= date_assignment {date_assignment date_assignment.role -> date_role date_role.name = 'creation date'} date_assignment.assigned_date -> date)
analysis_discipline_product_definition TO person_organization FROM authorization UoF (as creator)	PATH		32	(product_definition person_organization_item = product_definition person_organization_item <- applied_person_and_organization_assignment.items[i] applied_person_and_organization_assignment) (product_definition organization_item = product_definition organization_item <- applied_organization_assignment.items[i] applied_organization_assignment)
analysis_discipline_product_definition TO analysis_version (as version)	PATH			product_definition product_definition.formation -> product_definition_formation

Table 17 - Mapping table for part_identification UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
ANALYSIS_VERSION	product_definition_formation	41	35, 36, 37	
revision_letter	product_definition_formation.id	41		
contract_number (OPTIONAL)	contract.name	41	13	product_definition_formation contract_item = product_definition_formation contract_item <- applied_contract_assignment.items[i] applied_contract_assignment <= contract_assignment contract_assignment.assigned_contract -> contract contract.name
release_status	approval_status	41	11, 39	product_definition_formation approval_item = product_definition_formation approval_item <- applied_approval_assignment.items[i] applied_approval_assignment <= approval_assignment approval_assignment.assigned_approval -> approval approval.status -> approval_status
security_code	security_classification_level.name	41	24, 37, 44	product_definition_formation security_classification_item = product_definition_formation security_classification_item <- applied_security_classification.items[i] applied_security_classification <= security_classification_assignment security_classification_assignment.\ applied_security_classification -> security_classification security_classification.security_level -> security_classification_level security_classification_level.name
analysis_version TO analysis (as analysis_number)	PATH			product_definition_formation product_definition_formation.of_product -> product

Table 17 - Mapping table for part_identification UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
analysis_version TO (OPTIONAL) approval FROM authorization UoF (as approved_by)	PATH			product_definition_formation approval_item = product_definition_formation approval_item <- applied_approval_assignment.items[i] applied_approval_assignment
analysis_version TO person_organization FROM authorization UoF (as creator)		41	35	(product_definition_formation person_organization_item = product_definition_formation person_organization_item <- applied_person_and_organization_assignment.items[i] applied_person_and_organization_assignment) (product_definition_formation organization_item = product_definition_formation organization_item <- applied_organization_assignment.items[i] applied_organization_assignment)
DATE	(date_and_time) (date)	41 41	14, 15, 40	
DESIGN_DISCIPLINE_PRODUCT_DEFINITION	product_definition	41	31, 32, 43	
description	product_definition.description	41		
discipline_identification	application_context_element.name	41	43	product_definition product_definition.frame_of_reference -> product_definition_context {product_definition_context product_definition_context.life_cycle_stage = 'design'} product_definition_context <= application_context_element application_context_element.name
design_discipline_product_definition TO (OPTIONAL) approval FROM authorization UoF (as approved_by)	PATH			product_definition approval_item = product_definition approval_item <- applied_approval_assignment.items[i] applied_approval_assignment

Table 17 - Mapping table for part_identification UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
design_discipline_product_definition TO date (as creation_date)	PATH		16, 17, 31, 40	(product_definition date_and_time_item = product_definition date_and_time_item <- applied_date_and_time_assignment.items[i] applied_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment date_and_time_assignment.role -> date_time_role date_time_role.name = 'creation date'} date_and_time_assignment.assigned_date_and_time -> date_and_time) (product_definition date_item = product_definition date_item <- applied_date_assignment.items[i] applied_date_assignment <= date_assignment {date_assignment date_assignment.role -> date_role date_role.name = 'creation date'} date_assignment.assigned_date -> date)
design_discipline_product_definition TO (OPTIONAL) file (as cad_filename)	PATH		18	product_definition => product_definition_with_associated_documents product_definition_with_associated_documents.\ documentation_ids[i] -> document {document document.kind -> document_type document_type.product_data_type = 'cad file'}

Table 17 - Mapping table for part_identification UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
design_discipline_product_definition TO person_organization FROM authorization UoF (as creator)	PATH		32	(product_definition person_organization_item = product_definition person_organization_item <- applied_person_and_organization_assignment.items[i] applied_person_and_organization_assignment) (product_definition organization_item = product_definition organization_item <- applied_organization_assignment.items[i] applied_organization_assignment)
design_discipline_product_definition TO part_version (as version)	PATH			product_definition product_definition.formation -> product_definition_formation
DESIGN_MATERIAL	make_from_usage_option	44		
design_material TO anisotropic_material FROM material UoF (as material_callout)	PATH			make_from_usage_option <= product_definition_usage <= product_definition_relationship product_definition_relationship.related_product_definition -> product_definition
design_material TO isotropic_material FROM material UoF (as material_callout)	PATH			make_from_usage_option <= product_definition_usage <= product_definition_relationship product_definition_relationship.related_product_definition -> product_definition
design_material TO laminate_table FROM part_laminate_table UoF OR FROM zone_composite_constituents_and_their_representation UoF (as material_callout)	PATH			make_from_usage_option <= product_definition_usage <= product_definition_relationship product_definition_relationship.related_product_definition -> product_definition
design_material TO design_discipline_product_definition (as part_defined)	PATH			make_from_usage_option <= product_definition_usage <= product_definition_relationship product_definition_relationship.relating_product_definition -> product_definition
FILE	document	41	18	
MASS_MEASURE	mass_measure_with_unit	41		

Table 17 - Mapping table for part_identification UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
PART	product	41	33, 34, 35, 43	
part_classification (OPTIONAL)	product_category.name	41	42	product <- product_related_product_category.products[i] product_related_product_category <= product_category product_category.name
part_nomenclature	product.name	41		
part_number (UNIQUE)	product.id	41		
part_type	product_category.name	41	34, 42	product <- product_related_product_category.products[i] product_related_product_category <= product_category product_category.name
standard_part	product_category	41	42	product <- product_related_product_category.products[i] product_related_product_category <= product_category <- product_category_relationship.sub_category product_category_relationship product_category_relationship.category -> product_category product_category.name = ’standard part’

Table 17 - Mapping table for part_identification UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
part TO person_organization (as owner)	PATH			(product person_organization_item = product person_organization_item <- applied_person_and_organization_assignment.items[i] applied_person_and_organization_assignment) (product organization_item = product organization_item <- applied_organization_assignment.items[i] applied_organization_assignment)
PART_VERSION	product_definition_formation	41	35, 36, 37	
contract_number (OPTIONAL)	contract.name	41	13	product_definition_formation contract_item = product_definition_formation contract_item <- applied_contract_assignment.items[i] applied_contract_assignment <= contract_assignment contract_assignment.assigned_contract -> contract contract.name
make_or_buy_code (OPTIONAL)	product_definition_formation_with_\ specified_source.make_or_buy	41		product_definition_formation => product_definition_formation_with_specified_source product_definition_formation_with_specified_source.\ make_or_buy
release_status	approval_status	41	11, 39	product_definition_formation approval_item = product_definition_formation approval_item <- applied_approval_assignment.items[i] applied_approval_assignment <= approval_assignment approval_assignment.assigned_approval -> approval approval.status -> approval_status
revision_letter	product_definition_formation.id	41		

Table 17 - Mapping table for part_identification UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
security_code	security_classification_level.name	41	24, 37, 44	product_definition_formation security_classification_item = product_definition_formation security_classification_item <- applied_security_classification.items[i] applied_security_classification <= security_classification_assignment security_classification_assignment.\ applied_security_classification -> security_classification security_classification.security_level -> security_classification_level security_classification_level.name
part_version TO (OPTIONAL) approval FROM authorization UoF (as approved_by)	PATH			product_definition_formation approval_item = product_definition_formation approval_item <- applied_approval_assignment.items[i] applied_approval_assignment
part_version TO person_organization FROM authorization UoF (as creator)	PATH		35	(product_definition_formation person_organization_item = product_definition_formation person_organization_item <- applied_person_and_organization_assignment.items[i] applied_person_and_organization_assignment) (product_definition_formation organization_item = product_definition_formation organization_item <- applied_organization_assignment.items[i] applied_organization_assignment)
part_version TO part (as part_number)	PATH		35	product_definition_formation product_definition_formation.of_product -> product

Table 17 - Mapping table for part_identification UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
part_version TO (OPTIONAL) mass_measure (as weight)	PATH			product_definition_formation <- product_definition.formation product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition {property_definition => material_property} property_definition <- property_definition_representation.definition property_definition_representation {property_definition_representation => material_property_representation} property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => measure_representation_item <= measure_with_unit => mass_measure_with_unit

Table 18 - Mapping table for part_laminate_table UoF

Application element	AIM element	Source	Rules	Reference Path
COMPOSITE_ASSEMBLY_SEQUENCE_DEFINITION	composite_assembly_sequence_\ definition	209	8	
composite_assembly_sequence_definition TO (ONE OR MANY) composite_assembly FROM part_composite_constituents UoF (as components_in_sequence)	PATH			product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition -> product_definition
composite_assembly_sequence_definition TO (ONE OR MANY) filament_laminate FROM part_composite_constituents UoF (as components_in_sequence)	PATH			product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition -> product_definition
composite_assembly_sequence_definition TO (ONE OR MANY) ply FROM part_composite_constituents UoF (as components_in_sequence)	PATH			product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition -> product_definition
composite_assembly_sequence_definition TO (ONE OR MANY) ply_laminate FROM part_composite_constituents UoF (as components_in_sequence)	PATH			product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition -> product_definition

Table 18 - Mapping table for part_laminate_table UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
composite_assembly_sequence_definition TO (ONE OR MANY) processed_core FROM part_composite_constituents UoF (as components_in_sequence)	PATH			product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition -> product_definition
composite_assembly_sequence_definition TO (OPTIONAL) fea_material_property FROM fea_model UoF (as properties)	PATH			composite_assembly_sequence_definition <= product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition {property_definition => material_property} property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => material_property_representation => fea_material_property_representation} property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => (fea_material_property_representation_item) (measure_representation_item)

Table 18 - Mapping table for part_laminate_table UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
COMPOSITE_ASSEMBLY_TABLE #1: composite_assembly_table <= laminate_table #2: composite_assembly_table <= part_laminate_table	composite_assembly_definition	209		#1 (composite_assembly_definition <= product_definition) #2 (composite_assembly_definition <= product_definition)
composite_assembly_table TO (TWO OR MANY ORDERED) composite_assembly_sequence_definition (as sequence_groups)	PATH			composite_assembly_definition <= product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition product_definition => composite_assembly_sequence_definition
DIRECTION	direction	42		
LAMINATE_TABLE	product_definition	41		
laminate_table TO (OPTIONAL) surface_with_direction (as base_surface)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect {shape_aspect.name = 'base surface'} shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation

Table 18 - Mapping table for part_laminate_table UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
laminate_table TO reinforcement_orientation_basis (as basis)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => geometric_representation_item => placement => axis2_placement_3d
laminate_table TO (OPTIONAL) fea_material_property FROM fea_model UoF (as properties)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition {property_definition => material_property} property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => material_property_representation => fea_material_property_representation} property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => (fea_material_property_representation_item) (measure_representation_item)

Table 18 - Mapping table for part_laminate_table UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
laminate_table TO (OPTIONAL) surface (as resulting_surface)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect {shape_aspect.name = 'resulting surface'} shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation {representation => shape_representation} representation.items[i] -> representation_item => geometric_representation_item => surface
LOCATION	cartesian_point	42		
PART_LAMINATE_TABLE part_laminate_table <= laminate_table	product_definition	41		product_definition
PLY_LAMINATE_SEQUENCE_DEFINITION	ply_laminate_sequence_definition	209	28	
ply_laminate_sequence_definition TO (ONE OR MANY UNIQUE) ply FROM part_composite_constituents UoF (as plies_in_sequence)	PATH		28	ply_laminate_sequence_definition <= product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition -> product_definition

Table 18 - Mapping table for part_laminate_table UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
ply_laminate_sequence_definition TO (OPTIONAL) fea_material_property FROM fea_model UoF (as properties)	PATH			ply_laminate_sequence_definition <= product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition {property_definition => material_property} property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => material_property_representation => fea_material_property_representation} property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => #1 (fea_material_property_representation_item) #2 (measure_property_representation_item)
PLY_LAMINATE_TABLE #1: ply_laminate_table <= laminate_table #2: ply_laminate_table <= part_laminate_table	ply_laminate_definition	209		#1 (ply_laminate_definition <= product_definition) #2 (ply_laminate_definition <= product_definition)
ply_laminate_table TO (TWO OR MANY ORDERED) ply_laminate_sequence_definition (as sequence)	PATH			ply_laminate_definition <= product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition product_definition => ply_laminate_sequence_definition
REINFORCEMENT_ORIENTATION_BASIS	axis2_placement_3d	42		

Table 18 - Mapping table for part_laminate_table UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
reinforcement_orientation_basis TO location (as basis_location)	PATH			axis2_placement_3d <= placement placement.location -> cartesian_point
reinforcement_orientation_basis TO location (as orientation)	PATH			axis2_placement_3d axis2_placement_3d.axis -> direction
SURFACE	surface	42		
SURFACE_WITH_DIRECTION	shape_representation	41		
surface_with_direction TO surface (as defining_surface)	PATH			shape_representation <= representation representation.items[1] -> representation_item => geometric_representation_item => surface
surface_with_direction TO direction (as material_direction)	PATH			shape_representation <= representation representation.items[2] -> representation_item => geometric_representation_item => direction

Table 19 - Mapping table for part_shape UoF

Application element	AIM element	Source	Rules	Reference Path
ANALYSIS_SHAPE	shape_definition_representation	41		
analysis_shape TO analysis_discipline_product_definition (as analysis_view)	PATH			shape_definition_representation <= property_definition_representation property_definition_representation.definition -> property_definition {property_definition => product_definition_shape} property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition = product_definition product_definition
IDEALIZED_ANALYSIS_SHAPE #1: idealized_analysis_shape <= analysis_shape #2: idealized_analysis_shape <= shape	#1 (shape_definition_representation) #2 (shape_definition_representation)	41 41		#1 (shape_definition_representation #2 (shape_definition_representation) {shape_definition_representation <= property_definition_representation property_definition_representation.used_representation -> representation {representation => shape_representation} representation.name = 'idealized analysis shape'}

Table 19 - Mapping table for part_shape UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
idealized_analysis_shape TO nominal_design_shape (as basis)	PATH			shape_definition_representation <= property_definition_representation property_definition_representation.used_representation -> representation {representation =>shape_representation} representation < - representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation {representation =>shape_representation} representation <- property_definition_representation.used_representation property_definition_representation => shape_definition_representation
idealized_analysis_shape TO geometric_model_representation FROM advanced_boundary_representation UoF OR FROM composite_constituent_representation UoF OR FROM faceted_boundary_representation UoF OR FROM fea_model UoF OR FROM manifold_surface_with_topology UoF OR FROM non_topological_surface_and_wireframe UoF OR FROM wireframe_with_topology UoF (as defining_shape)	PATH			shape_definition_representation <= property_definition_representation property_definition_representation.used_representation -> representation => shape_representation
NODE_SHAPE #1: node_shape <= analysis_shape #2: node_shape <= shape	#1 (shape_definition_representation) #2 (shape_definition_representation)	41 41		#1 (shape_definition_representation) #2 (shape_definition_representation) {shape_definition_representation <= property_definition_representation property_definition_representation.used_representation -> representation {representation => shape_representation} representation.name = 'node shape'}

Table 19 - Mapping table for part_shape UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
node_shape TO point_model FROM fea_model UoF (as defining_shape)	PATH			shape_definition_representation <= property_definition_representation property_definition_representation.used_representation -> representation => shape_representation => point_representation
NOMINAL_DESIGN_SHAPE nominal_design_shape <= shape	shape_definition_representation	41		shape_definition_representation {shape_definition_representation <= property_definition_representation property_definition_representation.used_representation -> representation {representation => shape_representation} representation.name = 'nominal design shape'}
nominal_design_shape TO geometric_model_representation FROM advanced_boundary_representation UoF OR FROM composite_constituent_representation UoF OR FROM faceted_boundary_representation UoF OR FROM fea_model UoF OR FROM manifold_surface_with_topology UoF OR FROM non_topological_surface_and_wireframe UoF OR FROM wireframe_with_topology UoF (as defining_shape)	PATH			shape_definition_representation <= property_definition_representation property_definition_representation.used_representation -> representation => shape_representation
nominal_design_shape TO design_discipline_product_definition FROM part_identification UoF (as design_view)	PATH			shape_definition_representation <= property_definition_representation property_definition_representation.definition -> property_definition {property_definition => product_definition_shape} property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition = product_definition product_definition

Table 19 - Mapping table for part_shape UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
SHAPE	shape_definition_representation	41		
role #1: as nominal design shape #2: as idealized analysis shape #3: as node shape	property_definition.name	41		shape_definition_representation <= property_definition_representation property_definition_representation.used_representation -> representation {representation => shape_representation} #1 (representation.name = 'nominal design shape') #2 (representation.name = 'idealized analysis shape') #3 (representation.name = 'node shape')
SHAPE_ASPECT	shape_aspect	41		
shape_aspect TO (ZERO, ONE OR MANY UNIQUE) specification (as characteristics)	PATH		47	shape_aspect document_reference_item = shape_aspect document_reference_item <- applied_document_reference.items[i] application_document_reference
shape_aspect TO geometric_model_representation FROM advanced_boundary_representation UoF OR FROM composite_constituent_representation UoF OR FROM faceted_boundary_representation UoF OR FROM fea_model UoF OR FROM manifold_surface_with_topology UoF OR FROM non_topological_surface_and_wireframe UoF OR FROM wireframe_with_topology UoF (as geometry)	PATH		47	shape_aspect shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation property_definition_representation.used_representation -> representation => shape_representation
shape_aspect TO shape (as parent_shape)	PATH			shape_aspect shape_aspect.of_shape -> product_definition_shape <= property_definition <- property_definition_representation.definition property_definition_representation => shape_definition_representation

Table 20 - Mapping table for wire_frame_with_topology UoF

Application element	AIM element	Source	Rules	Reference Path
GEOMETRIC_MODEL_REPRESENTATION	shape_representation	41	21, 23, 25, 26, 47	
role	representation.name	43		shape_representation <= representation representation.name
geometric_model_representation TO (ONE OR MANY UNIQUE) geometry_element (as elements)	PATH			shape_representation <= representation representation.items -> representation_item => geometric_representation_item
GEOMETRY_ELEMENT	geometric_representation_item	43		
WIREFRAME_WITH_TOPOLOGY #1: wireframe_with_topology <= geometric_model_representation as edge based wireframe model #2: wireframe_with_topology <= geometric_model_representation as shell based wireframe model	#1 (edge_based_wireframe_shape_\ representation) #2 (shell_based_wireframe_shape_\ representation)	501 502	47 47	#1 (edge_based_wireframe_representation <= shape_representation) #2 (shell_based_wireframe_representation <= shape_representation)

Table 21 - Mapping table for zone_composite_constituents_and_their_representation UoF

Application element	AIM element	Source	Rules	Reference Path
ANGLE_MEASURE	plane_angle_measure	41		
BOUNDARY_CURVE_REPRESENTATION	geometric_set	42		
boundary_curve_representation TO (ONE OR MANY UNIQUE) curve (as bounds) #1: represent boundary as a parametric curve #2: represent boundary as a curve on a surface #3: represent boundary as a composite curve on a surface	PATH			geometric_set geometric_set[i].elements -> geometric_set_select = curve #1 (curve => pcurve) #2 (curve => surface_curve) #3 (curve => bounded_curve => composite_curve => composite_curve_on_surface)
CURVE #1: as a parametric curve #2: as curve on a surface #3: as a composite curve on a surface	#1 (pcurve) #2 (surface_curve) #3 (composite_curve_on_surface)	42 42 42		
DIRECTION	direction	42		
DRAPED_ORIENTATION_ANGLE draped_orientation_angle <= ply_orientation_angle	draped_defined_transformation	209		draped_defined_transformation <= item_defined_transformation => transformation_with_derived_angle
EDGE_ZONE_SHAPE edge_zone_shape <= zone_structural_makeup_shape_representation	shape_representation	41		shape_representation
edge_zone_shape TO boundary_curve_representation (as boundary)	PATH			shape_representation <= representation representation.items[i]-> representation_item => geometric_representation_item => geometric_set
edge_zone_shape TO loop (as boundary)	PATH			shape_representation <= representation representation.items[i]-> representation_item => topological_representation_item => loop

Table 21 - Mapping table for zone_composite_constituents_and_their_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
LAID_ORIENTATION_ANGLE laid_orientation_angle <= ply_orientation_angle	laid_defined_transformation	209		laid_defined_transformation <= item_defined_transformation => transformation_with_derived_angle
LAMINATE_TABLE	product_definition	41		
laminate_table TO surface_with_direction (as base_surface)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect {shape_aspect.name = 'base surface'} shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation
laminate_table TO reinforcement_orientation_basis (as basis)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => geometric_representation_item => placement => axis2_placement_3d

Table 21 - Mapping table for zone_composite_constituents_and_their_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
laminate_table TO (OPTIONAL) fea_material_property FROM fea_model UoF (as properties)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition {property_definition => material_property} property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => material_property_representation => fea_material_property_representation} property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => (fea_material_property_representation_item) (measure_representation_item)

Table 21 - Mapping table for zone_composite_constituents_and_their_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
laminate_table TO (OPTIONAL) surface (as resulting_surface)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect {shape_aspect.name = 'resulting surface'} shape_definition = shape_aspect characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation {representation => shape_representation} representation.items[i] -> representation_item => geometric_representation_item => surface
LENGTH_MEASURE	length_measure_with_unit	41		
LOCATION	cartesian_point	42		
LOOP	loop	42		
PERCENTAGE	ratio_measure	41		
PERCENTAGE_LAMINATE_TABLE percentage_laminate_table <= zone_structural_makeup	percentage_laminate_definition	209		percentage_laminate_definition <= product_definition

Table 21 - Mapping table for zone_composite_constituents_and_their_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
percentage_laminate_table TO (ONE OR MANY UNIQUE) percentage_ply (as table_components)	PATH			ply_percentage_laminate_definition <= product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition -> product_definition => percentage_ply_definition
percentage_laminate_table TO length_measure (as total_thickness)	PATH		19	percentage_laminate_definition <= product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized _definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => measure_representation_item <= measure_with_unit {measure_with_unit [measure_with_unit.value_component -> measure_value measure_value = positive_length_measure] [measure_with_unit.unit_component -> unit unit = named_unit named_unit => length_unit]} measure_with_unit => length_measure_with_unit

Table 21 - Mapping table for zone_composite_constituents_and_their_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
PERCENTAGE_PLY	percentage_ply_definition	209	45	
percentage_ply TO (OPTIONAL) zone_structural_makeup (as makeup_and_properties)	PATH			percentage_ply_definition <= product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship.name = \ 'makeup and properties'} {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition product_definition
percentage_ply TO stock_material FROM material UoF (as material)	PATH		45	percentage_ply_definition <= product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => make_from_usage_option} product_definition_relationship.related_product_definition product_definition
percentage_ply TO (OPTIONAL) ply_orientation_angle (as strength_orientation)	PATH			percentage_ply_definition <= product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation.used_representation -> representation <- representation_relationship.rep_1 representation_relationship => representation_relationship_with_transformation representation_relationship_with_transformation.\ transformation_operator -> transformation transformation = item_defined_transformation item_defined_transformation => transformation_with_derived_angle

Table 21 - Mapping table for zone_composite_constituents_and_their_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
percentage_ply TO percentage (as volume_percent)	PATH		19	percentage_ply_definition <= product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized _definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => measure_representation_item <= measure_with_unit {measure_with_unit [measure_with_unit.value_component -> measure_value measure_value = positive_ratio_measure] [measure_with_unit.unit_component -> unit unit = named_unit named_unit => ratio_unit]} measure_with_unit => ratio_measure_with_unit
PLY_ORIENTATION_ANGLE	transformation_with_derived_angle	209		
ply_orientation_angle to curve (as angle_reference)	PATH			transformation_with_derived_angle <= item_defined_transformation item_defined_transformation.transform_item_2 -> representation_item => geometric_representation_item => curve
ply_orientation_angle to direction (as angle_reference)	PATH			transformation_with_derived_angle <= item_defined_transformation item_defined_transformation.transform_item_2 -> representation_item => geometric_representation_item => direction

Table 21 - Mapping table for zone_composite_constituents_and_their_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
ply_orientation_angle to point_path (as angle_reference)	PATH			transformation_with_derived_angle <= item_defined_transformation item_defined_transformation.transform_item_2 -> representation_item => geometric_representation_item => direction {direction <= geometric_representation_item <= representation_item <- representation.items[i] representation {representation => shape_representation => point_and_vector} representation <- representation_relationship.rep_2 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_1 -> representation => shape_representation => point_path}
ply_orientation_angle to reinforcement_orientation_basis (as basis)	PATH			transformation_with_derived_angle <= item_defined_transformation item_defined_transformation.transform_item_1 -> representation_item => geometric_representation_item => placement => axis2_placement_3d
POINT	point	42		
POINT_AND_VECTOR	point_and_vector	209		
point_and_vector TO point (as location)	PATH			point_and_vector <= shape_representation <= representation representation.items[1] -> representation_item => geometric_representation_item => point

Table 21 - Mapping table for zone_composite_constituents_and_their_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
point_and_vector TO (EXACTLY TWO ORDERED) direction (as vector) #1: first (major) direction #2: second (minor) direction	PATH			point_and_vector <= shape_representation <= representation #1 [representation.items[2] -> representation_item => geometric_representation_item => direction] #2 [representation.items[3] -> representation_item => geometric_representation_item => direction]
POINT_PATH	point_path	209		
point_path TO (ONE OR MANY ORDERED) point_and_vector (as directions)	PATH			point_path <= shape_representation <= representation <- representation_relationship.rep_1 representation_relationship {representation_relationship => shape_representation_relationship} representation_relationship.rep_2 -> representation => shape_representation => point_and_vector
POINT_ZONE_SHAPE point_zone_shape <= zone_structural_makeup_shape_representation	shape_representation	41		
point_zone_shape TO point (as location)	PATH			shape_representation <= representation representation.items[i] -> representation_item => geometric_representation_item => point
REINFORCEMENT_ORIENTATION_BASIS	axis2_placement_3d	42		
reinforcement_orientation_basis TO location (as basis_location)	PATH			axis2_placement_3d <= placement placement.location -> cartesian_point

Table 21 - Mapping table for zone_composite_constituents_and_their_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
reinforcement_orientation_basis TO direction (as orientation)	PATH			axis2_placement_3d axis2_placement_3d.axis -> direction
Application element	**AIM element**	**Source**	**Rules**	**Reference Path**
SMEARED_MATERIAL smeared_material <= zone_structural_makeup	smeared_material_definition	209		smeared_material_definition <= product_definition
smeared_material to length_measure (as total_thickness)	PATH		19	smeared_material_definition <= product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_product_definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => measure_representation_item <= measure_with_unit {measure_with_unit measure_with_unit.value_component -> measure_value measure_value = positive_length_measure} {measure_with_unit measure_with_unit.unit_component -> unit unit = named_unit named_unit => length_unit} measure_with_unit => length_measure_with_unit
SURFACE	surface	42		
SURFACE_WITH_DIRECTION	shape_representation	41		
surface_with_direction TO surface (as defining_surface)	PATH			shape_representation <= representation representation.items[1] -> representation_item => geometric_representation_item => surface

Table 21 - Mapping table for zone_composite_constituents_and_their_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
surface_with_direction TO direction (as material_direction)	PATH			shape_representation <= representation representation.items[2] -> representation_item => geometric_representation_item => direction
THICKNESS_LAMINATE_TABLE thickness_laminate_table <= zone_structural_makeup	thickness_laminate_definition	209	48	thickness_laminate_definition <= product_definition
thickness_laminate_table TO (TWO OR MANY ORDERED) filament_laminate FROM part_composite_constituents UoF (as table_components)	PATH		48	thickness_laminate_definition <= product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition -> product_definition
thickness_laminate_table TO (TWO OR MANY ORDERED) ply FROM part_composite_constituents UoF (as table_components)	PATH		48	thickness_laminate_definition <= product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition -> product_definition
thickness_laminate_table TO (TWO OR MANY ORDERED) processed_core FROM part_composite_constituents UoF (as table_components)	PATH		48	thickness_laminate_definition <= product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => next_assembly_usage_occurrence} product_definition_relationship.related_product_definition -> product_definition

Table 21 - Mapping table for zone_composite_constituents_and_their_representation UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
ZONE_STRUCTURAL_MAKEUP zone_structural_makeup <= laminate_table	product_definition	41		product_definition
zone_structural_makeup TO (OPTIONAL) zone_structural_makeup_shape_representation (as boundary)	PATH			product_definition characterized_product_definition = product_definition characterized_definition = characterized_product_definition characterized_product_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect shape_definition = shapeaspect characterized_definition = shape_definition characterized_product_definition <- property_definition.definition property_definition <-property_definition_representation.definition property_definition_representation {property_definition_representation => shape_definition_representation} property_definition_representation.used_representation -> representation => shape_representation
ZONE_STRUCTURAL_MAKEUP_SHAPE_- REPRESENTATION	shape_representation	41		

The following rules are referenced in the preceding tables:

1. action_directive_requires_date_time
2. action_request_requires_date_time
3. action_request_requires_person_organization
4. approval_requires_approval_date_time
5. approval_requires_approval_person_organization
6. approvals_are_assigned
7. as_required_quantity
8. component_class_for_assembly_select
9. configuration_item_requires_person_organization
10. coordinated_assembly_and_shape
11. dependent_instantiable_approval_status
12. dependent_instantiable_certification_type
13. dependent_instantiable_contract_type
14. dependent_instantiable_date
15. dependent_instantiable_date_and_time
16. dependent_instantiable_date_role
17. dependent_instantiable_date_time_role
18. dependent_instantiable_document_type
19. dependent_instantiable_named_unit
20. dependent_instantiable_organization_role
21. dependent_instantiable_parametric_representation_context
22. dependent_instantiable_person_and_organization_role
23. dependent_instantiable_representation_item
24. dependent_instantiable_security_classification_level
25. dependent_instantiable_shape_representation
26. global_unit_assignment
27. pdu_requires_security_classification
28. ply_reference
29. ply_stock_material_select
30. product_concept_requires_configuration_item
31. product_definition_requires_date_time
32. product_definition_requires_person_organization
33. product_requires_person_organization
34. product_requires_product_category
35. product_requires_version
36. product_version_requires_person_organization
37. product_version_requires_security_classification
38. restrict_action_request_status
39. restrict_approval_status
40. restrict_date_time_role
41. restrict_person_organization_role
42. restrict_product_category_value
43. restrict_product_definition_context
44. restrict_security_classification_level
45. stock_material_reference
46. subtype_mandatory_action

47. subtype_mandatory_shape_representation
48. thickness_laminate_table_component_select
49. unique_version_change_order_rule
50. versioned_action_request_requires_status

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 textual material for constructs that are imported from the integrated resources and the AICs. 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 such items and subtypes apply exclusively to those items that are imported into the AIM.

EXPRESS specification:

```
*)

SCHEMA structural_analysis_design;

  USE FROM aic_advanced_brep;

  USE FROM aic_edge_based_wireframe;

  USE FROM aic_faceted_brep;

  USE FROM aic_geometrically_bounded_surface;

  USE FROM aic_geometrically_bounded_wireframe;

  USE FROM aic_manifold_surface;

  USE FROM aic_shell_based_wireframe;

  USE FROM action_schema
      (action,
       action_directive,
       action_method,
       action_request_solution,
       action_request_status,
       action_status,
       directed_action,
       versioned_action_request);


  USE FROM application_context_schema
      (application_context_element,
       application_protocol_definition,
       product_context,
       product_definition_context);
```



```
USE FROM approval_schema
    (approval,
     approval_date_time,
     approval_person_organization,
     approval_relationship,
     approval_role,
     approval_status);

USE FROM certification_schema
    (certification,
     certification_type);

USE FROM configuration_management_schema
    (configuration_effectivity,
     configuration_item);

USE FROM contract_schema
    (contract,
     contract_type);

USE FROM date_time_schema
    (calendar_date,
     date,
     date_and_time,
     date_role,
     date_time_role,
     ordinal_date,
     week_of_year_and_day_date);

USE FROM document_schema
    (document,
     document_relationship,
     document_type,
     document_usage_constraint,
     document_with_class);

USE FROM effectivity_schema
    (dated_effectivity,
     lot_effectivity,
     serial_numbered_effectivity);

USE FROM fea_scalar_vector_tensor_schema;

USE FROM finite_element_analysis_control_and_result_schema;

USE FROM geometric_model_schema
    (geometric_set);

USE FROM geometry_schema
    (cartesian_transformation_operator_2d,
     curve,
     dummy_gri,
     geometric_representation_context,
     geometric_representation_item,
     offset_curve_2d,
     placement,
     point);
```

```
USE FROM ISO13584_expressions_schema
    (comparison_equal,
     div_expression,
     minus_expression,
     mult_expression,
     plus_expression,
     real_literal,
     real_numeric_variable);


USE FROM ISO13584_generic_expressions_schema
    (environment,
     variable_semantics);


USE FROM management_resources_schema
    (action_assignment,
     action_request_assignment,
     approval_assignment,
     certification_assignment,
     contract_assignment,
     date_assignment,
     date_and_time_assignment,
     document_reference,
     name_assignment,
     organization_assignment,
     person_and_organization_assignment,
     security_classification_assignment);

USE FROM material_property_definition_schema
    (material_designation,
     material_designation_characterization,
     material_property,
     product_material_composition_relationship);

USE FROM material_property_representation_schema
    (data_environment,
     data_environment_relationship,
     material_property_representation);

USE FROM measure_schema
    (amount_of_substance_measure,
     amount_of_substance_measure_with_unit,
     amount_of_substance_unit,
     area_measure,
     area_measure_with_unit,
     area_unit,
     context_dependent_measure,
     context_dependent_unit,
     conversion_based_unit,
     count_measure,
     derived_unit,
     derived_unit_element,
     descriptive_measure,
     dimensional_exponents,
     electric_current_measure,
     global_unit_assigned_context,
     length_measure,
     length_measure_with_unit,
     length_unit,
     luminous_intensity_measure,
```



```
     mass_measure,
     mass_measure_with_unit,
     mass_unit,
     measure_value,
     measure_with_unit,
     named_unit,
     numeric_measure,
     parameter_value,
     plane_angle_measure,
     plane_angle_measure_with_unit,
     plane_angle_unit,
     positive_length_measure,
     positive_plane_angle_measure,
     positive_ratio_measure,
     ratio_measure,
     ratio_unit,
     si_prefix,
     si_unit,
     si_unit_name,
     solid_angle_measure,
     solid_angle_measure_with_unit,
     solid_angle_unit,
     thermodynamic_temperature_measure,
     thermodynamic_temperature_measure_with_unit,
     thermodynamic_temperature_unit,
     time_measure,
     time_measure_with_unit,
     time_unit,
     unit,
     volume_measure,
     volume_measure_with_unit,
     volume_unit);

USE FROM person_organization_schema
    (address,
     organization,
     organization_relationship,
     organization_role,
     organizational_address,
     organizational_project,
     person_and_organization,
     person_and_organization_role,
     personal_address);

USE FROM product_concept_schema
    (product_concept);

USE FROM product_definition_schema
    (product,
     product_category_relationship,
     product_definition,
     product_definition_effectivity,
     product_definition_formation,
     product_definition_formation_relationship,
     product_definition_formation_with_specified_source,
     product_definition_relationship,
     product_definition_with_associated_documents,
     product_related_product_category);
```

```
USE FROM product_property_definition_schema
    (characterized_definition,
     characterized_object,
     property_definition,
     product_definition_shape,
     shape_aspect,
     shape_aspect_relationship);

USE FROM product_property_representation_schema
    (context_dependent_shape_representation,
     property_definition_representation,
     shape_definition_representation,
     shape_representation,
     shape_representation_relationship);

USE FROM product_structure_schema
    (alternate_product_relationship,
     assembly_component_usage,
     assembly_component_usage_substitute,
     make_from_usage_option,
     next_assembly_usage_occurrence,
     product_definition_usage,
     promissory_usage_occurrence,
     quantified_assembly_component_usage,
     specified_higher_usage_occurrence);

USE FROM qualified_measure_schema
     (descriptive_representation_item,
      expanded_uncertainty,
      measure_qualification,
      measure_representation_item,
      precision_qualifier,
      qualified_representation_item,
      qualitative_uncertainty,
      standard_uncertainty,
      type_qualifier,
      uncertainty_qualifier);

USE FROM representation_schema
    (definitional_representation,
     functionally_defined_transformation,
     global_uncertainty_assigned_context,
     item_defined_transformation,
     mapped_item,
     parametric_representation_context,
     representation_context,
     representation_item,
     representation_map,
     representation_relationship,
     representation_relationship_with_transformation);

USE FROM security_classification_schema
    (security_classification,
     security_classification_level);

USE FROM shape_dimension_schema
    (dimensional_size,
     shape_dimension_representation);

USE FROM structural_response_definition_schema;
```

```
USE FROM structural_response_representation_schema;

USE FROM topology_schema
   (dummy_tri);
(*
```

NOTE The schemas referenced above can be found in the following parts of ISO 10303:

action_schema	ISO 10303-41
aic_advanced_brep	ISO 10303-514
aic_edge_based_wireframe	ISO 10303-501
aic_faceted_brep	ISO 10303-512
aic_geometrically_bounded_surface	ISO 10303-507
aic_geometrically_bounded_wireframe	ISO 10303-510
aic_manifold_surface	ISO 10303-509
aic_shell_based_wireframe	ISO 10303-502
application_context_schema	ISO 10303-41
approval_schema	ISO 10303-41
certification_schema	ISO 10303-41
configuration_management_schema	ISO 10303-44
contract_schema	ISO 10303-41
date_time_schema	ISO 10303-41
document_schema	ISO 10303-41
effectivity_schema	ISO 10303-41
fea_scalar_vector_tensor_schema	ISO 10303-104
finite_element_analysis_control_and_result_schema	ISO 10303-104
geometric_model_schema	ISO 10303-42
geometry_schema	ISO 10303-42
ISO13584_expressions_schema	ISO 13584-20
ISO13584_generic_expressions_schema	ISO 13584-20
management_resources_schema	ISO 10303-41
material_property_definition_schema	ISO 10303-45
material_property_representation_schema	ISO 10303-45
measure_schema	ISO 10303-41
person_organization_schema	ISO 10303-41
product_concept_schema	ISO 10303-44
product_definition_schema	ISO 10303-41
product_property_definition_schema	ISO 10303-41
product_property_representation_schema	ISO 10303-41
product_structure_schema	ISO 10303-44
qualified_measure_schema	ISO 10303-45
representation_schema	ISO 10303-43
security_classification_schema	ISO 10303-41
shape_dimension_schema	ISO 10303-47
structural_response_definition_schema	ISO 10303-104
structural_response_representation_schema	ISO 10303-104
topology_schema	ISO 10303-42

5.2.1 Fundamental concepts and assumptions

ISO 10303-209 is designed to be used in the design and finite element analysis of three dimensional metallic and composite product information. A fundamental concept of this schema is that the design organization controls the configuration of the design discipline product information, and the analysis organization controls the configuration of the analysis discipline product information. The analysis shape representations are based in whole on the design shape, upon an idealization of the design shape, solely upon a shape of three dimensional points, or a combination of all. Composite constituent information (such as ply boundaries and laminate tables) is associated with the analysis- or design-discipline product definitions, or both.

5.2.1.1 Relating the shape and analysis of a product to its configuration data

The shape of a product in this part of ISO 10303 is represented by the **shape_representation** entity. This entity and its subtypes define the geometric and/or topological entities that make up a particular representation type. Each mechanical part or assembly that is of interest will be given by an instance of the **product** entity. Each product will have at least one version given by an instance of the **product_-definition_formation** entity. Each version may have one or more definitions given by the **product_-definition** entity. The **product_definition_context.life_cycle_stage** for the design discipline product definition is specified as 'design'. Each design discipline product definition may have its design shape represented by relating the **product_definition** instance to the appropriate **shape_representation** instance. This relationship is given by an instance of the entity **shape_definition_representation**. A **shape_-definition_representation** inherits attributes from its supertype **property_definition_representation**. In this context, the **used_representation** attribute of **property_definition_representation** references the **shape_representation** entity which contains the geometry and/or topology for the shape of the part. At the same time, a second attribute (**definition)** of **property_definition_representation** references the **property_definition** entity which in turn points to a select type called **characterized_definition**. This type allows for the representation of a **shape_aspect**, **shape_aspect_relationship,** or a **characterized_product_-definition** that is also a select type.

To specify that the property of interest is the shape of the **product_definition**, the **product_definition_-shape** subtype of **property_definition** shall be used. This subtype is constrained to use the **characterized_-product_definition** select type. The **characterized_product_definition** allows for the representation of a **product_definition** or a **product_definition_relationship** via a reference through its select list. In order to specify the shape of a product, the product definition shall be referenced here. This reference ensures that the **product_definition_shape** entity is used to define the shape of a **product_definition**.

If the property of interest is an aspect of the shape of the **product_definition** or a relationship between two aspects of the shape of a **product_definition**, the **property_definition** shall be used and its **definition** attribute shall reference an instance of a **shape_aspect** or a **shape_aspect_relationship**.

The constraint **subtype_mandatory_shape_representation** specifies the subtypes of **shape_-representation** that shall be used to specify the shape of a part. The representation of a **shape_aspect** or a **shape_aspect_relationship** may be given by a any set of **representation_item**s.

Similarly the analysis of products in this part of ISO 10303 is represented by a finite element model entity (**fea_model**) and associated entities. The **fea_model** is associated with a **product_definition_context.life_-cycle_stage** that is specified to be 'analysis'. Idealized and node shape representations are related to the **product_definition** as described above. There is a separate versioning of the analysis discipline **product_-definition**s.

5.2.1.2 Relating the design shape of a component to the design shape of its assembly

There are two methods used to relate a component part's design shape to the design shape of the assembly part in which it is assembled. The first method (Method 1) consists of defining the shape for each part (component and assembly), and then relating the two shapes and providing the information that defines the orientation of the component part with respect to the assembly part through a transformation. The second method (Method 2) consists of defining the shape for each part (component and assembly), and then incorporating the shape of the component directly in the shape of the assembly. The first method shall be used to relate the shapes that are represented by different representation types. The second method may be used for the incorporation of a component's representation into the assembly's representation if the two representation types are the same.

Both methods employ the **shape_representation** and **product_definition** entities. The first method also employs the **product_definition_relationship**, **shape_representation_relationship** AND **representation_relationship_with_transformation** (an instance of each entity forming a complex entity instance of the two entities with an AND relationship), and **context_dependent_shape_representation** entities. The second method employs the **mapped_item** and **representation_map** entities.

When using the first method to relate the shape of the component to the shape of the assembly, each of the **shape_representation** entities that define the shapes of the component and assembly **product_definition**s is related through references in the **shape_representation_relationship** entity. Orientation of the component in the assembly is provided by the formation of a complex instance of the **shape_-representation_relationship** AND **representation_relationship_with_transformation** entities. The **representation_relationship_with_transformation** entity references a **transformation** that is a select type allowing the orientation to be defined using a **placement** entity in each representation for an **item_defined_-transformation** or a **cartesian_transformation_operator** entity for a **functionally_defined_-transformation**. In addition, an instance of the **context_dependent_shape_representation** shall be given to explicitly relate the **shape_representation_relationship** that defines the relationship of the two shapes to the **product_definition_relationship** that defines the assembly-component relationship between the two **product_definitions**.

When using the second method to relate the shape of the component to the shape of the assembly, the **shape_representation** entity that defines the geometry and/or topology for the component part's shape is referenced by an instance of a **representation_map** entity that is referenced by the **mapping_source** attribute of an instance of the **mapped_item** entity. The attribute **mapped_representation** of the **representation_map** will reference the **shape_representation** subtype that defines the geometric and/or topological representation of the component's shape. The instance of the **mapped_item** entity is then added to the set of **items** in the **shape_representation** entity that defines the geometry and/or topology for the assembly part.

5.2.1.3 Relating a finite element analysis model to the shape of a product

The shape information for a finite element model is specified by the location of nodes. The nodes are connected by finite elements that represent the solid continuum of the part. The node locations in turn are specified by at least one of the following: a collection of points in space (**point_model**); the design shape (**nominal_design_shape**), and an idealization of design shape (**idealized_analysis_shape**) that is related to a design shape. The **idealized_analysis_shape** can be related to a conceptual design shape for which a specific geometric representation is not required. The design shape may be a component or an assembly of components, depending on the analysis viewpoint.

Attributes of a finite element model are related to geometry indirectly through the relationship between the nodes and shape information.

EXAMPLE The load upon a surface of a part can be derived by summing the load information of the associated nodes and the elements associated with those nodes.

5.2.1.4 Types of shape representation

This part of ISO 10303 defines six types of representation for shapes of parts, one type of representation for composite constituents, and one for finite element nodes. Part shape representations include wireframe representations with topology, wireframe and surface representations that are geometrically bounded (no topology), manifold surface models, faceted boundary representation solid models, and advanced boundary representation solid models. The composite constituent shape representations include a 3D geometry set for an unstructured composite part representation, and a topological and geometric sheet representation for plies. A shape representation of points only is included to allow a finite element model to be exchanged without any associated part shape information. Each of these types of representation is self-contained meaning that one type may not contain another type. Each of the types is given by a subtype of the **shape_-representation** entity. Each subtype contains local rules that govern the types of geometric and/or topological entities that can be used in it. Every **shape_representation** shall be one of the subtypes unless it is used as the representation of the shape of an assembly. In that case, the **shape_representation** instance will contain only **placement** entities in its set of **items** if Method 1 is used define the orientation of the components' representations in it, or **mapped_item** entities if Method 2 is used to incorporate the component **shape_representation**s into the assembly **shape_representation** (see 5.2.1.2). Any **shape_-representation** that is referenced by a **representation_map** to implement Method 2 shall be of the same type as the **shape_representation** that has in its set of **items** the **mapped_item** which references this **representation_map**.

5.2.1.5 Finite element analysis model, and associated controls and results

Many types of analyses can be conducted to ensure the performance and integrity of a product. Different aspects of a product are idealized and then analyzed as a continuum. Exact mathematical models for any but the simplest continuum shapes are intractable. Therefore analytical methods that represent the continuum as discrete tractable shapes are used. There are many discrete analytical methodologies, such as the finite element, finite difference, and boundary element methods. This part of ISO 10303 addresses the finite element analysis only.

In performing an analysis with finite element methods, the continuum of a product is discretized into a finite element model that consists of a mesh of points (nodes) in the continuum that are connected with elements. The elements represent finite portions of the continuum that when connected with shared nodes collectively respond as would the entire continuum. The elements have associated physical and material properties. There are also coordinate systems, groups, and administrative information associated with the finite element model. Load, constraint, analysis selection and output control information are combined with the finite element model to form a complete input to finite element analysis. Once the analysis is performed, the results may be output at the nodes and at non-nodal positions within an element. There are other possible output information such as total strain energy that are not associated with a position within the finite element model.

The entities of ISO 10303-104 are incorporated as a whole in this part of ISO 10303. The reader should consult ISO 10303-104 for further details concerning the finite element analysis.

5.2.1.6 Composite and metallic materials

The material for a design or analysis discipline product definition may be a simple homogeneous (isotropic) metallic material, or a more complex composite laminate material description. Both design and analysis product definitions, and finite element material response matrices may have associated material specification definitions and property representations.

5.2.1.7 Composite laminate constituent definition and representation

A composite part is defined by the constituents that are laminated in layers to create the part. This internal makeup is defined for the total part by a **composite_assembly_definition** or a **ply_laminate_definition**. Alternately the internal make-up can be defined by one or many **thickness_laminate_definition** that applies to a point or a zone **shape_aspect** of the part. Also this internal make-up for a **shape_aspect** can be idealized by using this t**hickness_laminate_definition** or the **percentage_ply_definition**.

A **composite_assembly_definition** differs from a **ply_laminate_definition** in that it can contain constituents other than plies and may include other assemblies. A **composite_assembly_definition** is made up of layers or sequences defined by **composite_assembly_sequence_definition**s. The **composite_-assembly_definition** is related to the first **composite_assembly_sequence_definition** with a **next_assembly_usage_occurrence** entity. In this **next_assembly_usage_occurrence**, the **next_assembly_usage_occurrence.relating_product_definition** identifies the **composite_assembly_-definition** and the **next_assembly_usage_occurrence.related_product_definition** identifies the first **composite_assembly_sequence_definition**. The **next_assembly_usage_occurrence.relating_product_-definition** of the next **next_assembly_usage_occurrence** identifies the first layer or sequence defined by a **composite_laminate_sequence_definition**. Subsequent **composite_assembly_sequence_definition**s are ordered using similar **next_assembly_usage_occurrence**s. The constituent parts are related to the appropriate layer or sequence (**composite_laminate_sequence_definition**) using a **next_assembly_usage_occurrence** for each constituent part in the layer or sequence. The **next_assembly_usage_occurrence.relating_product_definition** identifies the particular **composite_-assembly_sequence_definition** and the **next_assembly_usage_occurrence.related_product_definition** identifies the **product_definition** for the constituent part.

A **ply_laminate_definition** is made up solely of layers of plies. Each layer or sequence is defined by a **ply_-laminate_sequence_definition**. The **ply_laminate_definition** is related to the first layer or sequence with a **next_assembly_usage_occurrence**. In this **next_assembly_usage_occurrence**, the **next_assembly_usage_occurrence.relating_product_definition** identifies the **ply_laminate_definition** and the **next_assembly_usage_occurrence.related_product_definition** identifies the first **ply_laminate_-sequence_definition**. The **next_assembly_usage_occurrence.relating_product_definition** of the next

next_assembly_usage_occurrence identifies the first layer or sequence defined by a **ply_laminate_-sequence_definition**. Subsequent **ply_laminate_sequence_definition**s are ordered using similar **next_assembly_usage_occurrence**s. The plies are related to the appropriate layer or sequence (**ply_-laminate_sequence_definition**) using a **next_assembly_usage_occurrence** for each ply in the layer or sequence. The **next_assembly_usage_occurrence.relating_product_definition** identifies the particular **ply_laminate_sequence_definition** and the **next_assembly_usage_occurrence.related_product_-definition** identifies the **product_definition** for the ply.

A **thickness_laminate_definition** is similar to an **composite_assembly_definition**, but it applies to a shape aspect. Since each layer or sequence is local, the definition is a single constituent part. The **next_assembly_usage_occurrence.related_product_definition** identifies the **thickness_laminate_-definition** while the **relating_product_definition** is the first product in the sequence. Subsequent products are defined in order using **next_assembly_usage_occurrence**.

A **percentage_laminate_definition** is an idealization that the defines the cross-section by percentages of material. The **percentage_laminate_definition** is the **product_definition** of the section. The parts making up a percentage of this are defined by a **percentage_ply_definition**. The **next_assembly_usage_occurrence.relating_product_definition** identifies the **percentage_laminate_-definition**. All **percentage_ply_definition**s are related to this **percentage_laminate_definition**. Each **percentage_ply_definition** has a **measure_value** and **product** defined to as their percentage and material.

5.2.1.8 Use of global rules

Many of the relationships among different entities in the integrated resource parts of ISO 10303 are specified using the most generic cardinality of zero or more between two related entities. This cardinality means that the relationship is optional or there may be one or more instances of a related entity that is related to a single instance of the relating entity. This part of ISO 10303 uses global rules to constrain that cardinality. In some cases the constraint is made to be one to one, and in some instances at least one. Examples of these rules include **approval_requires_approval_date_time** for exactly one, and **change_-request_requires_person_organization** for one or more.

Global rules are also used to restrict the values of STRING type attributes to be only those that are applicable within the context of the design and analysis of mechanical parts and assemblies. Examples of these rules are **restrict_approval_status** and **restrict_person_organization_role**.

5.2.1.9 Assignment of units

Units are assigned to the representation of shape on a global basis. This is done by the creation of an instance of the **global_unit_assigned_context**. This entity contains an attribute that allows for a set of units to be assigned to a **representation_context**. Each of the **shape_representation** entities has a context for its representation. If units of measure are desired for a particular instance of **shape_representation**, then that instance shall have a **global_unit_assigned_context** in its **context_of_items** attribute.

Units are also assigned to a finite element model on a global basis, utilizing the same approach as for shape representations. The **global_unit_assigned_context** for the **fea_model** are specified by the **context_of_-items** attribute inherited from the **representation** supertype. The units apply to all aspects of a finite element model such as geometric and material properties, and all scalars, vectors, and tensors.

5.2.2 AP209 type definitions

5.2.2.1 action_item

An **action_item** specifies those objects to which an **action** may be assigned.

EXPRESS specification:

```
*)
TYPE action_item = SELECT
  (product_definition,
   product_definition_formation,
   product_definition_relationship);
END_TYPE;
(*
```

5.2.2.2 action_request_item

An **action_request_item** specifies those objects to which a **versioned_action_request** may be assigned.

EXPRESS specification:

```
*)
TYPE action_request_item = SELECT
  (product_definition,
   product_definition_formation,
   product_definition_relationship);
END_TYPE;
(*
```

5.2.2.3 approval_item

An **approval_item** specifies those objects to which an **approval** may be assigned.

EXPRESS specification:

```
*)
TYPE approval_item = SELECT
  (action_directive,
   certification,
   configuration_item,
   configuration_effectivity,
   contract,
   control,
   fea_model,
   product_definition,
   product_definition_formation,
   result,
   retention,
   security_classification,
   versioned_action_request);
END_TYPE;
(*
```

5.2.2.4 certification_item

A **certification_item** specifies those objects to which a **certification** may be assigned.

EXPRESS specification:

```
*)
TYPE certification_item = SELECT
  (product_definition_relationship);
END_TYPE;
(*
```

5.2.2.5 contract_item

A **contract_item** specifies those objects to which a **contract** may be assigned.

EXPRESS specification:

```
*)
TYPE contract_item = SELECT
  (product_definition_formation);
END_TYPE;
(*
```

5.2.2.6 date_item

A **date_item** specifies those objects to which a **date** may be assigned. The role for the **date** is controlled by the **applied_date_correlation** function.

EXPRESS specification:

```
*)
TYPE date_item = SELECT
  (action_directive,
   approval_person_organization,
   certification,
   contract,
   product_definition,
   retention,
   security_classification,
   versioned_action_request);
END_TYPE;
(*
```

5.2.2.7 date_time_item

A **date_time_item** specifies those objects to which a **date_and_time** may be assigned. The role for the **date_and_time** is controlled by the **applied_date_time_correlation** function.

EXPRESS specification:

```
*)
TYPE date_time_item = SELECT
  (action_directive,
   approval_person_organization,
   certification,
   contract,
```



```
   product_definition,
   retention,
   security_classification,
   versioned_action_request);
END_TYPE;
(*
```

5.2.2.8 document_reference_item

A **document_reference_item** specifies those objects to which a **document** may be assigned.

EXPRESS specification:

```
*)
TYPE document_reference_item = SELECT
  (product_definition,
   shape_aspect);
END_TYPE;
(*
```

5.2.2.9 name_item

A **name_item** specifies those objects to which a label may be assigned.

EXPRESS specification:

```
*)
TYPE name_item = SELECT
  (expression_conversion_based_unit);
END_TYPE;
(*
```

5.2.2.10 organization_item

A **organization_item** specifies those objects to which a **organization** may be assigned. The role for the **organization** is controlled by the **applied_organization_correlation** function.

EXPRESS specification:

```
*)
TYPE organization_item = SELECT
  (action_directive,
   configuration_item,
   contract,
   product,
   product_definition,
   product_definition_formation,
   security_classification,
   versioned_action_request);
END_TYPE;
(*
```

5.2.2.11 person_organization_item

A **person_organization_item** specifies those objects to which a **person_organization** may be assigned. The role for the **person_and_organization** is controlled by the **applied_person_and_organization_-correlation** function.

EXPRESS specification:

```
*)
TYPE person_organization_item = SELECT
  (action_directive,
   configuration_item,
   contract,
   product,
   product_definition,
   product_definition_formation,
   security_classification,
   versioned_action_request);
END_TYPE;
(*
```

5.2.2.12 retention_item

A **retention_item** specifies those objects to which a **retention** may be assigned.

EXPRESS specification:

```
*)
TYPE retention_item = SELECT
  (control,
   document,
   material_property,
   product,
   product_definition,
   product_definition_formation,
   product_definition_relationship,
   representation,
   result);
END_TYPE;
(*
```

5.2.2.13 security_classification_item

A **security_classification_item** specifies those objects to which a **security_classification** may be assigned.

EXPRESS specification:

```
*)
TYPE security_classification_item = SELECT
  (product_definition,
   product_definition_formation,
   product_definition_usage);
END_TYPE;
(*
```

5.2.3 AP209 entity definitions

5.2.3.1 applied_action_assignment

An **applied_action_assignment** relates an **action** to a particular piece of product data.

EXPRESS specification:

```
*)
ENTITY applied_action_assignment
  SUBTYPE OF (action_assignment);
  items : SET [1:?] OF action_item;
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **action_item**s which identifies the product data to which an **action** is assigned.

5.2.3.2 applied_action_request_assignment

An **applied_action_request_assignment** relates a **versioned_action_request** to a particular piece of product data.

EXPRESS specification:

```
*)
ENTITY applied_action_request_assignment
  SUBTYPE OF (action_request_assignment);
  items : SET [1:?] OF action_request_item;
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **action_request_item**s which identifies the product data to which a **versioned_action_-request** is assigned.

5.2.3.3 applied_approval_assignment

An **applied_approval_assignment** relates an **approval** to a particular piece of product data.

EXPRESS specification:

```
*)
ENTITY applied_approval_assignment
  SUBTYPE OF (approval_assignment);
  items : SET [1:?] OF approval_item;
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **approval_item**s which identifies the product data to be approved.

5.2.3.4 applied_certification_assignment

An **applied_certification_assignment** assigns a **certification** to a part that is supplied by an external organization.

EXPRESS specification:

```
*)
ENTITY applied_certification_assignment
  SUBTYPE OF (certification_assignment);
  items : SET [1:?] OF certification_item;
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **certification_item**s which identifies the **supplied_part_relationship** entities to which the **certification** is assigned.

5.2.3.5 applied_contract_assignment

An **applied_contract_assignment** relates a **contract** to a **product_definition_formation**.

EXPRESS specification:

```
*)
ENTITY applied_contract_assignment
  SUBTYPE OF (contract_assignment);
  items : SET [1:?] OF contract_item;
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **contract_item**s which identifies the **product** versions to which the contract is assigned.

5.2.3.6 applied_date_and_time_assignment

An **applied_date_and_time_assignment** assigns a **date_and_time** to a particular piece of product data.

EXPRESS specification:

```
*)
ENTITY applied_date_and_time_assignment
  SUBTYPE OF (date_and_time_assignment);
  items : SET [1:?] OF date_time_item;
```



```
WHERE
  WR1: applied_date_time_correlation (SELF);
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **date_time_item**s which identifies the product data to which a **date_and_time** is assigned.

Formal propositions:

WR1: The **applied_date_time_correlation** function that correlates roles of dates and times to elements of product data shall be satisfied.

5.2.3.7 applied_date_assignment

An **applied_date_assignment** assigns a **date** to a particular piece of product data.

EXPRESS specification:

```
*)
ENTITY applied_date_assignment
  SUBTYPE OF (date_assignment);
  items : SET [1:?] OF date_item;
WHERE
  WR1: applied_date_correlation (SELF);
END_ENTITY;
(*
```

Formal propositions:

WR1: The **applied_date_correlation** function that correlates roles of dates to elements of product data shall be satisfied.

Attribute definitions:

items: a set of **date_item**s which identifies the product data to which a **date** is assigned.

5.2.3.8 applied_document_reference

An **applied_document_reference** associates a **document** to a complete definition of a product or to some aspect of its shape.

```
*)
ENTITY applied_document_reference
  SUBTYPE OF (document_reference);
  items : SET [1:?] OF document_reference_item;
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **document_reference**s which identifies the product data to which a **document** is assigned.

5.2.3.9 applied_name_assignment

An **applied_name_assignment** assigns a label to an object.

In the representation of a conversion algebraic expression that is used to derive one unit from another, the name of the derived unit can be represented by using the entity **applied_name_assignment**.

EXAMPLE In the representation of the conversion that relates the unit Degree Fahrenheit to the unit Degree Celsius by the expression:

$$T_f = 1.8*T_c + 32$$

an instance of **expression_conversion_based_unit** is used to represent the unit Degree Fahrenheit. The name of the unit Degree Fahrenheit is represented by associating the instance of **expression_conversion_based_unit** to an instance of **applied_name_assignment** with the label 'Degree Fahrenheit'.

EXPRESS specification:

```
*)
ENTITY applied_name_assignment
  SUBTYPE of (name_assignment);
  items : SET [1:?] OF name_item;
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **name_items** which identifies the objects to which a label is assigned..

5.2.3.10 applied_organization_assignment

An **applied_organization_assignment** relates an **organization** to a particular piece of product data.

EXPRESS specification:

```
*)
ENTITY applied_organization_assignment
  SUBTYPE OF (organization_assignment);
  items : SET [1:?] OF organization_item;
WHERE
  WR1: applied_organization_correlation (SELF);
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **organization_item**s which identifies the product data to which an **organization** is assigned.

Formal proposition

WR1: The **applied_organization_correlation** function that correlates roles of organizations to elements of product data shall be satisfied.

5.2.3.11 applied_person_and_organization_assignment

An **applied_person_and_organization_assignment** relates a **person_and_organization** to a particular piece of product data.

EXPRESS specification:

```
*)
ENTITY applied_person_and_organization_assignment
  SUBTYPE OF (person_and_organization_assignment);
  items : SET [1:?] OF person_organization_item;
WHERE
  WR1: applied_person_and_organization_correlation (SELF);
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **person_and_organization_item**s which identifies the product data to which a **person_and_-organization** is assigned.

Formal proposition

WR1: The **applied_person_and_organization_correlation** function that correlates roles of persons and organizations to elements of product data shall be satisfied.

5.2.3.12 applied_security_classification_assignment

An **applied_security_classification_assignment** relates a **security_classification** to a particular piece of product data.

EXPRESS specification:

```
*)
ENTITY applied_security_classification_assignment
  SUBTYPE OF (security_classification_assignment);
  items : SET [1:?] OF security_classification_item;
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **security_classification_item**s which identifies the product data to which a **security_-classification** is assigned.

5.2.3.13 beveled_sheet_representation

A **beveled_sheet_representation** is a **shape_representation** that has a significant thickness, and whose edge is beveled at a constant angle around its entire perimeter. It is an extension of the concept of a thin sheet to products such as honeycomb core that have significant thickness but are easily formed to mate with a curved surface.

EXPRESS specification:

```
*)
ENTITY beveled_sheet_representation
  SUBTYPE OF (shape_representation);
END_ENTITY;
(*
```

5.2.3.14 composite_assembly_definition

A **composite_assembly_definition** is a **product_definition** containing an ordered list of constituents that make up the composite assembly. The ordered list is defined by use of **next_assembly_usage_occurrence**.

NOTE 1 The valid constituents are ply, ply_laminate, filament_laminate, processed_core, and composite_assembly products.

NOTE 2 This **product_definition** defines composite assemblies.

EXPRESS specification:

```
*)
ENTITY composite_assembly_definition
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN (SELF,
                             'PRODUCT_DEFINITION_SCHEMA.' +
                             'PRODUCT_DEFINITION_RELATIONSHIP.' +
                             'RELATING_PRODUCT_DEFINITION') |
                             'PRODUCT_STRUCTURE_SCHEMA.' +
                             'NEXT_ASSEMBLY_USAGE_OCCURRENCE' IN
                             TYPEOF (pdr))) = 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: The **composite_assembly_definition** shall be used by one **next_assembly_usage_occurrence**.

NOTE The first next_assembly_usage_occurrence.relating_product_definition shall be the composite_assembly_-definition, and the next_assembly_usage_occurrence.related_product_definition shall be the first composite_assembly_-sequence_definition in the table. Thereafter the subsequent next_assembly_usage_occurrence chain together composite_assembly_sequence_definitions entities into an ordered table.

5.2.3.15 composite_assembly_sequence_definition

An **composite_assembly_sequence_definition** is a **product_definition** of a single sequence or layer within a composite_assembly. This sequence or layer contains one or more component_class_for_assembly members.

The **next_assembly_usage_occurrence** establishes the composite constituents of this **product_definition**.

NOTE 1 Products on a layer or sequence shall not overlap. If they overlap they are to be related to separate composite_-assembly_sequence_definition.

NOTE 2 A product could be used at multiple locations in a sequence or layer.

EXPRESS specification:

```
*)
ENTITY composite_assembly_sequence_definition
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN (SELF,
                             'PRODUCT_DEFINITION_SCHEMA.' +
                             'PRODUCT_DEFINITION_RELATIONSHIP.' +
                             'RELATING_PRODUCT_DEFINITION') |
                             'PRODUCT_STRUCTURE_SCHEMA.' +
                             'NEXT_ASSEMBLY_USAGE_OCCURRENCE' IN
                             TYPEOF (pdr))) > 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: An **composite_assembly_sequence_definition** shall be used by one or many **next_assembly_usage_occurrence**.

NOTE The next_assembly_usage_occurrence.relating_product_definition shall be the composite_assembly_-sequence_definition. The next_assembly_usage_occurrence.related_product_definition thus associated comprise the BAG of product_definition entities (that are mappings of component_class_for_assembly ARM objects) for the table.

5.2.3.16 composite_sheet_representation

A **composite_sheet_representation** is a **shape_representation** of a sheet product. A sheet product is a product whose thickness is constant and small in comparison to its length and width. Most sheet products will conform or can be readily formed to the surface which they are placed. This product shape is represented as an actual shape on some surface or is projected to a surface that has a defined role for the part.

A **composite_sheet_representation** shall be either a **geometrically_bounded_surface_shape_-representation** or a **manifold_surface_shape_representation**.

EXPRESS specification:

```
*)
ENTITY composite_sheet_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (['AIC_GEOMETRICALLY_BOUNDED_SURFACE.' +
       'GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION',
       'AIC_MANIFOLD_SURFACE.' +
       'MANIFOLD_SURFACE_SHAPE_REPRESENTATION'] * TYPEOF (SELF)) = 1;
END_ENTITY;
(*
```

Informal propositions:

IP: The thickness shall be consistent with the thickness of the constituent part that it represents.

Formal propositions:

WR1: Any instance of **composite_sheet_representation** shall be either of type **geometrically_bounded_-surface_representation**, or of type **manifold_surface_shape_representation**.

5.2.3.17 design_make_from_relationship

A **design_make_from_relationship** specifies that one **product**'s design has been designed from another **product**'s design. The use of the **design_make_from_relationship** also implies that the physical product resulting from the manufacture of the **relating_product_definition** is to be used as the basis for the manufacture of the **related_product_definition**.

EXPRESS specification:

```
*)
ENTITY design_make_from_relationship
  SUBTYPE OF (product_definition_relationship);
END_ENTITY;
(*
```

Attribute definitions:

SELF/product_definition_relationship.relating_product_definition: the **product_definition** that is the source for the make from relationship.

SELF/product_definition_relationship.relating_product_definition: the **product_definition** that is created from the source **product_definition**.

5.2.3.18 draped_defined_transformation

A draped_defined_transformation is an orientation_angle that defines a position and direction within a part at a point. The warp direction of a fabric shall be placed in the specified direction prior to draping over the part.

NOTE- Once the fabric has been draped, the orientation angle at any given point shall be derived from the draped shape of the fabric.

EXPRESS specification:

```
*)
ENTITY draped_defined_transformation
  SUBTYPE OF (item_defined_transformation);
END_ENTITY;
(*
```

5.2.3.19 expression_conversion_based_unit

An **expression_conversion_based_unit** is a type of **named_unit** and **variable_semantics** that represents a unit which is converted from another unit, using any algebraic expression.

The conversion that is used to derive one unit from another is represented by associating to each unit involved in the conversion expression the concept of variable and describing the algebraic expression that relates these variables. The unit that is derived is represented by the **expression_conversion_based_unit**.

EXAMPLE In the representation of the conversion that relates the unit Degree Fahrenheit to the unit Degree Celsius by the expression:

$$T_f = 1.8*T_c + 32$$

an instance of **expression_conversion_based_unit** is used to represent the unit Degree Fahrenheit.

EXPRESS specification:

```
*)
ENTITY expression_conversion_based_unit
  SUBTYPE of (named_unit, variable_semantics);
INVERSE
  associated_variable_environment : environment FOR semantics;
END_ENTITY;
(*
```

Attribute definitions:

associated_variable_environment: the **environment** that specifies the representation for the derived unit.

5.2.3.20 flat_pattern_ply_representation_relationship

A **flat_pattern_ply_representation_relationship** is the relationship between two **shape_representation**s of a flat pattern. This specialization of **shape_representation_relationship** adds meaning to the two attributes: **rep_1**, **rep_2.**

EXPRESS specification:

```
*)
ENTITY flat_pattern_ply_representation_relationship
  SUBTYPE OF (shape_representation_relationship);
WHERE
  WR1: 'STRUCTURAL_ANALYSIS_DESIGN.SHAPE_REPRESENTATION' IN
       (TYPEOF (SELF\representation_relationship.rep_1) *
        TYPEOF (SELF\representation_relationship.rep_2));
  WR2: SELF\representation_relationship.rep_1.
       context_of_items\geometric_representation_context.
       coordinate_space_dimension = 3;
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation_relationship.rep_1: a **shape_representation** from which the flat pattern was derived.
SELF\representation_relationship.rep_2: the **shape_representation** that is the flat pattern shape.

Formal propositions:

WR1: rep_1 and **rep_2** shall be **shape_representation**s.

WR2: rep_1 shall always be 3D.

5.2.3.21 laid_defined_transformation

A **laid_defined_transformation** is an **orientation_angle** that explicitly defines a direction at a point within a part.

EXPRESS specification:

```
*)
ENTITY laid_defined_transformation
  SUBTYPE OF (item_defined_transformation);
END_ENTITY;
(*
```

5.2.3.22 named_unit_variable

A **named_unit_variable** is a type of **named_unit** and **variable_semantics** that represents the reference unit which is converted to another unit, using any algebraic expression.

The conversion that is used to derive one unit from another is represented by associating to each unit involved in the conversion expression the concept of variable and describing the algebraic expression that relates these variables. The unit that is converted is represented by the **named_unit_variable**.

EXAMPLE In the representation of the conversion that relates the unit Degree Fahrenheit to the unit Degree Celsius by the expression:

$$T_f = 1.8*T_c + 32$$

an instance of **named_unit_variable** is used to represent the unit Degree Celsius.

EXPRESS specification:

```
*)
ENTITY named_unit_variable
  SUBTYPE OF (named_unit, variable semantics);
INVERSE
  associated_variable_environment : environment FOR semantics;
END_ENTITY;
(*
```

Attribute definitions:

associated_variable_environment: the **environment** that specifies the representation for the named unit.

5.2.3.23 percentage_laminate_definition

A **percentage_laminate_definition** is an idealization that says that the cross-section is defined by percentages of material. The parts making up a percentage of this are defined by a **percentage_ply_-definition**.

The **next_assembly_usage_occurrence** establishes the percentage_ply constituents of this **product_-definition**.

EXPRESS specification:

```
*)
ENTITY percentage_laminate_definition
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN (SELF,
                             'PRODUCT_DEFINITION_SCHEMA.' +
                             'PRODUCT_DEFINITION_RELATIONSHIP.' +
                             'RELATING_PRODUCT_DEFINITION') |
                             'PRODUCT_STRUCTURE_SCHEMA.' +
                             'NEXT_ASSEMBLY_USAGE_OCCURRENCE' IN
                             TYPEOF (pdr))) > 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **percentage_laminate_definition** shall be used by one or many **next_assembly_usage_-occurrence**.

NOTE The next_assembly_usage_occurrence.relating_product_definition shall be the percentage_laminate_definition. The next_assembly_usage_occurrence.related_product_definition thus associated comprise the SET of product_-definition entities (that are mappings of percentage_ply ARM objects) for the table.

5.2.3.24 percentage_ply_definition

A **percentage_ply_definition** is a **product_definition** that represents a portion of the overall thickness of a laminate. This portion of the overall thickness, defined by a **measure_representation_item**, is of unique material and orientation.

The makeup and properties of a **percentage_ply_definition** shall not be represented in turn by a **percentage_laminate_definition**.

EXPRESS specification:

```
*)
ENTITY percentage_ply_definition
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN (SELF, 'PRODUCT_DEFINITION_SCHEMA.' +
       'PRODUCT_DEFINITION_RELATIONSHIP.RELATING_PRODUCT_DEFINITION') |
       ('STRUCTURAL_ANALYSIS_DESIGN.PERCENTAGE_LAMINATE_DEFINITION'
       IN TYPEOF (pdr.related_product_definition)) AND
       (pdr.name = 'makeup and properties'))) = 0;
END_ENTITY;
```

```
(*
```

Informal propositions:

IP: Each **percentage_ply_definition** will have an associated **transformation_with_derived_angle**, **measure_representation_item**, and **product_definition_relationship.related_product_definition**.

Formal propositions:

WR1: A **percentage_ply_definition** shall be not be related to a **percentage_laminate_definition** by a **product_definition_relationship** whose **name** attribute is "makeup and properties".

5.2.3.25 ply_laminate_definition

A **ply_laminate_definition** is **product_definition** containing an ordered list of **ply_laminate_sequence_-definition**s. This ordered list is defined by the use of **next_assembly_usage_occurrence**.

NOTE 1 The valid constituents are ply products.

NOTE 2 This table is only valid to describe a ply_laminate product.

EXPRESS specification:

```
*)
ENTITY ply_laminate_definition
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN (SELF,
                             'PRODUCT_DEFINITION_SCHEMA.' +
                             'PRODUCT_DEFINITION_RELATIONSHIP.' +
                             'RELATING_PRODUCT_DEFINITION') |
                             'PRODUCT_STRUCTURE_SCHEMA.' +
                             'NEXT_ASSEMBLY_USAGE_OCCURRENCE' IN
                             TYPEOF (pdr))) = 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **ply_laminate_definition** shall be used by one **next_assembly_usage_occurrence**.

NOTE The first next_assembly_usage_occurrence.relating_product_definition shall be the ply_laminate_definition, and the next_assembly_usage_occurrence.related_product_definition shall be the first ply_laminate_sequence_-definition in the table. Thereafter the subsequent next_assembly_usage_occurrences chain together ply_laminate_-sequence_definition entities into an ordered table.

5.2.3.26 ply_laminate_sequence_definition

A **ply_laminate_sequence_definition** is a **product_definition** of a unique sequence or layer with a ply laminate product. This sequence or layer contains one or more ply where a ply is constant thickness material with a contiguous structure and a defined boundary.

The **next_assembly_usage_occurrence** establishes the ply constituents of this **product_definition**.

EXPRESS specification:

```
*)
ENTITY ply_laminate_sequence_definition
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN (SELF,
                             'PRODUCT_DEFINITION_SCHEMA.' +
                             'PRODUCT_DEFINITION_RELATIONSHIP.' +
                             'RELATING_PRODUCT_DEFINITION') |
                             'PRODUCT_STRUCTURE_SCHEMA.' +
                             'NEXT_ASSEMBLY_USAGE_OCCURRENCE' IN
                             TYPEOF (pdr))) > 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **ply_laminate_sequence_definition** shall be used by one or many **next_assembly_-usage_occurrence**.

NOTE The next_assembly_usage_occurrence.relating_product_definition shall be the ply_laminate_sequence_-definition. The next_assembly_usage_occurrence.related_product_definition thus associated comprise the SET of product_definition entities (that are mappings of ply ARM objects) for the table.

5.2.3.27 point_and_vector

A **point_and_vector** is a type of shape representation specifying a position in space together with two directions.

EXPRESS specification:

```
*)
ENTITY point_and_vector
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (SELF\representation.items) = 3;
  WR2: ('GEOMETRY_SCHEMA.POINT'
        IN TYPEOF (SELF\representation.items[1])) AND
       ('GEOMETRY_SCHEMA.DIRECTION'
        IN TYPEOF (SELF\representation.items[2])) AND
       ('GEOMETRY_SCHEMA.DIRECTION'
        IN TYPEOF (SELF\representation.items[3]));
END_ENTITY;
(*
```

Formal propositions:

WR1: The set of **items** of a **point_and_vector** shall contain three **representation_item**s.

WR2: The first **representation_item** shall be a **point**, and the second and third **representation_item**s shall be a **direction**.

5.2.3.28 point_path

A **point_path** is a list of points and associated directions specifying a path across a **ply_laminate surface** over which a **ply_laminate** will be laid. It is used to define the **ply_orientation_angle** of the major and minor directions at each point along the path. This ordered list is defined by the use of **representation_-relationship**.

EXPRESS specification:

```
*)
ENTITY point_path
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (USEDIN (SELF, 'REPRESENTATION_SCHEMA.' +
                             'REPRESENTATION_RELATIONSHIP.REP_1')) = 1;
END_ENTITY;
(*
```

WR1: A **point_path** shall be used by one **representation_relationship**.

NOTE The first representation_relationship.rep_1 shall be the point_path, and the representation_relationship.rep_2 shall be the first point_and_vector in the list. Thereafter the subsequent representation_relationships chain together point_and_vector entities into an ordered list.

5.2.3.29 point_representation

A **point_representation** is an aggregation of one or more nodes in a **node_set**, and therefore their associated points, that establishes the shape and location of one or more elements in a finite element model.

EXPRESS specification:

```
*)
ENTITY point_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY ( it <* SELF\representation.items |
       (NOT (SIZEOF (
       ['STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.NODE_SET',
        'REPRESENTATION_SCHEMA.MAPPED_ITEM',
        'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
        'FEA_AXIS2_PLACEMENT_3D'] *
        TYPEOF (it)) = 1 )))) = 0;
  WR2: SIZEOF (QUERY ( it <* SELF\representation.items |
       (SIZEOF (
       ['STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.NODE_SET',
        'REPRESENTATION_SCHEMA.MAPPED_ITEM'] *
        TYPEOF (it)) = 1 ))) >= 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: The set of **items** in a **point_representation** shall contain only **node_set**, **mapped_item**, or **fea_-axis2_placement_3d** objects.

WR2: There shall be at least one **node_set** or **mapped_item** in the set of **items** in a **point_representation**.

5.2.3.30 retention

A **retention** is a type of **action** that defines a period of time that product data needs to be retained due to organizational policy or to legal requirements.

EXPRESS specification:

```
*)
ENTITY retention
  SUBTYPE OF (action);
WHERE
  WR1: SIZEOF (QUERY (aa <* USEDIN (SELF,
         'MANAGEMENT_RESOURCES_SCHEMA.' +
         'ACTION_ASSIGNMENT.ASSIGNED_ACTION') |
         'STRUCTURAL_ANALYSIS_DESIGN.RETENTION_ASSIGNMENT'
         IN TYPEOF (aa) )) > 0;
  WR2: SIZEOF (QUERY (ada <* USEDIN (SELF,
         'STRUCTURAL_ANALYSIS_DESIGN.' +
         'APPLIED_DATE_ASSIGNMENT.ITEMS') |
         ada.role.name = 'start date')) +
       SIZEOF (QUERY (adata <* USEDIN (SELF,
         'STRUCTURAL_ANALYSIS_DESIGN.' +
         'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS') |
         adata.role.name = 'start date')) = 1;
  WR3: SIZEOF (QUERY (ada <* USEDIN (SELF,
         'STRUCTURAL_ANALYSIS_DESIGN.' +
         'APPLIED_DATE_ASSIGNMENT.ITEMS') |
         ada.role.name = 'earliest end date')) +
       SIZEOF (QUERY (adata <* USEDIN (SELF,
         'STRUCTURAL_ANALYSIS_DESIGN.' +
         'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS') |
         adata.role.name = 'earliest end date')) = 1;
  WR4: SIZEOF (QUERY (ada <* USEDIN (SELF,
         'STRUCTURAL_ANALYSIS_DESIGN.' +
         'APPLIED_DATE_ASSIGNMENT.ITEMS') |
         ada.role.name = 'latest end date')) +
       SIZEOF (QUERY (adata <* USEDIN (SELF,
         'STRUCTURAL_ANALYSIS_DESIGN.' +
         'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS') |
         adata.role.name = 'latest end date')) = 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: The **retention** shall be referenced as the **assigned_action** by at least one instance of **retention_-assignment**.

WR2: The **retention** shall either be contained in the set of **items** of an **applied_date_assignment** that references as its **role** a **date_role** with a **name** of 'start date' or shall be contained in the set of **items** of an **applied_date_and_time_assignment** that references as its **role** a **date_time_role** with a **name** of 'start date'.

WR3: The **retention** shall either be contained in the set of **items** of an **applied_date_assignment** that references as its role a **date_role** with a **name** of 'earliest end date' or shall be contained in the set of **items** of an **applied_date_and_time_assignment** that references as its **role** a **date_time_role** with a **name** of 'earliest end date'.

WR4: The **retention** shall either be contained in the set of **items** of an **applied_date_assignment** that references as its role a **date_role** with a **name** of 'latest end date' or shall be contained in the set of items of an **applied_date_and_time_assignment** that references as its **role** a **date_time_role** with a **name** of 'latest end date'.

5.2.3.31 retention_assignment

A **retention_assignment** is a type of **action_assignment** that assigns a **retention** to a **retention_item** that needs to be retained.

EXPRESS specification:

```
*)
ENTITY retention_assignment
  SUBTYPE OF (action_assignment);
  items : SET [1:?] OF retention_item;
WHERE
  WR1: 'STRUCTURAL_ANALYSIS_DESIGN.RETENTION' IN
        TYPEOF (SELF.assigned_action);
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **retention_item**s which identifies the product data to be retained.

Formal propositions:

WR1: The **retention_assignment** shall reference as its **assigned_action** an **action** that is of type **retention**.

5.2.3.32 smeared_material_definition

A **smeared_material_definition** is an idealized **product_definition** of a laminate product that smears the constituent materials together into a single definition that is representative of the whole.

EXPRESS specification:

```
*)
ENTITY smeared_material_definition
  SUBTYPE OF (product_definition);
END_ENTITY;
(*
```

5.2.3.33 supplied_part_relationship

A **supplied_part_relationship** relates the identifications of two **product**s and specifies that one is the identification used by the internal design organization and the other is the identification used by a supplier.

EXPRESS specification:

```
*)
ENTITY supplied_part_relationship
  SUBTYPE OF (product_definition_relationship);
```

```
END_ENTITY;
(*
```

Attribute definitions:

SELF\product_definition_relationship.relating_product_definition: the **product_definition** that is the design organization's definition and identification of the **product**.

SELF\product_definition_relationship.related_product_definition: the **product_definition** that is definition and identification of the product for the organization which supplies the part or design to the design organization.

Informal propositions:

IP1: The shapes for design organization's definition and the supplying organization's definition shall be equal with respect to form, fit, and function.

5.2.3.34 thickness_laminate_definition

A **thickness_laminate_definition** is a **product_definition** containing an ordered list of constituents applicable for a particular **shape_aspect** of a product. The ordered list is defined by use of **next_assembly_usage_occurrence**.

NOTE The definition may be of the actual products such as plies or may be an idealized representation where multiple layers or core is represented as a ply with a thickness defined. For the first case the thickness would be defined from the material.

EXPRESS specification:

```
*)
ENTITY thickness_laminate_definition
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN (SELF,
                             'PRODUCT_DEFINITION_SCHEMA.' +
                             'PRODUCT_DEFINITION_RELATIONSHIP.' +
                             'RELATING_PRODUCT_DEFINITION') |
                             'PRODUCT_STRUCTURE_SCHEMA.' +
                             'NEXT_ASSEMBLY_USAGE_OCCURRENCE' IN
                             TYPEOF (pdr))) = 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **thickness_laminate_definition** shall be used by one **next_assembly_usage_occurrence**.

NOTE The first next_assembly_usage_occurrence.relating_product_definition shall be the thickness_laminate_-definition, and the next_assembly_usage_occurrence.related_product_definition shall be the first thickness_laminate_-table_component ARM object in the table. Thereafter the subsequent next_assembly_usage_occurrences chain together product_definition entities (that are thickness_laminate_table_component ARM objects) into an ordered table.

5.2.3.35 transformation_with_derived_angle

A **transformation_with_derived_angle** is an **item_defined_transformation** that defines the primary reinforcement orientation direction within a ply at a particular location within a laminate. The angle is defined by the transformation from an **axis2_placement_3d** that defines the laminate reference directions and a second **axis2_placement_3d** that defines the orientation of the reinforcement.

EXAMPLE The warp direction of a woven fabric has a direction defined by **axis2_placement_3d** and the orientation angle is defined relative to a reinforcement orientation basis by a **transformation_with_derived_angle.**

EXPRESS specification:

```
*)
ENTITY transformation_with_derived_angle
  SUBTYPE OF (item_defined_transformation);
DERIVE
  orientation_angle : plane_angle_measure := derive_angle (
                      SELF\item_defined_transformation.
                           transform_item_1,
                      SELF\item_defined_transformation.
                           transform_item_2);
WHERE
  WR1: 'GEOMETRY_SCHEMA.AXIS2_PLACEMENT_3D' IN
       TYPEOF(SELF\item_defined_transformation.transform_item_1);
  WR2: 'GEOMETRY_SCHEMA.AXIS2_PLACEMENT_3D' IN
       TYPEOF(SELF\item_defined_transformation.transform_item_2);
  WR3: (SELF\item_defined_transformation.transform_item_1\
             axis2_placement_3d.p[3].direction_ratios[1] =
        SELF\item_defined_transformation.transform_item_2\
             axis2_placement_3d.p[3].direction_ratios[1])
       AND
       (SELF\item_defined_transformation.transform_item_1\
             axis2_placement_3d.p[3].direction_ratios[2] =
        SELF\item_defined_transformation.transform_item_2\
             axis2_placement_3d.p[3].direction_ratios[2])
       AND
       (SELF\item_defined_transformation.transform_item_1\
             axis2_placement_3d.p[3].direction_ratios[3] =
        SELF\item_defined_transformation.transform_item_2\
             axis2_placement_3d.p[3].direction_ratios[3]);
END_ENTITY;
(*
```

Attribute definitions:

orientation_angle: the angle between the two directions.

Informal propositions:

IP1: If a **curve** is used to specify the laminate reference directions, an **axis2_placement_3d** shall be associated with the direction tangent to the **curve** at the point where the angle is to be derived.

IP2: If a **point_path** is used to specify the laminate reference directions, **axis2_placement_3d**s shall be associated with the major and minor directions of the set of **point_and_vector**s associated with the **point_-path**.

Formal propositions:

WR1: The first item in the transformation shall be an **axis2_placement_3d**.

WR2: The second item in the transformation shall be an **axis2_placement_3d**.

WR3: Direction 3 for both **axis2_placement_3d**s shall be3 the same.

5.2.4 AP209 imported entity modifications

5.2.4.1 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 rule:

The following global rule defined in this part of ISO 10303 applies to the **action** entity:

— subtype_mandatory_action (see 5.2.5.46).

5.2.4.2 action_directive

The base definition of the **action_directive** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **action** entity:

— action_directive_requires_date_time (see 5.2.5.1).

— unique_version_change_order_rule (see 5.2.5.49).

5.2.4.3 action_request_status

The base definition of the **action_request_status** 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_status** entity:

— restrict_action_request_status (see 5.2.5.38);

— versioned_action_request_requires_status (see 5.2.5.50).

5.2.4.4 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 **approval** entity:

— approval_requires_approval_date_time (see 5.2.5.4);

— approval_requires_approval_person_organization (see 5.2.5.5);

— approvals_are_assigned (see 5.2.5.6).

5.2.4.5 approval_assignment

The base definition of the **approval_assignment** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **approval_assignment** entity:

— approvals_are_assigned (see 5.2.5.6).

5.2.4.6 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 rule:

The following global rule defined in this part of ISO 10303 applies to the **approval_date_time** entity:

— approval_requires_approval_date_time (see 5.2.5.4).

5.2.4.7 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 rule:

The following global rule defined in this part of ISO 10303 applies to the **approval_person_organization** entity:

— approval_requires_approval_person_organization (see 5.2.5.5).

5.2.4.8 approval_status

The base definition of the **approval_status** 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 **approval_status** entity:

— dependent_instantiable_approval_status (see 5.2.5.11);

— restrict_approval_status (see 5.2.5.39).

5.2.4.9 certification_type

The base definition of the **certification_type** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **certification_type** entity:

— dependent_instantiable_certification_type (see 5.2.5.12).

5.2.4.10 configuration_item

The base definition of the **configuration_item** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **configuration_item** entity:

— configuration_item_requires_person_organization (see 5.2.5.9).

5.2.4.11 contract_type

The base definition of the **contract_type** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **contract_type** entity:

— dependent_instantiable_contract_type (see 5.2.5.13).

5.2.4.12 date

The base definition of the **date** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **date** entity:

— dependent_instantiable_date (see 5.2.5.14).

5.2.4.13 date_and_time

The base definition of the **date_and_time** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **date_and_time** entity:

— dependent_instantiable_date_and_time (see 5.2.5.15).

5.2.4.14 date_role

The base definition of the **date_role** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rules defined in this part of ISO 10303 applies to the **date_role** entity:

— dependent_instantiable_date_role (see 5.2.5.16);

— restrict_date_time_role (see 5.2.5.40).

5.2.4.15 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:

— dependent_instantiable_date_time_role (see 5.2.5.17);

— restrict_date_time_role (see 5.2.5.40).

5.2.4.16 document_type

The base definition of the **document_type** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **document_type** entity:

— dependent_instantiable_document_type (see 5.2.5.18).

5.2.4.17 global_unit_assigned_context

The base definition of the **global_unit_assigned_context** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **global_unit_assigned_context** entity:

— global_unit_assignment (see 5.2.5.26).

5.2.4.18 make_from_usage_option

The base definition of the **make_from_usage_option** entity is given in ISO 10303-44. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rules defined in this part of ISO 10303 applies to the **make_from_usage_option** entity:

— component_class_for_assembly_select (see 5.2.5.8);

— ply_reference (see 5.2.5.28);

— ply_stock_material_select (see 5.2.5.29);

— stock_material_reference (see 5.2.5.45);

— thickness_laminate_table_component_select (see 5.2.5.48);

5.2.4.19 measure_with_unit

The base definition of the **measure_with_unit** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **measure_with_unit** entity:

— as_required_quantity (see 5.2.5.7).

5.2.4.20 named_unit

The base definition of the **named_unit** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **named_unit** entity:

— dependent_instantiable_named_unit (see 5.2.5.19).

5.2.4.21 next_assembly_usage_occurrence

The base definition of the **next_assembly_usage_occurrence** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **next_assembly_usage_occurrence** entity:

— coordinated_assembly_and_shape (see 5.2.5.10).

5.2.4.22 organization_role

The base definition of the **organization_role** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rules defined in this part of ISO 10303 applies to the **organization_role** entity:

— dependent_instantiable_organization_role (see 5.2.5.20);

— restrict_person_organization_role (see 5.2.5.41).

5.2.4.23 parametric_representation_context

The base definition of the **parametric_representation_context** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **parametric_representation_-context** entity:

— dependent_instantiable_parametric_representation_context (see 5.2.5.21).

5.2.4.24 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:

— dependent_instantiable_person_and_organization_role (see 5.2.5.22);

— restrict_person_organization_role (see 5.2.5.41).

5.2.4.25 product

The base definition of the **product** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 applies to the **product** entity:

— product_requires_person_organization (see 5.2.5.33);

— product_requires_product_category (see 5.2.5.34);

— product_requires_version (see 5.2.5.35).

5.2.4.26 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:

—product_definition_requires_date_time (see 5.2.5.31);

—product_definition_requires_person_organization (see 5.2.5.32).

5.2.4.27 product_definition_context

The base definition of the **product_definition_context** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **product_definition_context** entity:

— restrict_product_definition_context (see 5.2.5.43).

5.2.4.28 product_definition_formation

The base definition of the **product_definition_formation** 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_formation** entity:

— product_requires_version (see 5.2.5.35);

— product_version_requires_person_organization (see 5.2.5.36);

— product_version_requires_security_classification (see 5.2.5.37).

5.2.4.29 product_definition_usage

The base definition of the **product_definition_usage** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rule:

The following global rule defined in this part of ISO 10303 applies to the **product_definition_usage** entity:

—pdu_requires_security_classification (see 5.2.5.27).

5.2.4.30 product_related_product_category

The base definition of the **product_related_product_category** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 applies to the **product_related_product_-category** entity:

— component_class_for_assembly_select (see 5.2.5.8);

— ply_reference (see 5.2.5.28);

— ply_stock_material_select (see 5.2.5.29);

— product_requires_product_category (see 5.2.5.34);

— restrict_product_category_value (see 5.2.5.42);

— stock_material_reference (see 5.2.5.45);

— thickness_laminate_table_component_select (see 5.2.5.48).

5.2.4.31 representation_item

The base definition of the **representation_item** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

<u>Associated global rule:</u>

The following global rule defined in this part of ISO 10303 applies to the **representation_item** entity:

— dependent_instantiable_representation_item (see 5.2.5.23).

5.2.4.32 security_classification_level

The base definition of the **security_classification_level** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

<u>Associated global rules:</u>

The following global rules defined in this part of ISO 10303 applies to the **security_classification_level** entity:

— dependent_instantiable_security_classification_level (see 5.2.5.24);

— restrict_security_classification_level (see 5.2.5.44).

5.2.4.33 shape_representation

The base definition of the **shape_representation** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

<u>Associated global rules:</u>

The following global rules defined in this part of ISO 10303 applies to the **shape_representation** entity:

— dependent_instantiable_shape_representation (see 5.2.5.25);

— subtype_mandatory_shape_representation (see 5.2.5.47).

5.2.4.34 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 rule:

The following global rule defined in this part of ISO 10303 applies to the **versioned_action_request** entity:

—action_request_requires_date_time (see 5.2.5.2).

— action_request_requires_person_organization (see 5.2.5.3).

— versioned_action_request_requires_status (see 5.2.5.50).

5.2.5 AP209 rule definitions

5.2.5.1 action_directive_requires_date_time

The **action_directive_requires_date_time** rule specifies that each instance of an **action_directive** shall be referenced by exactly one **applied_date_and_time_assignment** or **applied_date_assignment**. This rule enforces the requirement for every **action_directive** to have a date on which work is begun for a new design or analysis, or for a change to an existing design or analysis.

NOTE The coordination of the different role values with the assignment of **date_time_assignment** to different entities is specified in the **applied_date_and_time_correlation** function. This function is invoked locally to **applied_date_-and_time_assignment**. The coordination of the different role values with the assignment of **date_assignment** to different entities is specified in the **applied_date_correlation** function. This function is invoked locally to **applied_-date_assignment**.

EXPRESS specification:

```
*)
RULE action_directive_requires_date_time FOR
 (action_directive, applied_date_and_time_assignment,
  applied_date_assignment);
WHERE
  WR1: SIZEOF (QUERY (ad <* action_directive |
       NOT ((SIZEOF (QUERY (adta <* applied_date_and_time_assignment |
                    (ad IN adta.items) AND
                    (adta.role.name IN ['change date', 'start date']))) +
             SIZEOF (QUERY (ada <* applied_date_assignment |
                    (ad IN ada.items) AND
                    (ada.role.name IN ['change date', 'start date'])))
             ) = 1 ))) = 0;
END_RULE;
(*
```


Argument definitions:

action_directive: identifies the set of all instances of **action_directive** entities.

applied_date_and_time_assignment: identifies the set of all instances of **applied_date_and_time_-assignment** entities.

applied_date_assignment: identifies the set of all instances of **applied_date_assignment** entities.

Formal propositions:

WR1: For each instance of **action_directive**, there shall be exactly one instance of **applied_date_and_-time_assignment** or **applied_date_assignment** that contain the instance of **action_directive** in its set of **items**.

5.2.5.2 action_request_requires_date_time

The **action_request_requires_date_time** rule specifies that each instance of a **versioned_action_request** shall be referenced by exactly one **applied_date_and_time_assignment** or **applied_date_assignment**. This rule enforces the requirement for every **versioned_action_request** to have a request date indicating the date on which the **versioned_action_request** was issued.

NOTE The coordination of the different role values with the assignment of **date_time_assignment** to different entities is specified in the **applied_date_and_time_correlation** function. This function is invoked locally to **applied_date_-and_time_assignment**. The coordination of the different role values with the assignment of **date_assignment** to different entities is specified in the **applied_date_correlation** function. This function is invoked locally to **applied_-date_assignment**.

EXPRESS specification:

```
*)
RULE action_request_requires_date_time FOR
 (versioned_action_request, applied_date_and_time_assignment,
  applied_date_assignment);
WHERE
  WR1: SIZEOF (QUERY (ar <* versioned_action_request |
       NOT ((SIZEOF (QUERY (adta <* applied_date_and_time_assignment |
                     ar IN adta.items)) +
             SIZEOF (QUERY (ada <* applied_date_assignment |
                     ar IN ada.items))) = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

versioned_action_request: identifies the set of all instances of **versioned_action_request** entities.

applied_date_and_time_assignment: identifies the set of all instances of **applied_date_and_time_-assignment** entities.

applied_date_assignment: identifies the set of all instances of **applied_date_assignment** entities.

Formal propositions:

WR1: For each instance of **versioned_action_request**, there shall be exactly one instance of **applied_-date_and_time_assignment** or **applied_date_assignment** that contains the instance of **versioned_action_-request** in its set of **items**.

5.2.5.3 action_request_requires_person_organization

The **action_request_requires_person_organization** rule specifies that every **versioned_action_request** shall be referenced by at least one **applied_person_and_organization_assignment** or **applied_-organization_assignment**. This rule specifies the need for every **versioned_action_request** to have at least one recipient who receives the **versioned_action_request**.

NOTE The coordination of the different role values with the assignment of **person_organization_assignment** to different entities is specified in the **applied_person_and_organization_correlation** function. This function is invoked locally to **applied_person_and_organization_assignment**. The coordination of the different role values with the assignment of **organization_assignment** to different entities is specified in the **applied_organization_correlation** function. This function is invoked locally to **applied_organization_assignment**.

EXPRESS specification:

```
*)
RULE action_request_requires_person_organization FOR
 (versioned_action_request, applied_person_and_organization_assignment,
  applied_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (ar <* versioned_action_request |
       NOT ((SIZEOF (QUERY (apoa <*
                     applied_person_and_organization_assignment |
                     (ar IN apoa.items) AND
                     (apoa.role.name = 'request recipient'))) +
             SIZEOF (QUERY (aoa <* applied_organization_assignment |
                     (ar IN aoa.items) AND
                     (aoa.role.name = 'request recipient')))) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

versioned_action_request: identifies the set of all instances of **versioned_action_request** entities.

applied_person_and_organization_assignment: identifies the set of all instances of **applied_person_-and_organization_assignment** entities.

applied_organization_assignment: identifies the set of all instances of **applied_organization_assignment** entities.

Formal propositions:

WR1: For each instance of **versioned_action_request**, there shall be at least one instance of **applied_-person_and_organization_assignment** or **applied_organization_assignment** that contains the instance of **versioned_action_request** in its set of **items**.

5.2.5.4 approval_requires_approval_date_time

The **approval_requires_approval_date_time** rule specifies that each instance of **approval** shall be referenced by exactly one **approval_date_time**. This rule enforces the requirement for every approval to have a date on which the approval obtained its specified status.

EXPRESS specification:

```
*)
RULE approval_requires_approval_date_time FOR
 (approval, approval_date_time);
WHERE
  WR1: SIZEOF (QUERY (e <* approval |
       NOT (SIZEOF (QUERY (s <* approval_date_time |
       e :=: s.dated_approval)) = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

approval: identifies the set of all instances of **approval** entities.

approval_date_time: identifies the set of all instances of **approval_date_time** entities.

Formal propositions:

WR1: For each instance of **approval**, there shall be exactly one instance of **approval_date_time** that contains the instance of **approval** as its **dated_approval** attribute.

5.2.5.5 approval_requires_approval_person_organization

The **approval_requires_approval_person_organization** specifies that each instance of **approval** shall have at least one **approval_person_organization** referencing it. This rule enforces the requirement for an approval to be authorized by one or more people within their organizations.

EXPRESS specification:

```
*)
RULE approval_requires_approval_person_organization FOR
 (approval, approval_person_organization);
WHERE
  WR1: SIZEOF (QUERY (e <* approval |
       NOT (SIZEOF (QUERY (s <* approval_person_organization |
       e :=: s.authorized_approval)) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

approval: identifies the set of all instances of **approval** entities.

approval_person_organization: identifies the set of all instances of **approval_person_organization** entities.

Formal propositions:

WR1: For each instance of **approval**, there shall be one or more instances of **approval_person_-organization** that contains the instance of **approval** as its **authorized_approval** attribute.

5.2.5.6 approvals_are_assigned

The **approvals_are_assigned** rule specifies that each instance of **approval** shall be assigned by at least one instance of approval_assignment entity.

EXPRESS specification:

```
*)
RULE approvals_are_assigned FOR
 (approval, approval_assignment);
WHERE
  WR1: SIZEOF (QUERY (app <* approval |
       NOT (SIZEOF (QUERY (aa <* approval_assignment |
       app :=: aa.assigned_approval)) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

approval: identifies the set of all instances of **approval** entities.

approval_assignment: identifies the set of all instances of **approval_assignment** entities.

Formal propositions:

WR1: For each instance of **approval**, there shall be one or more instances of **approval_assignment** which contains the instance of **approval** as its **assigned_approval** attribute.

5.2.5.7 as_required_quantity

The **as_required_quantity** rule specifies the use of the **descriptive_measure** type in the **measure_with_-unit** entity. The value of the **descriptive_measure** STRING type shall always be "as required". This rule enforces the requirement for the specification of the use of an amount to be as required.

EXPRESS specification:

```
*)
RULE as_required_quantity FOR
 (measure_with_unit);
WHERE
  WR1: SIZEOF (QUERY (m <* measure_with_unit |
       ('MEASURE_SCHEMA.DESCRIPTIVE_MEASURE'
        IN TYPEOF (m.value_component)) AND
       (NOT (m.value_component = 'as required')))) = 0;
END_RULE;
(*
```

Argument definitions:

measure_with_unit: identifies the set of all instances of **measure_with_unit** entities.

Formal propositions:

WR1: For each instance of **measure_with_unit**, if the type of the attribute **value** is **descriptive_measure** then the value of the attribute shall be "as required".

5.2.5.8 component_class_for_assembly_select

The **component_class_for_assembly_select** rule states that **product**s associated with a **composite_-assembly_sequence_definition** shall have a **product_related_product_category.name** of "ply", "ply laminate", "filament laminate", "processed core", or "composite assembly".

EXPRESS specification:

```
*)
RULE component_class_for_assembly_select FOR
 (composite_assembly_sequence_definition,
  next_assembly_usage_occurrence,
  product_related_product_category);
LOCAL
  i,j,k :  INTEGER;
  dkuhr :  LOGICAL;
  nnauo :  INTEGER;
  nprpc :  INTEGER;
  rp    :  product;
END_LOCAL;
  dkuhr := TRUE;
  REPEAT i:= LOINDEX (composite_assembly_sequence_definition) TO
            HIINDEX (composite_assembly_sequence_definition);
    nnauo := 0;
    REPEAT j:= LOINDEX (next_assembly_usage_occurrence) TO
              HIINDEX (next_assembly_usage_occurrence);
      IF (composite_assembly_sequence_definition[i] =
        next_assembly_usage_occurrence[j].relating_product_definition) THEN
        rp := next_assembly_usage_occurrence[j].related_product_definition.
              formation.of_product;
        nprpc := 0;
        REPEAT k:= LOINDEX (product_related_product_category) TO
                  HIINDEX (product_related_product_category);
          IF ((rp IN product_related_product_category[k].products) AND
              (product_related_product_category[k].name IN ['ply',
               'ply laminate', 'filament laminate', 'processed core',
               'composite assembly'])) THEN
              nprpc := nprpc + 1;
          END_IF;
        END_REPEAT;
        IF (nprpc = 1) THEN
          nnauo := nnauo + 1;
        ELSE
          dkuhr := FALSE;
          ESCAPE;
        END_IF;
      END_IF;

    END_REPEAT;
    IF (dkuhr = FALSE) THEN
```

```
      ESCAPE;
    END_IF;
    IF(nnauo = 0) THEN
      dkuhr := FALSE;
      ESCAPE;
    END_IF;
  END_REPEAT;
WHERE
  WR1:  dkuhr;
END_RULE;
(*
```

Argument definitions:

composite_assembly_sequence_definition: identifies the set of all instances of **composite_assembly_-sequence_definition** entities.

next_assembly_usage_occurrence: identifies the set of all instances of **next_assembly_usage_occurrence** entities.

product_related_product_category: identifies the set of all instances of **product_related_product_-category** entities.

Formal propositions:

WR1: **Composite_assembly_sequence_definition**s shall be related through at least one **next_assembly_usage_occurrence** relationship to **product_definition**s which are associated with **product**s that are in the **product_related_product_category** with the **name** atttribute of "ply", "ply laminate", "filament laminate", "composite assembly", or "processed core".

5.2.5.9 configuration_item_requires_person_organization

The **configuration_item_requires_person_organization** rule specifies that every **configuration_item** shall be referenced by exactly one **applied_person_and_organization_assignment** or **applied_-organization_assignment**. This rule specifies the need for every **configuration_item** to have a configuration manager who is responsible for it.

NOTE The coordination of the different role values with the assignment of **person_organization_assignment** to different entities is specified in the **applied_person_and_organization_correlation** function. This function is invoked locally to **applied_person_and_organization_assignment**. The coordination of the different role values with the assignment of **organization_assignment** to different entities is specified in the **applied_organization_correlation** function. This function is invoked locally to **applied_organization_assignment**.

EXPRESS specification:

```
*)
RULE configuration_item_requires_person_organization FOR
 (configuration_item, applied_person_and_organization_assignment,
applied_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (ci <* configuration_item |
       NOT ((SIZEOF (QUERY (apoa <*
                    applied_person_and_organization_assignment |
                    ci IN apoa.items)) +
```



```
              SIZEOF (QUERY (aoa <* applied_organization_assignment |
                     ci IN aoa.items))) = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

configuration_item: identifies the set of all instances of **configuration_item** entities.

applied_person_and_organization_assignment: identifies the set of all instances of **applied_person_-and_organization_assignment** entities.

applied_organization_assignment: identifies the set of all instances of **applied_organization_assignment** entities.

Formal propositions:

WR1: For each instance of **configuration_item**, there shall be exactly one instance of **applied_person_-and_organization_assignment** or **applied_organization_assignment** that contains the instance of **configuration_item** in its set of **items**.

5.2.5.10 coordinated_assembly_and_shape

The **coordinated_assembly_and_shape** rule specifies that the relationship between two **product_-definition** entities that are an assembly and a component in a **next_assembly_usage_occurrence** and the relationship between two **shape_representation** entities that contain the representation of the shapes of the assembly and the component in a **shape_representation_relationship** shall be explicitly related through the **context_dependent_shape_representation**. This rule calls the function **assembly_shape_is_defined** that returns true if the assembly relationship and shape relationship are explicitly related.

EXPRESS specification:

```
*)
RULE coordinated_assembly_and_shape FOR
 (next_assembly_usage_occurrence);
WHERE
  WR1: SIZEOF (QUERY (nauo <* next_assembly_usage_occurrence |
       NOT (assembly_shape_is_defined (nauo)))) = 0;
END_RULE;
(*
```

Argument definitions:

next_assembly_usage_occurrence: identifies the set of all instances of **next_assembly_usage_occurrence** entities.

Formal propositions:

WR1: For each of instance of **next_assembly_usage_occurrence**, the **assembly_shape_is_defined** function shall be true.

5.2.5.11 dependent_instantiable_approval_status

The **dependent_instantiable_approval_status** rule specifies that all instances of **approval_status** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_approval_status FOR
 (approval_status);
WHERE
  WR1: SIZEOF (QUERY (ast <* approval_status |
       NOT (SIZEOF (USEDIN (ast, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

approval_status: identifies the set of all instances of **approval_status**.

Formal propositions:

WR1: For each instance of **approval_status**, there shall be a reference to the **approval_status** instance from an attribute of another entity.

5.2.5.12 dependent_instantiable_certification_type

The **dependent_instantiable_certification_type** rule specifies that all instances of **certification_type** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_certification_type FOR
 (certification_type);
WHERE
  WR1: SIZEOF (QUERY (ct <* certification_type |
       NOT (SIZEOF (USEDIN (ct, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

certification_type: identifies the set of all instances of **certification_type**.

Formal propositions:

WR1: For each instance of **certification_type**, there shall be a reference to the **certification_type** instance from an attribute of another entity.

5.2.5.13 dependent_instantiable_contract_type

The **dependent_instantiable_contract_type** rule specifies that all instances of **contract_type** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_contract_type FOR
 (contract_type);
WHERE
  WR1: SIZEOF (QUERY (ct <* contract_type |
       NOT (SIZEOF (USEDIN (ct, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

contract_type: identifies the set of all instances of **contract_type**.

Formal propositions:

WR1: For each instance of **contract_type**, there shall be a reference to the **contract_type** instance from an attribute of another entity.

5.2.5.14 dependent_instantiable_date

The **dependent_instantiable_date** rule specifies that all instances of **date** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_date FOR
 (date);
WHERE
  WR1: SIZEOF (QUERY (dt <* date |
       NOT (SIZEOF (USEDIN (dt, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

date: identifies the set of all instances of **date**.

Formal propositions:

WR1: For each instance of **date**, there shall be a reference to the **date** instance from an attribute of another entity.

5.2.5.15 dependent_instantiable_date_and_time

The **dependent_instantiable_date_and_time** rule specifies that all instances of **date_and_time** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_date_and_time FOR
 (date_and_time);
WHERE
  WR1: SIZEOF (QUERY (dat <* date_and_time |
       NOT (SIZEOF (USEDIN (dat, ''')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

date_and_time: identifies the set of all instances of **date_and_time**.

Formal propositions:

WR1: For each instance of **date_and_time**, there shall be a reference to the **date_and_time** instance from an attribute of another entity.

5.2.5.16 dependent_instantiable_date_role

The **dependent_instantiable_date_role** rule specifies that all instances of **date_role** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_date_role FOR
 (date_role);
WHERE
  WR1: SIZEOF (QUERY (dr <* date_role |
       NOT (SIZEOF (USEDIN (dr, ''')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

date_role: identifies the set of all instances of **date_role**.

Formal propositions:

WR1: For each instance of **date_role**, there shall be a reference to the **date_role** instance from an attribute of another entity.

5.2.5.17 dependent_instantiable_date_time_role

The **dependent_instantiable_date_time_role** rule specifies that all instances of **date_time_role** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_date_time_role FOR
 (date_time_role);
WHERE
  WR1: SIZEOF (QUERY (dtr <* date_time_role |
       NOT (SIZEOF (USEDIN (dtr, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

date_time_role: identifies the set of all instances of **date_time_role**.

Formal propositions:

WR1: For each instance of **date_time_role**, there shall be a reference to the **date_time_role** instance from an attribute of another entity.

5.2.5.18 dependent_instantiable_document_type

The **dependent_instantiable_document_type** rule specifies that all instances of **document_type** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_document_type FOR
 (document_type);
WHERE
  WR1: SIZEOF (QUERY (dt <* document_type |
       NOT (SIZEOF (USEDIN (dt, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

document_type: identifies the set of all instances of **document_type**.

Formal propositions:

WR1: For each instance of **document_type**, there shall be a reference to the **document_type** instance from an attribute of another entity.

5.2.5.19 dependent_instantiable_named_unit

The **dependent_instantiable_named_unit** rule specifies that all instances of **named_unit** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_named_unit FOR
 (named_unit);
WHERE
  WR1: SIZEOF (QUERY (nu <* named_unit |
       NOT (SIZEOF (USEDIN (nu, ''')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

named_unit: identifies the set of all instances of **named_unit**.

Formal propositions:

WR1: For each instance of **named_unit**, there shall be a reference to the **named_unit** instance from an attribute of another entity.

5.2.5.20 dependent_instantiable_organization_role

The **dependent_instantiable_organization_role** rule specifies that all instances of **organization_role** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_organization_role FOR
 (organization_role);
WHERE
  WR1: SIZEOF (QUERY (orl <* organization_role |
       NOT (SIZEOF (USEDIN (orl, ''')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

organization_role: identifies the set of all instances of **organization_role**.

Formal propositions:

WR1: For each instance of **organization_role**, there shall be a reference to the **organization_role** instance from an attribute of another entity.

5.2.5.21 dependent_instantiable_parametric_representation_context

The **dependent_instantiable_parametric_representation_context** rule specifies that all instances of **parametric_representation_context** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_parametric_representation_context FOR
 (parametric_representation_context);
WHERE
  WR1: SIZEOF (QUERY (prc <* parametric_representation_context |
       NOT (SIZEOF (USEDIN (prc, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

parametric_representation_context: identifies the set of all instances of **parametric_representation_-context**.

Formal propositions:

WR1: For each instance of **parametric_representation_context**, there shall be a reference to the **parametric_representation_context** instance from an attribute of another entity.

5.2.5.22 dependent_instantiable_person_and_organization_role

The **dependent_instantiable_person_and_organization_role** rule specifies that all instances of **person_-and_organization_role** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_person_and_organization_role FOR
 (person_and_organization_role);
WHERE
  WR1: SIZEOF (QUERY (paor <* person_and_organization_role |
       NOT (SIZEOF (USEDIN (paor, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

person_and_organization_role: identifies the set of all instances of **person_and_organization_role**.

Formal propositions:

WR1: For each instance of **person_and_organization_role**, there shall be a reference to the **person_and_-organization_role** instance from an attribute of another entity.

5.2.5.23 dependent_instantiable_representation_item

The **dependent_instantiable_representation_item** rule specifies that all instances of **representation_item** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_representation_item FOR
 (representation_item);
WHERE
  WR1: SIZEOF (QUERY (ri <* representation_item |
       NOT (SIZEOF (USEDIN (ri, ''))  >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

representation_item: identifies the set of all instances of **representation_item**.

Formal propositions:

WR1: For each instance of **representation_item**, there shall be a reference to the **representation_item** instance from an attribute of another entity.

5.2.5.24 dependent_instantiable_security_classification_level

The **dependent_instantiable_security_classification_level** rule specifies that all instances of **security_-classification_level** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_security_classification_level FOR
 (security_classification_level);
WHERE
  WR1: SIZEOF (QUERY (scl <* security_classification_level |
       NOT (SIZEOF (USEDIN (scl, ''))  >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

security_classification_level: identifies the set of all instances of **security_classification_level**.

Formal propositions:

WR1: For each instance of **security_classification_level**, there shall be a reference to the **security_-classification_level** instance from an attribute of another entity.

5.2.5.25 dependent_instantiable_shape_representation

The **dependent_instantiable_shape_representation** rule specifies that all instances of **shape_-representation** are dependent on the usage to define another entity.

EXPRESS specification:

```
*)
RULE dependent_instantiable_shape_representation FOR
 (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (sr <* shape_representation |
       NOT (SIZEOF (USEDIN (sr, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

shape_representation: identifies the set of all instances of **shape_representation**.

Formal propositions:

WR1: For each instance of **shape_representation**, there shall be a reference to the **shape_representation** instance from an attribute of another entity.

5.2.5.26 global_unit_assignment

The **global_unit_assignment** rule specifies the units that shall be defined for a **global_unit_assigned_-context**. The rule states that every **global_unit_assigned_context** shall have at least two elements in its set of **unit**s, and that every **global_unit_assigned_context** shall contain units of length and plane angle.

EXPRESS specification:

```
*)
RULE global_unit_assignment FOR
 (global_unit_assigned_context);
WHERE
  WR1: SIZEOF (QUERY (guac <* global_unit_assigned_context |
       NOT (SIZEOF (guac.units) >= 2))) = 0;
  WR2: SIZEOF (QUERY (guac <* global_unit_assigned_context |
        NOT ((SIZEOF (QUERY (u <* guac.units |
              'MEASURE_SCHEMA.LENGTH_UNIT'
                IN TYPEOF (u))) = 1)
             AND
             (SIZEOF (QUERY (u <* guac.units |
              'MEASURE_SCHEMA.PLANE_ANGLE_UNIT'
                IN TYPEOF (u))) = 1)
        )) = 0;
END_RULE;
(*
```

Argument definitions:

global_unit_assigned_context: identifies the set of all instances of **global_unit_assigned_context** entities.

Formal propositions:

WR1: For each instance of **global_unit_assigned_context**, the set of **units** shall contain at least 2 elements.

WR2: For each instance of **global_unit_assigned_context**, the set of **units** shall contain one **length_unit** and one **plane_angle_unit**.

5.2.5.27 pdu_requires_security_classification

The **pdu_requires_security_classification** rule specifies that every **product_definition_usage** shall be referenced by exactly one **applied_security_classification_assignment**. This rule enforces the requirement for every **product_definition** that is the **related_product_definition** in a **product_definition_usage** to have a security classification within the context of the **product_definition_usage**.

EXPRESS specification:

```
*)
RULE pdu_requires_security_classification FOR
 (product_definition_usage,
  applied_security_classification_assignment);
WHERE
  WR1: SIZEOF (QUERY (e <* product_definition_usage |
     NOT (SIZEOF (QUERY (s <* applied_security_classification_assignment |
     e IN s.items)) = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

product_definition_usage: identifies the set of all instances of **product_definition_usage** entities to be constrained.

applied_security_classification_assignment: identifies the set of all instances of **applied_security_-classification_assignment** entities to be constrained.

Formal propositions:

WR1: For each instance of **product_definition_usage** there shall be exactly one instance of **applied_-security_classification_assignment** that contains the instance of **product_definition_usage** in its set of **items**.

5.2.5.28 ply_reference

The ply_reference rules states that **product**s associated with a **ply_laminate_sequence_definition** shall have a **product_related_product_category.name** of "ply".

EXPRESS Specification:

```
*)
RULE ply_reference FOR
 (ply_laminate_sequence_definition,
  next_assembly_usage_occurrence,
  product_related_product_category);
LOCAL
  i,j,k :  INTEGER;
  dkuhr :  LOGICAL;
  nnauo :  INTEGER;
```

```
  nprpc :  INTEGER;
  rp    :  product;
END_LOCAL;

  dkuhr := TRUE;
  REPEAT i:= LOINDEX (ply_laminate_sequence_definition) TO
             HIINDEX (ply_laminate_sequence_definition);

    nnauo := 0;
    REPEAT j:= LOINDEX (next_assembly_usage_occurrence) TO
               HIINDEX (next_assembly_usage_occurrence);

      IF (ply_laminate_sequence_definition[i] =
        next_assembly_usage_occurrence[j].relating_product_definition) THEN
        rp := next_assembly_usage_occurrence[j].related_product_definition.
              formation.of_product;
        nprpc := 0;
        REPEAT k:= LOINDEX (product_related_product_category) TO
                   HIINDEX (product_related_product_category);

          IF ((product_related_product_category[k].name = 'ply') AND
              (rp IN product_related_product_category[k].products)) THEN
              nprpc := nprpc + 1;
          END_IF;

        END_REPEAT;

        IF (nprpc = 1) THEN
          nnauo := nnauo + 1;
        ELSE
          dkuhr := FALSE;
          ESCAPE;
        END_IF;

      END_IF;

    END_REPEAT;

    IF (dkuhr = FALSE) THEN
      ESCAPE;
    END_IF;
    IF(nnauo = 0) THEN
      dkuhr := FALSE;
      ESCAPE;
    END_IF;

  END_REPEAT;


WHERE
  WR1:  dkuhr;
END_RULE;
(*
```

Argument definitions:

ply_laminate_sequence_definition: identifies the set of all instances of **ply_laminate_sequence_-definition** entities.

next_assembly_usage_occurrence: identifies the set of all instances of **next_assembly_usage_occurrence** entities.

product_related_product_category: identifies the set of all instances of **product_related_product_-category** entities.

Formal propositions:

WR1: **Ply_laminate_sequence_definition**s shall be related through at least one **next_assembly_-usage_occurrence** relationship to **product_definition**s which are associated with **product**s that are in the **product_related_product_category** with the **name** atttribute of "ply".

5.2.5.29 ply_stock_material_select

The **ply_stock_material_select** requires that any "ply" **product** be made from a "isotropic material", "filament assembly", or "discontinuous fiber assembly" material product. The make from is established by a **make_from_usage_option**. The relationship is between two **product_definition**s to **product_-definition_formation** to **product**s that are in the **product_related_product_category**.

EXPRESS specification:

```
*)
RULE ply_stock_material_select FOR
 (product_related_product_category,
  make_from_usage_option);
LOCAL
  i,j,k,kp :  INTEGER;
  dkuhr    :  LOGICAL;
  nmfuo    :  INTEGER;
  nprpc    :  INTEGER;
  rp       :  product;
END_LOCAL;

dkuhr := TRUE;
REPEAT kp:= LOINDEX (product_related_product_category) TO
            HIINDEX (product_related_product_category);


 IF (product_related_product_category[kp].name = 'ply') THEN
  REPEAT i:= LOINDEX (product_related_product_category[kp].products) TO
             HIINDEX (product_related_product_category[kp].products);

    nmfuo := 0;
    REPEAT j:= LOINDEX (make_from_usage_option) TO
               HIINDEX (make_from_usage_option);

      IF (product_related_product_category[kp].products[i] =
        make_from_usage_option[j].relating_product_definition.
    formation.of_product) THEN

        rp := make_from_usage_option[j].related_product_definition.
              formation.of_product;
        REPEAT k:= LOINDEX (product_related_product_category) TO
                   HIINDEX (product_related_product_category);

          IF ((rp IN product_related_product_category[k].products) AND
              (product_related_product_category[k].name IN
               ['isotropic material', 'filament assembly',
```



```
                'discontinuous fiber assembly'])) THEN
              nprpc := nprpc + 1;
          END_IF;

        END_REPEAT;

        IF (nprpc = 1) THEN
          nmfuo := nmfuo + 1;
        ELSE
          dkuhr := FALSE;
          ESCAPE;
        END_IF;

      END_IF;

    END_REPEAT;

    IF (dkuhr = FALSE) THEN
      ESCAPE;
    END_IF;
    IF(nmfuo <> 1) THEN
      dkuhr := FALSE;
      ESCAPE;
    END_IF;

  END_REPEAT;
  END_IF;

END_REPEAT;

WHERE
  WR1: dkuhr;
END_RULE;
(*
```

Argument definitions:

product_related_product_category: identifies the set of all instances of **product_related_product_-category** entities.

make_from_usage_option: identifies the set of all instances of **make_from_usage_option** entities.

Formal propositions:

WR1: Any **product** that is in the **product_related_product_category** with the **name** of "ply" shall be associated with the **relating_product_definition** of a **make_from_usage_option** relationship whose **related_product_definition** is associated with a **product** that is in the **product_related_product_category** with the **name** of "isotropic material", "filament assembly", or "discontinuous fiber assembly".

5.2.5.30 product_concept_requires_configuration_item

The **product_concept_requires_configuration_item** rule specifies that every **product_concept** shall be referenced by at least one **configuration_item**. This rule enforces the requirement for every **product_-concept** to have at least one **configuration_item** in it.

EXPRESS specification:

```
*)
RULE product_concept_requires_configuration_item FOR
 (product_concept, configuration_item);
WHERE
  WR1: SIZEOF (QUERY (pc <* product_concept |
       NOT (SIZEOF (QUERY (ci <* configuration_item |
       pc :=: ci.item_concept)) >=1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

product_concept: identifies the set of all instances of **product_concept** entities.

configuration_item: identifies the set of all instances of **configuration_item** entities.

Formal propositions:

WR1: For each instance of **product_concept**, there shall be at least one instance of **configuration_item** that contains the instance of **product_concept** as the value of its **item_concept** attribute.

5.2.5.31 product_definition_requires_date_time

The **product_definition_requires_date_time** rule specifies that each instance of a **product_definition** shall be referenced by exactly one **applied_date_and_time_assignment** or **applied_date_assignment**. This rule enforces the requirement for every **product_definition** to have a date indicating the date on which the definition of the product was created.

NOTE The coordination of the different role values with the assignment of **date_time_assignment** to different entities is specified in the **applied_date_and_time_correlation** function. This function is invoked locally to **applied_date_-and_time_assignment**. The coordination of the different role values with the assignment of **date_assignment** to different entities is specified in the **applied_date_correlation** function. This function is invoked locally to **applied_-date_assignment**.

EXPRESS specification:

```
*)
RULE product_definition_requires_date_time FOR
 (product_definition, applied_date_and_time_assignment,
  applied_date_assignment);
WHERE
  WR1: SIZEOF (QUERY (pd <* product_definition |
       NOT ((SIZEOF (QUERY (adta <* applied_date_and_time_assignment |
                    pd IN adta.items)) +
             SIZEOF (QUERY (ada <* applied_date_assignment |
                    pd IN ada.items))) = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

product_definition: identifies the set of all instances of **product_definition** entities.

applied_date_and_time_assignment: identifies the set of all instances of **applied_date_and_time_-assignment** entities.

applied_date_assignment: identifies the set of all instances of **applied_date_assignment** entities.

Formal propositions:

WR1: For each instance of **product_definition**, exactly one instance of **applied_date_and_time_-assignment** or **applied_date_assignment** shall contain the instance of **product_definition** in its set of **items**.

5.2.5.32 product_definition_requires_person_organization

The **product_definition_requires_person_organization** rule specifies that every **product_definition** shall be referenced by exactly one **applied_person_and_organization_assignment** or **applied_organization_-assignment**. This rule specifies the need for every **product_definition** to have a creator who is responsible for the creation of the particular definition of the design or analysis.

NOTE The coordination of the different role values with the assignment of **person_organization_assignment** to different entities is specified in the **applied_person_and_organization_correlation** function. This function is invoked locally to **applied_person_and_organization_assignment**. The coordination of the different role values with the assignment of **organization_assignment** to different entities is specified in the **applied_organization_correlation** function. This function is invoked locally to **applied_organization_assignment**.

EXPRESS specification:

```
*)
RULE product_definition_requires_person_organization FOR
  (product_definition, applied_person_and_organization_assignment,
   applied_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (pd <* product_definition |
       NOT ((SIZEOF (QUERY (apoa <*
                     applied_person_and_organization_assignment |
                     pd IN apoa.items)) +
             SIZEOF (QUERY (aoa <* applied_organization_assignment |
                     pd IN aoa.items))) = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

product_definition: identifies the set of all instances of **product_definition** entities.

applied_person_and_organization_assignment: identifies the set of all instances of **applied_person_-and_organization_assignment** entities.

applied_organization_assignment: identifies the set of all instances of **applied_organization_assignment** entities.

Formal propositions:

WR1: For each instance of **product_definition**, exactly one instance of **applied_person_and_-organization_assignment** or **applied_organization_assignment** shall contain the instance of **product_-definition** in its set of **items**.

5.2.5.33 product_requires_person_organization

The **product_requires_person_organization** rule specifies that each instance of **product** shall be referenced by an instance of **applied_person_and_organization_assignment** or **applied_organization_-assignment**. This rule enforces the requirement for every product to have an owner.

NOTE The coordination of the different role values with the assignment of **person_organization_assignment** to different entities is specified in the **applied_person_and_organization_correlation** function. This function is invoked locally to **applied_person_and_organization_assignment**. The coordination of the different role values with the assignment of **organization_assignment** to different entities is specified in the **applied_organization_correlation** function. This function is invoked locally to **applied_organization_assignment**.

EXPRESS specification:

```
*)
RULE product_requires_person_organization FOR
 (product, applied_person_and_organization_assignment,
  applied_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (prod <* product |
        NOT ((SIZEOF (QUERY (apoa <*
                    applied_person_and_organization_assignment |
                    prod IN apoa.items)) +
             SIZEOF (QUERY (aoa <* applied_organization_assignment |
                    prod IN aoa.items))) = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

product: identifies the set of all instances of **product** entities.

applied_person_and_organization_assignment: identifies the set of all instances of **applied_person_-and_organization_assignment** entities.

applied_organization_assignment: identifies the set of all instances of **applied_organization_assignment** entities.

Formal propositions:

WR1: For each instance of **product**, exactly one instance of **applied_person_and_organization_-assignment** or **applied_organization_assignment** shall contain the instance of **product** in its set of **items**.

5.2.5.34 product_requires_product_category

The **product_requires_product_category** rule specifies the requirement that each **product** shall be referenced by exactly one **product_related_product_category** that defines whether the product is an "assembly", "inseparable assembly", "detail" or "customer furnished equipment".

NOTE The **product_requires_product_category** rule constrains the relationship between **product_related_-product_category** and **product**. It says that each **product** shall be referenced by a **product_related_product_-category** whose **name** attribute value is either "assembly", "inseparable assembly", "detail" or "customer furnished equipment". The rule does not preclude the **product** to be referenced by **product_related_product_category** entities which have **name** attributes that are different from those listed above. Valid values for the **name** attribute of the **product_related_product_category** entity are defined in the rule **restrict_product_category_value**.

EXPRESS specification:

```
*)
RULE product_requires_product_category FOR
 (product, product_related_product_category);
WHERE
  WR1: SIZEOF (QUERY (prod <* product |
       NOT (SIZEOF (QUERY (prpc <* product_related_product_category |
       (prod IN prpc.products) AND
       (prpc.name IN ['assembly', 'inseparable assembly', 'detail',
       'customer furnished equipment']))) = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

product: identifies the set of all instances of **product** entities.

product_related_product_category: identifies the set of all instances of **product_related_product_-category** entities.

Formal propositions:

WR1: For each instance of **product**, there shall be exactly one instance of **product_related_product_-category** that contains a **name** attribute that has a value of either "assembly", "inseparable assembly", "detail" or "custome furnished equipment".

5.2.5.35 product_requires_version

The **product_requires_version** rule specifies that each instance of **product** shall be referenced by at least one instance of **product_definition_formation**. This rule enforces the requirement for every product to have one or more versions.

EXPRESS specification:

```
*)
RULE product_requires_version FOR
 (product, product_definition_formation);
WHERE
  WR1: SIZEOF (QUERY (prod <* product |
```

```
       NOT (SIZEOF (QUERY (pdf <* product_definition_formation |
       prod :=: pdf.of_product)) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

product: identifies the set of all instances of **product** entities.

product_definition_formation: identifies the set of all instances of **product_definition_formation** entities.

Formal propositions:

WR1: For each instance of **product**, there shall be one or more instances of **product_definition_formation** that contains an **of_product** attribute value equal to that instance of **product**.

5.2.5.36 product_version_requires_person_organization

The **product_version_requires_person_organization** rule specifies that every **product_definition_-formation** shall be referenced by exactly one **applied_person_and_organization_assignment** or **applied_-organization_assignment** in the role of creator, and one or more **applied_person_and_organization_-assignment** or **applied_organization_assignment** in the role of either part supplier, analysis supplier, or design supplier. This rule specifies the need for every **product_definition_formation** to have a creator and supplier who are responsible for the creation or delivery of the particular version of the analysis or design. The meanings of creator, part supplier, analysis supplier, and design supplier are found in as part of the definition of the **restrict_person_organization_role** rule.

NOTE The coordination of the different role values with the assignment of **person_organization_assignment** to different entities is specified in the **applied_person_and_organization_correlation** function. This function is invoked locally to **applied_person_and_organization_assignment**. The coordination of the different role values with the assignment of **organization_assignment** to different entities is specified in the **applied_organization_correlation** function. This function is invoked locally to **applied_organization_assignment**.

EXPRESS specification:

```
*)
RULE product_version_requires_person_organization FOR
 (product_definition_formation,
  applied_person_and_organization_assignment,
  applied_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (pdf <* product_definition_formation |
       NOT ((SIZEOF (QUERY (apoa <*
                     applied_person_and_organization_assignment |
                     (pdf IN apoa.items) AND
                     (apoa.role.name = 'creator'))) +
             SIZEOF (QUERY (aoa <*
                     applied_organization_assignment |
                     (pdf IN aoa.items) AND
                     (aoa.role.name = 'creator')))) = 1 ))) = 0;
  WR2: SIZEOF (QUERY (pdf <* product_definition_formation |
       NOT ((SIZEOF (QUERY (apoa <*
                     applied_person_and_organization_assignment |
                     (pdf IN apoa.items) AND (apoa.role.name IN
```



```
                      ['analysis supplier', 'design supplier',
                       'part supplier']))) +
              SIZEOF (QUERY (aoa <*
                      applied_organization_assignment |
                      (pdf IN aoa.items) AND (aoa.role.name IN
                      ['analysis supplier', 'design supplier',
                       'part supplier']))))  >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

product_definition_formation: identifies the set of all instances of **product_definition_formation** entities.

applied_person_and_organization_assignment: identifies the set of all instances of **applied_person_-and_organization_assignment** entities.

applied_organization_assignment: identifies the set of all instances of **applied_organization_assignment** entities.

Formal propositions:

WR1: For each instance of **product_definition_formation**, there shall be exactly one instance of **applied_-person_and_organization_assignment** or **applied_organization_assignment** that contains the instance of **product_definition_formation** in its set of **items** and its **role** attribute references a **person_and_-organization_role** or **organization_role** that has a value of "creator" in its **name** attribute.

WR2: For each instance of **product_definition_formation**, there shall be at least one instance of **applied_-person_and_organization_assignment** or **applied_organization_assignment** that contains the instance of **product_definition_formation** in its set of **items** and its **role** attribute references a **person_and_-organization_role** or **organization_role** that has a value of either "analysis supplier", "design supplier", or "part supplier" in its **name** attribute.

5.2.5.37 product_version_requires_security_classification

The **product_version_requires_security_classification** rule specifies that each instance of **product_-definition_formation** shall be referenced by exactly one instance of **applied_security_classification_-assignment**. This rule enforces the requirement for every version of a design to have a security classification.

EXPRESS specification:

```
*)
RULE product_version_requires_security_classification FOR
 (product_definition_formation,
  applied_security_classification_assignment);
WHERE
  WR1: SIZEOF (QUERY (e <* product_definition_formation |
       NOT (SIZEOF (QUERY (s <*
            applied_security_classification_assignment |
            e IN s.items)) = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

product_definition_formation: identifies the set of all instances of **product_definition_formation** entities.

applied_security_classification_assignment: identifies the set of all instances of **applied_security_-classification_assignment** entities.

Formal propositions:

WR1: For each instance of **product_definition_formation**, there shall be exactly one instance of **applied_-security_classification_assignment** that contains the instance of **product_definition_formation** in its set of **items**.

5.2.5.38 restrict_action_request_status

The **restrict_action_request_status** rule specifies the permitted values for the status of a **versioned_-action_request**.

EXPRESS specification:

```
*)
RULE restrict_action_request_status FOR
 (action_request_status);
WHERE
  WR1: SIZEOF (QUERY (ars <* action_request_status |
       NOT (ars.status IN ['proposed', 'in work', 'issued', 'hold']))) = 0;
END_RULE;
(*
```

Argument definitions:

action_request_status: identifies the set of all instances of **action_request_status** entities.

Formal propositions:

WR1: For each instance of the **action_request_status** entity, the status attribute shall have a value of "proposed", "in work", "issued", or "hold".

Attribute value definitions:

proposed: the **versioned_action_request** has been completed and is awaiting review and authorization.

in work: the **versioned_action_request** is being developed for possible inclusion in the design or analysis.

issued: the **versioned_action_request** has been authorized for inclusion in the design or analysis.

hold: the **versioned_action_request** has been reviewed and not authorized to be included in the design or analysis.

5.2.5.39 restrict_approval_status

The **restrict_approval_status** rule specifies that the only values of **approval_status** permitted shall be "approved", "not yet approved", "disapproved" or "withdrawn".

EXPRESS specification:

```
*)
RULE restrict_approval_status FOR
 (approval_status);
WHERE
  WR1: SIZEOF (QUERY (ast <* approval_status |
       NOT (ast.name IN
       ['approved', 'not yet approved', 'disapproved', 'withdrawn']))) = 0;
END_RULE;
(*
```

Argument definitions:

approval: identifies the set of all instances of **approval** entities.

Formal propositions:

WR1: For each instance of **approval**, the value of the **status** attribute shall be either "approved", "not yet approved", "disapproved" or "withdrawn".

Attribute value definitions:

approved: specifies that the required authorizations have been obtained for a particular role of approval for a piece of product data.

not yet approved: specifies that the required authorizations are pending for a particular role of approval for a piece of product data.

disapproved: specifies that the required authorizations have been denied for a particular role of approval for a piece of product data.

withdrawn: specifies that the required authorizations have been revoked for a particular role of approval for a piece of product data.

5.2.5.40 restrict_date_time_role

The **restrict_date_time_role** rule specifies the permitted roles for **date_and_time** and **date** entities. This rule enforces the requirement for the roles of **date_and_time** and **date** entities to be "change date", "creation date", "request date", "release date", "start date", "earliest end date", "latest end date", "contract date", "certification date", "classification date", or "declassification date".

EXPRESS specification:

```
*)
RULE restrict_date_time_role FOR
 (date_time_role, date_role);
```



```
WHERE
  WR1: (SIZEOF (QUERY (dtr <* date_time_role |
        NOT (dtr.name IN ['change date', 'creation date',
       'request date', 'release date','start date',
       'earliest end date', 'latest end date', 'contract date',
       'sign off date', 'certification date',
       'classification date', 'declassification date']))) +
        SIZEOF (QUERY (dr <* date_role |
        NOT (dr.name IN ['change date', 'creation date',
       'request date', 'release date','start date',
       'earliest end date', 'latest end date', 'contract date',
       'sign off date', 'certification date',
       'classification date', 'declassification date'])))) = 0;
END_RULE;
(*
```

Argument definitions:

date_time_role: identifies the set of all instances of **date_time_role** entities.

date_role: identifies the set of all instances of **date_role** entities.

Formal propositions:

WR1: For each instance of **date_time_role** and **date_role**, the name attribute shall have the value of "change date", "creation date", "request date", "release date", "start date", "earliest end date", "latest end date", "contract date", sign off date", "certification date", "classification date", or "declassification date".

Attribute value definitions:

change date: identifies the date and time when work is begun for a change to an existing design or analysis.

creation date: identifies the date and time when either a definition or a version of a product comes into existence.

request date: identifies the date and time given when work is requested for a design or analysis.

release date: identifies the date and time when a design is initially released or a change is incorporated into a design or analysis.

start date: identifies the date and time when work is begun for a new design or analysis, or the start date for retention of product data.

earliest end date: identifies the date and time when product data may be deleted.

latest end date: identifies the date and time when product data shall be deleted.

contract date: identifies the date and time when a contract for a design or analysis goes into effect.

sign off date: identifies the date and time when an authorization is given to an approval.

certification date: identifies the date and time when a certification goes into effect.

classification date: identifies the date and time when a security classification goes into effect.

declassification date: identifies the date and time when a security classification goes out of effect.

5.2.5.41 restrict_person_organization_role

The **restrict_person_organization_role** rule specifies the permitted roles for **person_and_organization** and **organization** entities. This rule enforces the requirement for the roles of **person_and_organization** and **organization** entities to be "request recipient", "initiator", "part supplier", "design supplier", "analysis supplier", "configuration manager", "contractor", "classification officer", "creator", "design owner", or "analysis owner".

EXPRESS specification:

```
*)
RULE restrict_person_organization_role FOR
 (person_and_organization_role, organization_role);
WHERE
  WR1: (SIZEOF (QUERY (porl <* person_and_organization_role |
        NOT (porl.name IN ['request recipient', 'initiator',
       'part supplier','design supplier', 'analysis supplier',
       'creator', 'design owner', 'analysis owner', 'contractor',
       'configuration manager', 'classification officer']))) +
        SIZEOF (QUERY (orl <* organization_role |
        NOT (orl.name IN ['request recipient', 'initiator',
       'part supplier','design supplier', 'analysis supplier',
       'creator', 'design owner', 'analysis owner', 'contractor',
       'configuration manager', 'classification officer'])))) = 0;
END_RULE;
(*
```

Argument definitions:

person_organization_role: identifies the set of all instances of **person_organization_role** entities.

organization_role: identifies the set of all instances of **organization_role** entities.

Formal propositions:

WR1: For each instance of **person_organization_role** and **organization_role**, the name attribute shall have a value of "request recipient", "initiator", "part supplier", "design supplier", "analysis supplier", "configuration manager", "contractor", "classification officer", "creator", "design owner", or "analysis owner".

Attribute value definitions:

request recipient: identifies a person within an organization who is responsible for receiving a **change_-request** or **start_request** and taking action based on the request.

initiator: identifies a person within an organization who is responsible for creating the **change_request** or **start_request**.

part supplier: identifies a person within an organization who is responsible for supplying a part.

design supplier: identifies a person within an organization who is responsible for supplying the design of a part

analysis supplier: identifies a person within an organization who is responsible for supplying the analysis of a part.

configuration manager: identifies a person within an organization who is responsible for assigning the configuration information about a design or analysis.

contractor: identifies a person within an organization who is responsible for the information pertaining to a contract that applies to a design or analysis.

classification officer: identifies a person within an organization who is responsible for the classification and declassification of parts.

creator: identifies a person within an organization who is responsible for the creation of a particular product definition or version.

design owner: identifies a person within an organization who is responsible for the design of all aspects of a **product**.

analysis owner: identifies a person within an organization who is responsible for the analysis of all aspects of a **product**.

5.2.5.42 restrict_product_category_value

The **restrict_product_category_value** rule specifies the set of values that a **product_category** that is related to a **product** may contain.

<u>EXPRESS specification</u>:

```
*)
RULE restrict_product_category_value FOR
 (product_related_product_category);
WHERE
  WR1: SIZEOF (QUERY (prpc <*
       product_related_product_category |
       NOT (prpc.name IN ['assembly', 'detail',
       'customer furnished equipment', 'inseparable assembly',
       'standard part', 'linear static analysis',
       'linear modes and frequencies analysis',
       'cast', 'coined', 'drawn', 'extruded', 'forged',
       'formed', 'machined', 'molded', 'rolled', 'sheared',
       'anisotropic material', 'composite assembly',
       'discontinuous fiber assembly', 'filament assembly',
       'filament laminate', 'isotropic material', 'ply',
       'ply laminate', 'ply piece', 'processed core',
       'stock core']))) = 0;
END_RULE;
(*
```


Argument definitions:

product_related_product_category: identifies the set of all instances of **product_related_product_-category** entities.

Formal propositions:

WR1: The **name** attribute of a **product_related_product_category** shall be either "assembly", "detail", "customer furnished equipment", "inseparable assembly", "standard part", "linear static analysis", "linear modes and frequencies analysis", "cast", "coined", "drawn", "extruded", "forged", "formed", "machined", "molded", "rolled", "sheared", "anisotropic material", "composite assembly", "discontinuous fiber assembly", "filament assembly", "filament laminate", "isotropic material", "ply", "ply laminate", "ply piece", "processed core", or "stock core".

Attribute value restriction definitions:

assembly: identifies a part that consists of a collection of other parts which are put together to satisfy a particular function.

detail: identifies a part that exists at the lowest level of the bill of materials structure.

customer furnished equipment: identifies a part that has been furnished to the design agency by the customer.

inseparable assembly: identifies a part that after being put together cannot be dis-assembled without causing physical harm to at least one of the components of the assembly.

standard part: identifies a part that has a design specified externally to the bill-of-material in which an assembly is defined.

linear static analysis: identifies an analysis studying linear structural responses of a part under static loading.

linear modes and frequencies analysis: identifies a linear dynamic analysis concerning natural frequencies and corresponding mode shapes for a structural part.

cast: identifies a class of parts that are intended to be produced by a manufacturing process in which a liquid form of material is poured into a mold and hardened.

coined: identifies a class of parts that are intended to be produced by a manufacturing process in which a liquid form of material is forced into well-confined dies under high pressure to fill the shape of the die.

drawn: identifies a class of parts that are intended to be produced by a manufacturing process in which a metal is pulled in order to remove material from a raw stock and form an indenture in the material of a defined shape.

extruded: identifies a class of parts that are intended to be produced by a manufacturing process in which material is squeezed by a compressing operation that changes the shape of the raw stock.

forged: identifies a class of parts that are intended to be produced by a manufacturing process in which metal materials are worked into useful shapes under the influence of external force at high temperature.

formed: identifies a class of parts that are intended to be produced by a manufacturing process in which metal materials subjected to stresses which change their shape.

machined: identifies a class of parts that are intended to be produced by a manufacturing process which employs a piece of automated equipment.

molded: identifies a class of parts that are intended to be produced by a manufacturing process in which material is heated past its melting point and poured into a mold to form its shape.

rolled: identifies a class of parts that are intended to be produced by a manufacturing process in which material is passed between two rollers in order to reduce it to the thickness equal to the space between the two rollers.

sheared: identifies a class of parts that are intended to be produced by a manufacturing process in which material is cut to alter its shape.

anisotropic material: identifies a material whose properties vary depending on the orientation of the material.

composite assembly: identifies a physical or conceived assembly that is made of multiple materials that are bonded together. The composite layup assemblies may be used as a composite part. A composite_-assembly can be made from a combination of other composite_assemblies.

discontinuous fiber assembly: identifies a stock material that is a collection of fibers suspended in a homogenous material (matrix). Usually the length of the fibers are relatively the same within a detail. The orientation of the fibers is usually random.

filament assembly: identifies a stock material which is a collection of yarns or tows combined together in some manner and frequently embedded in a matrix of homogenous material. The filament assembly is usually measured by weight and sometimes length to describe the amount needed for a filament laminate.

filament laminate: identifies a bonded material of two or more bundles of filament construction. The filament laminate defines a volume to fill with filament assemblies.

isotropic material: identifies stock material that is made up of a single uniform internal structure.

ply: identifies one of the layers of constant thickness material that makes up a composite part. A ply is a contiguous structure with a defined boundary that is made of one or more ply pieces.

ply laminate: identifies two or more plies that mate with each other. The plies have unique orientation and shape within the ply laminate.

ply piece: identifies a single portion of a ply that may be combined with other ply pieces on the same layer to make a ply. A ply_piece has the same material and orientation direction as defined for the ply of which it is a portion.

processed core: identifies the central component of a sandwich construction to which the faces or skins are attached. The primary structural purpose of processed core is the transfer of shear loads between the faces or skins. Processed core is stock core that has been machined, formed, stabilized, or bonded together. A core may have a potting compound or adhesive applied to it to make it more rigid or provide solid attachment points.

stock core: identifies the material that is purchased with the intent of processing it so that it can be placed within a laminate and play a role in enhancing the stiffness of the laminate.

5.2.5.43 restrict_product_definition_context

The **restrict_product_definition_context** rule specifies the permitted values for the **life_cycle_stage** of a **product_definition_context**.

EXPRESS specification:

```
*)
RULE restrict_product_definition_context FOR
 (product_definition_context);
WHERE
  WR1: SIZEOF (QUERY (pdc <* product_definition_context |
       NOT (pdc.life_cycle_stage IN ['analysis', 'design']))) = 0;
END_RULE;
(*
```

Argument definitions:

product_definition_context: identifies the set of all instances of **product_definition_context** entities.

Formal propositions:

WR1: For each instance of the **product_definition_context** entity, the **life_cycle_stage** attribute shall have a value of "design" or "analysis".

Attribute value definitions:

design: One of the stages of the product life cycle in which the exchange of information is given.

analysis: One of the stages of the product life cycle in which the exchange of information is given.

NOTE The **application_context_element** attribute **name** shall be used to modify the **product_definition_context** in order to specify the exact context within design or analysis within which the **product_definition** is applicable.

5.2.5.44 restrict_security_classification_level

The **restrict_security_classification_level** rule specifies the permitted levels of security. This rule enforces the requirement for the levels of security to be "classified", "unclassified", "proprietary", "confidential", "secret", or "top secret".

EXPRESS specification:

```
*)
RULE restrict_security_classification_level FOR
 (security_classification_level);
WHERE
  WR1: SIZEOF (QUERY (scl <* security_classification_level |
       NOT (scl.name IN ['unclassified', 'classified', 'proprietary',
       'confidential', 'secret', 'top secret']))) = 0;
END_RULE;
(*
```

Argument definitions:

security_classification_level: identifies the set of all instances of **security_classification_level** entities.

Formal propositions:

WR1: For each instance of **security_classification_level**, the **level** attribute shall contain a value of "classified", "unclassified", "proprietary", "confidential", "secret" or "top secret".

Attribute value definitions:

unclassified: identifies the classification level for which no security is necessary.

classified: identifies the classification level for which security is necessary, but the classification details are not given.

proprietary: identifies the classification level for which the disclosure of information about the part, the design of the part, or the analysis of the part would risk an organizations market or competitive advantage.

confidential: identifies the classification level for which the disclosure of information about the part, the design of the part, or the analysis of the part would cause damage to national or organizational security.

secret: identifies the classification level for which the disclosure of information about the part, the design of the part, or the analysis of the part would cause serious damage to national or organizational security.

top secret: identifies the classification level for which the disclosure of information about the part, the design of the part, or the analysis of the part would cause exceptionally grave damage to national or organizational security.

5.2.5.45 stock_material_reference

The **stock_material_reference** rule states that **product**s associated with a **percentage_ply_definition** shall have components that have a **product_related_product_category.name** of "isotropic material", "anisotropic material", "filament assembly", "stock core", or "discontinuous fiber assembly".

EXPRESS specification:

```
*)
RULE stock_material_reference FOR
 (percentage_ply_definition,
```

```
  make_from_usage_option,
  product_related_product_category);
LOCAL
  i,j,k :  INTEGER;
  dkuhr :  LOGICAL;
  nmfuo :  INTEGER;
  nprpc :  INTEGER;
  rp     :  product;
END_LOCAL;

  dkuhr := TRUE;
  REPEAT i:= LOINDEX (percentage_ply_definition) TO
             HIINDEX (percentage_ply_definition);

    nmfuo := 0;
    REPEAT j:= LOINDEX (make_from_usage_option) TO
               HIINDEX (make_from_usage_option);

      IF (percentage_ply_definition[i] =
        make_from_usage_option[j].relating_product_definition) THEN

        rp := make_from_usage_option[j].related_product_definition.
              formation.of_product;
        nprpc := 0;
        REPEAT k:= LOINDEX (product_related_product_category) TO
                   HIINDEX (product_related_product_category);

          IF ((rp IN product_related_product_category[k].products) AND
              (product_related_product_category[k].name IN
              ['anisotropic material', 'isotropic material', 'stock core',
               'filament assembly', 'discontinuous fiber assembly'])) THEN
              nprpc := nprpc + 1;
          END_IF;

        END_REPEAT;

        IF (nprpc = 1) THEN
          nmfuo := nmfuo + 1;
        ELSE
          dkuhr := FALSE;
          ESCAPE;
        END_IF;

      END_IF;

    END_REPEAT;

    IF (dkuhr = FALSE) THEN
      ESCAPE;
    END_IF;
    IF(nmfuo = 0) THEN
      dkuhr := FALSE;
      ESCAPE;
    END_IF;

  END_REPEAT;

WHERE
  WR1:  dkuhr;
END_RULE;
(*
```

Argument definitions:

percentage_ply_definition: identifies the set of all instances of **percentage_ply_definition** entities.

make_from_usage_option: identifies the set of all instances of **make_from_usage_option** entities.

product_related_product_category: identifies the set of all instances of **product_related_product_-category** entities.

Formal propositions:

WR1: **Percentage_ply_definition**s shall be related through at least one **make_from_usage_option** relationship to **product_definition**s which are associated with **product**s that are in the **product_related_-product_category** with the **name** atttribute of "isotropic material", "anisotropic material", "filament assembly", "discontinuous fiber assembly", or "stock core".

5.2.5.46 subtype_mandatory_action

The **subtype_mandatory_action** rule specifies that all **action** entities shall be a **directed_action** entity.

EXPRESS specification:

```
*)
RULE subtype_mandatory_action FOR
 (action);
WHERE
  WR1: SIZEOF (QUERY (act <* action |
       NOT ('ACTION_SCHEMA.DIRECTED_ACTION'
            IN TYPEOF (act)))) = 0;
END_RULE;
(*
```

Argument definitions:

action: identifies the set of all instances of **action** entities to be constrained.

Formal propositions:

WR1: Each instance of **action** shall be a **directed_action**.

5.2.5.47 subtype_mandatory_shape_representation

The **subtype_mandatory_shape_representation** rule requires all **shape_representation** entities to be either **geometrically_bounded_wireframe_shape_representation**, **geometrically_bounded_surface_-shape_representation**, **edge_based_wireframe_shape_representation**, **point_representation**, **shell_-based_wireframe_shape_representation**, **manifold_surface_shape_representation**, **faceted_brep_-shape_representation**, **advanced_brep_shape_representation,** **shape_dimension_representation**, **composite_sheet_representation;** or contain only **placement** or **mapped_item** entities in their respective sets of **items**; or be the representation of a **shape_aspect** or a relationship between two **shape_aspect**s. The rule defines the constraint that establishes the different types of representations allowed for shapes in this part of ISO 10303.

EXPRESS specification:

```
*)
RULE subtype_mandatory_shape_representation FOR
 (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (sr <* shape_representation |
       NOT
       ((SIZEOF (['AIC_GEOMETRICALLY_BOUNDED_SURFACE.' +
        'GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION',
        'AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.' +
        'GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_REPRESENTATION',
        'AIC_EDGE_BASED_WIREFRAME.' +
        'EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION',
        'AIC_SHELL_BASED_WIREFRAME.' +
        'SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION',
        'AIC_MANIFOLD_SURFACE.' +
        'MANIFOLD_SURFACE_SHAPE_REPRESENTATION',
        'STRUCTURAL_ANALYSIS_DESIGN.POINT_PATH',
        'STRUCTURAL_ANALYSIS_DESIGN.POINT_AND_VECTOR',
        'STRUCTURAL_ANALYSIS_DESIGN.POINT_REPRESENTATION',
        'AIC_FACETED_BREP.FACETED_BREP_SHAPE_REPRESENTATION',
        'AIC_ADVANCED_BREP.ADVANCED_BREP_SHAPE_REPRESENTATION',
        'SHAPE_DIMENSION_SCHEMA.SHAPE_DIMENSION_REPRESENTATION',
        'STRUCTURAL_ANALYSIS_DESIGN.BEVELED_SHEET_REPRESENTATION',
        'STRUCTURAL_ANALYSIS_DESIGN.COMPOSITE_SHEET_REPRESENTATION'] *
         TYPEOF (sr)) > 0)
        OR
        (SIZEOF (QUERY (it <* sr\representation.items |
         NOT (SIZEOF (['GEOMETRY_SCHEMA.PLACEMENT',
                       'REPRESENTATION_SCHEMA.MAPPED_ITEM'] *
                      TYPEOF (it)) = 1 ))) = 0)
        OR
        (SIZEOF (QUERY (sdr <* QUERY (pdr <* USEDIN (sr,
        'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.' +
        'PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTATION') |
        'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.' +
        'SHAPE_DEFINITION_REPRESENTATION' IN TYPEOF (pdr)) |
         NOT (SIZEOF (['PRODUCT_PROPERTY_DEFINITION_SCHEMA.SHAPE_ASPECT',
        'PRODUCT_PROPERTY_DEFINITION_SCHEMA.SHAPE_ASPECT_RELATIONSHIP'] *
         TYPEOF (sdr.definition.definition)) = 1 ))) = 0)
        ))) = 0;
END_RULE;
(*
```

Argument definitions:

shape_representation: identifies the set of all instances of **shape_representation** entities to be constrained.

Formal propositions:

WR1: Each instance of **shape_representation** shall be either a **geometrically_bounded_wireframe_shape_representation**, **geometrically_bounded_surface_shape_representation**, **edge_based_wireframe_shape_representation**, **point_representation**, **shell_based_wireframe_shape_representation**, **manifold_surface_shape_representation**, **faceted_brep_shape_representation**, **advanced_brep_shape_representation, shape_dimension_representation**, **composite_sheet_representation** or **beveled_sheet_representation**; or contain only **placement** or **mapped_item** objects in their respective sets of **items**; or be the representation of a **shape_aspect** or a relationship between two **shape_aspect**s.

5.2.5.48 thickness_laminate_table_component_select

The **thickness_laminate_table_component_select** rule states that **product**s associated with a **thickness_-laminate_definition** shall have a product_related_product_category.name of "ply", "filament laminate", or "processed core".

EXPRESS specification:

```
*)
RULE thickness_laminate_table_component_select FOR
 (thickness_laminate_definition,
  next_assembly_usage_occurrence,
  product_related_product_category);
LOCAL
  i,j,k :  INTEGER;
  dkuhr :  LOGICAL;
  nnauo :  INTEGER;
  nprpc :  INTEGER;
  rp    :  product;
END_LOCAL;
  dkuhr := TRUE;
  REPEAT i:= LOINDEX (thickness_laminate_definition) TO
            HIINDEX (thickness_laminate_definition);
    nnauo := 0;

    REPEAT j:= LOINDEX (next_assembly_usage_occurrence) TO
              HIINDEX (next_assembly_usage_occurrence);

      IF (thickness_laminate_definition[i] =
        next_assembly_usage_occurrence[j].relating_product_definition) THEN
        rp := next_assembly_usage_occurrence[j].related_product_definition.
              formation.of_product;
        nprpc := 0;
        REPEAT k:= LOINDEX (product_related_product_category) TO
                  HIINDEX (product_related_product_category);

          IF ((rp IN product_related_product_category[k].products) AND
              (product_related_product_category[k].name IN
               ['ply', 'filament laminate', 'processed core'])) THEN
              nprpc := nprpc + 1;
          END_IF;
        END_REPEAT;

        IF (nprpc = 1) THEN
          nnauo := nnauo + 1;
        ELSE
          dkuhr := FALSE;
          ESCAPE;
        END_IF;
      END_IF;

    END_REPEAT;
    IF (dkuhr = FALSE) THEN
      ESCAPE;
    END_IF;
    IF (nnauo <> 1) THEN
      dkuhr := FALSE;
      ESCAPE;
    END_IF;
```



```
  END_REPEAT;
WHERE
  WR1:  dkuhr;
END_RULE;
(*
```

Argument definitions:

thickness_laminate_definition: identifies the set of all instances of **thickness_laminate_definition** entities.

next_assembly_usage_occurrence: identifies the set of all instances of **next_assembly_usage_occurrence** entities.

product_related_product_category: identifies the set of all instances of **product_related_product_-category** entities.

Formal propositions:

WR1: Each **thickness_laminate_definition** shall be related through one **next_assembly_usage_-occurrence** relationship to a **product_definition** which has an associated **product** that is in the **product_-related_product_category** with the **name** atttribute of "ply", "filament laminate", or "processed core".

5.2.5.49 unique_version_change_order_rule

The **unique_version_change_order_rule** calls a function that returns true if an **action_directive** updates **product_definition_formation**s that are versions different products. This rule specifies that a single **action_directive** shall not change more than one version of a single **product**, but may change more than one **product_definition_formation** if each **product_definition_formation** references a different **product**.

EXPRESS specification:

```
*)
RULE unique_version_change_order_rule FOR
 (action_directive);
WHERE
  WR1: SIZEOF (QUERY (ad <* action_directive |
       NOT (unique_version_change_order (ad)))) = 0;
END_RULE;
(*
```

Argument definitions:

action_directive: identifies the set of all instances of **action_directive** entities.

Formal propositions:

WR1: For each instance of **action_directive**, the **unique_version_change_order** function shall return a value of true.

5.2.5.50 versioned_action_request_requires_status

The **versioned_action_request_requires_status** rule specifies that each instance of **versioned_action_-request** shall have exactly one status. The status of a **versioned_action_request** is defined by the **action_-request_status** entity.

EXPRESS specification:

```
*)
RULE versioned_action_request_requires_status FOR
 (versioned_action_request, action_request_status);
WHERE
  WR1: SIZEOF (QUERY (ar <* versioned_action_request |
       NOT (SIZEOF (QUERY (ars <* action_request_status |
       ar :=: ars.assigned_request)) = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

versioned_action_request: identifies the set of all instances of **versioned_action_request**.

action_request_status: identifies the set of all instances of **action_request_status**.

Formal propositions:

WR1: For each instance of **versioned_action_request** there shall be exactly one instance of **action_-request_status** that contains an **assigned_request** attribute value equal to that instance of **versioned_-action_request**.

5.2.6 AP209 function definitions

5.2.6.1 applied_date_correlation

The **applied_date_correlation** boolean function returns true if the **name** attribute value of the **date_role** entity is coordinated with the type of entity selected in the **items** of the **applied_date_assignment** entity.

EXAMPLE If the role of a **date_time** is "creation date" then all of the **items** in the **applied_date_assignment** shall be **product_definition** entities.

EXPRESS specification:

```
*)
FUNCTION applied_date_correlation
  (e : applied_date_assignment) : BOOLEAN;
  LOCAL
    dt_role : STRING;
  END_LOCAL;
    dt_role := e\date_assignment.role.name;
  CASE dt_role OF
    'change date'           : IF SIZEOF (e.items) <>
                                 SIZEOF (QUERY (x <* e.items |
                                 'ACTION_SCHEMA.ACTION_DIRECTIVE'
                                 IN TYPEOF (x)))
                                 THEN RETURN(FALSE);
```



```
                           END_IF;
'creation date'          : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'PRODUCT_DEFINITION_SCHEMA.' +
                              'PRODUCT_DEFINITION'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
'earliest end date'      : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'STRUCTURAL_ANALYSIS_DESIGN.RETENTION'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
'latest end date'        : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'STRUCTURAL_ANALYSIS_DESIGN.RETENTION'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
'request date'           : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'ACTION_SCHEMA.VERSIONED_ACTION_REQUEST'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
'release date'           : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'ACTION_SCHEMA.ACTION_DIRECTIVE'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
'start date'             : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              SIZEOF (['ACTION_SCHEMA.ACTION_DIRECTIVE',
                              'STRUCTURAL_ANALYSIS_DESIGN.RETENTION'] *
                              TYPEOF (x)) = 1 ))
                              THEN RETURN(FALSE);
                           END_IF;
'sign off date'          : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'APPROVAL_SCHEMA.' +
                              'APPROVAL_PERSON_ORGANIZATION'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
'contract date'          : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'CONTRACT_SCHEMA.CONTRACT'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
'certification date'     : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'CERTIFICATION_SCHEMA.CERTIFICATION'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
'classification date'    : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'SECURITY_CLASSIFICATION_SCHEMA.' +
                              'SECURITY_CLASSIFICATION'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
```

```
                                END_IF;
    'declassification date' : IF SIZEOF (e.items) <>
                                   SIZEOF (QUERY (x <* e.items |
                                   'SECURITY_CLASSIFICATION_SCHEMA.' +
                                   'SECURITY_CLASSIFICATION'
                                   IN TYPEOF (x)))
                                   THEN RETURN(FALSE);
                                END_IF;
    OTHERWISE : RETURN(TRUE);
  END_CASE;
  RETURN (TRUE);
END_FUNCTION;
(*
```

Argument definitions:

e: identifies the input **applied_date_assignment** to be checked.

5.2.6.2 applied_date_time_correlation

The **applied_date_time_correlation** boolean function returns true if the **name** attribute value of the **date_-time_role** entity is coordinated with the type of entity selected in the **items** of the **applied_date_and_-time_assignment** entity.

EXAMPLE If the role of a **date_and_time** is "creation date" then all of the **items** in the **applied_date_and_time_-assignment** shall be **product_definition** entities.

EXPRESS specification:

```
*)
FUNCTION applied_date_time_correlation
  (e : applied_date_and_time_assignment) : BOOLEAN;
  LOCAL
    dt_role : STRING;
  END_LOCAL;
    dt_role := e\date_and_time_assignment.role.name;
  CASE dt_role OF
    'change date'            : IF SIZEOF (e.items) <>
                                   SIZEOF (QUERY (x <* e.items |
                                   'ACTION_SCHEMA.ACTION_DIRECTIVE'
                                   IN TYPEOF (x)))
                                   THEN RETURN(FALSE);
                                END_IF;
    'creation date'          : IF SIZEOF (e.items) <>
                                   SIZEOF (QUERY (x <* e.items |
                                   'PRODUCT_DEFINITION_SCHEMA.' +
                                   'PRODUCT_DEFINITION'
                                   IN TYPEOF (x)))
                                   THEN RETURN(FALSE);
                                END_IF;
    'earliest end date'      : IF SIZEOF (e.items) <>
                                   SIZEOF (QUERY (x <* e.items |
                                   'STRUCTURAL_ANALYSIS_DESIGN.RETENTION'
                                   IN TYPEOF (x)))
                                   THEN RETURN(FALSE);
                                END_IF;
    'latest end date'        : IF SIZEOF (e.items) <>
                                   SIZEOF (QUERY (x <* e.items |
                                   'STRUCTURAL_ANALYSIS_DESIGN.RETENTION'
```

```
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
    'request date'         : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'ACTION_SCHEMA.VERSIONED_ACTION_REQUEST'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
    'release date'         : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'ACTION_SCHEMA.ACTION_DIRECTIVE'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
    'start date'           : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              SIZEOF (['ACTION_SCHEMA.ACTION_DIRECTIVE',
                              'STRUCTURAL_ANALYSIS_DESIGN.RETENTION'] *
                              TYPEOF (x)) = 1 ))
                              THEN RETURN(FALSE);
                           END_IF;
    'sign off date'        : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'APPROVAL_SCHEMA.' +
                              'APPROVAL_PERSON_ORGANIZATION'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
    'contract date'        : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'CONTRACT_SCHEMA.CONTRACT'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
    'certification date'   : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'CERTIFICATION_SCHEMA.CERTIFICATION'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
    'classification date'  : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'SECURITY_CLASSIFICATION_SCHEMA.' +
                              'SECURITY_CLASSIFICATION'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
    'declassification date' : IF SIZEOF (e.items) <>
                              SIZEOF (QUERY (x <* e.items |
                              'SECURITY_CLASSIFICATION_SCHEMA.' +
                              'SECURITY_CLASSIFICATION'
                              IN TYPEOF (x)))
                              THEN RETURN(FALSE);
                           END_IF;
    OTHERWISE : RETURN(TRUE);
  END_CASE;
  RETURN (TRUE);
END_FUNCTION;
(*
```

Argument definitions:

e: identifies the input **applied_date_and_time_assignment** to be checked.

5.2.6.3 applied_organization_correlation

The **applied_organization_correlation** boolean function returns true if the **name** attribute value of the **organization_role** entity is coordinated with the type of entity selected in the **items** of a **applied_-organization_assignment** entity.

EXAMPLE If the role for an **organization** is "request recipient" then all of the **items** in the set shall be **versioned_-action_request** entities.

EXPRESS specification:

```
*)
FUNCTION applied_organization_correlation
  (e : applied_organization_assignment) : BOOLEAN;
  LOCAL
    po_role : STRING;
  END_LOCAL;
    po_role := e\organization_assignment.role.name;
  CASE po_role OF
    'request recipient'      : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'ACTION_SCHEMA.' +
                                  'VERSIONED_ACTION_REQUEST'
                                  IN TYPEOF (x)))
                                  THEN RETURN(FALSE);
                               END_IF;
    'initiator'              : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  SIZEOF (['ACTION_SCHEMA.' +
                                  'VERSIONED_ACTION_REQUEST',
                                  'ACTION_SCHEMA.ACTION_DIRECTIVE'] *
                                  TYPEOF (x)) = 1 ))
                                  THEN RETURN(FALSE);
                               END_IF;
    'creator'                : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  SIZEOF (['PRODUCT_DEFINITION_SCHEMA.' +
                                  'PRODUCT_DEFINITION_FORMATION',
                                  'PRODUCT_DEFINITION_SCHEMA.' +
                                  'PRODUCT_DEFINITION'] *
                                  TYPEOF (x)) = 1 ))
                                  THEN RETURN (FALSE);
                               END_IF;
    'part supplier'          : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'PRODUCT_DEFINITION_SCHEMA.' +
                                  'PRODUCT_DEFINITION_FORMATION'
                                  IN TYPEOF (x)))
                                  THEN RETURN(FALSE);
                               END_IF;
    'analysis supplier'      : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'PRODUCT_DEFINITION_SCHEMA.' +
                                  'PRODUCT_DEFINITION_FORMATION'
                                  IN TYPEOF (x)))
                                  THEN RETURN(FALSE);
```



```
                                END_IF;
    'design supplier'         : IF SIZEOF (e.items) <>
                                   SIZEOF (QUERY (x <* e.items |
                                   'PRODUCT_DEFINITION_SCHEMA.' +
                                   'PRODUCT_DEFINITION_FORMATION'
                                   IN TYPEOF (x)))
                                   THEN RETURN(FALSE);
                                END_IF;
    'analysis owner'          : IF SIZEOF (e.items) <>
                                   SIZEOF (QUERY (x <* e.items |
                                   'PRODUCT_DEFINITION_SCHEMA.PRODUCT'
                                   IN TYPEOF (x)))
                                   THEN RETURN(FALSE);
                                END_IF;
    'design owner'            : IF SIZEOF (e.items) <>
                                   SIZEOF (QUERY (x <* e.items |
                                   'PRODUCT_DEFINITION_SCHEMA.PRODUCT'
                                   IN TYPEOF (x)))
                                   THEN RETURN(FALSE);
                                END_IF;
    'configuration manager'   : IF SIZEOF (e.items) <>
                                   SIZEOF (QUERY (x <* e.items |
                                   'CONFIGURATION_MANAGEMENT_SCHEMA.' +
                                   'CONFIGURATION_ITEM'
                                   IN TYPEOF (x)))
                                   THEN RETURN(FALSE);
                                END_IF;
    'contractor'              : IF SIZEOF (e.items) <>
                                   SIZEOF (QUERY (x <* e.items |
                                   'CONTRACT_SCHEMA.CONTRACT'
                                   IN TYPEOF (x)))
                                   THEN RETURN(FALSE);
                                END_IF;
    'classification officer' : IF SIZEOF (e.items) <>
                                   SIZEOF (QUERY (x <* e.items |
                                   'SECURITY_CLASSIFICATION_SCHEMA.' +
                                   'SECURITY_CLASSIFICATION'
                                   IN TYPEOF (x))) THEN
                                   RETURN(FALSE);
                                END_IF;
    OTHERWISE : RETURN(TRUE);
  END_CASE;
  RETURN (TRUE);
END_FUNCTION;
(*
```

Argument definitions:

e: identifies the input **applied_organization_assignment** to be checked.

5.2.6.4 applied_person_and_organization_correlation

The **applied_person_and_organization_correlation** boolean function returns true if the **name** attribute value of the **person_organization_role** entity is coordinated with the type of entity selected in the **items** of a **applied_person_and_organization_assignment** entity.

EXAMPLE If the role for a **person_and_organization** is "request recipient" then all of the **items** in the set shall be **versioned_action_request** entities.

EXPRESS specification:

```
*)
FUNCTION applied_person_and_organization_correlation
  (e : applied_person_and_organization_assignment) : BOOLEAN;
  LOCAL
    po_role : STRING;
  END_LOCAL;
    po_role := e\person_and_organization_assignment.role.name;
  CASE po_role OF
    'request recipient'      : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'ACTION_SCHEMA.' +
                                  'VERSIONED_ACTION_REQUEST'
                                  IN TYPEOF (x)))
                                  THEN RETURN(FALSE);
                               END_IF;
    'initiator'              : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  SIZEOF (['ACTION_SCHEMA.' +
                                  'VERSIONED_ACTION_REQUEST',
                                  'ACTION_SCHEMA.ACTION_DIRECTIVE'] *
                                  TYPEOF (x)) = 1 ))
                                  THEN RETURN(FALSE);
                               END_IF;
    'creator'                : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  SIZEOF (['PRODUCT_DEFINITION_SCHEMA.' +
                                  'PRODUCT_DEFINITION_FORMATION',
                                  'PRODUCT_DEFINITION_SCHEMA.' +
                                  'PRODUCT_DEFINITION'] *
                                  TYPEOF (x)) = 1 ))
                                  THEN RETURN (FALSE);
                               END_IF;
    'part supplier'          : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'PRODUCT_DEFINITION_SCHEMA.' +
                                  'PRODUCT_DEFINITION_FORMATION'
                                  IN TYPEOF (x)))
                                  THEN RETURN(FALSE);
                               END_IF;
    'analysis supplier'      : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'PRODUCT_DEFINITION_SCHEMA.' +
                                  'PRODUCT_DEFINITION_FORMATION'
                                  IN TYPEOF (x)))
                                  THEN RETURN(FALSE);
                               END_IF;
    'design supplier'        : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'PRODUCT_DEFINITION_SCHEMA.' +
                                  'PRODUCT_DEFINITION_FORMATION'
                                  IN TYPEOF (x)))
                                  THEN RETURN(FALSE);
                               END_IF;
    'analysis owner'         : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'PRODUCT_DEFINITION_SCHEMA.PRODUCT'
                                  IN TYPEOF (x)))
                                  THEN RETURN(FALSE);
                               END_IF;
    'design owner'           : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'PRODUCT_DEFINITION_SCHEMA.PRODUCT'
```



```
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                             END_IF;
    'configuration manager'  : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'CONFIGURATION_MANAGEMENT_SCHEMA.' +
                                'CONFIGURATION_ITEM'
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                             END_IF;
    'contractor'             : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'CONTRACT_SCHEMA.CONTRACT'
                                IN TYPEOF (x)))
                                THEN RETURN(FALSE);
                             END_IF;
    'classification officer' : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'SECURITY_CLASSIFICATION_SCHEMA.' +
                                'SECURITY_CLASSIFICATION'
                                IN TYPEOF (x))) THEN
                                RETURN(FALSE);
                             END_IF;
    OTHERWISE : RETURN(TRUE);
  END_CASE;
  RETURN (TRUE);
END_FUNCTION;
(*
```

Argument definitions:

e: identifies the input **applied_person_and_organization_assignment** to be checked.

5.2.6.5 assembly_shape_is_defined

The **assembly_shape_is_defined** function accepts a **next_assembly_usage_occurrence** as input and returns a boolean result. The function will return true if the shape is defined for the **product_definition** that is the **related_product_definition** and the shape is defined for the **product_definition** that is the **relating_-product_definition** in the **next_assembly_usage_occurrence**, and the two shapes are related through a **shape_representation_relationship**, and the two relationships are related through **context_dependent_-shape_representation**. The function will also return true if the shape of the **related_product_definition** or the **relating_product_definition** is not defined.

The only time this function will return false is when the shapes are defined for the **related_product_-definition** and the **relating_product_definition** and related through **shape_representation_relationship**, but the **next_assembly_usage_occurrence** and the **shape_representation_relationship** are not explicitly related through **context_dependent_shape_representation**.

EXPRESS specification:

```
*)
FUNCTION assembly_shape_is_defined
  (assy: next_assembly_usage_occurrence) : BOOLEAN;
  LOCAL
    sdr_set     : SET OF shape_definition_representation := [];
    sdr_a_set   : SET OF shape_definition_representation := [];
    srr_set     : SET OF shape_representation_relationship := [];
```

```
  pr1_set    : SET OF property_definition := [];
  pr2_set    : SET OF property_definition := [];
  pdrel_set  : SET OF product_definition_relationship := [];
  i,j        : INTEGER;
END_LOCAL;

-- Gather all instances of shape_definition_representation where
-- the component part has a representation defined for it.

pr1_set := bag_to_set (USEDIN (assy.related_product_definition,
  'PRODUCT_PROPERTY_DEFINITION_SCHEMA.PROPERTY_DEFINITION.' +
  'DEFINITION'));
REPEAT i := 1 TO HIINDEX(pr1_set) BY 1;
  sdr_set := sdr_set + QUERY (pdr <*
  bag_to_set (USEDIN (pr1_set[i],
  'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.' +
  'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) |
  'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.' +
  'SHAPE_DEFINITION_REPRESENTATION' IN TYPEOF(pdr));
END_REPEAT;

pdrel_set := bag_to_set (USEDIN (assy.related_product_definition,
  'PRODUCT_DEFINITION_SCHEMA.PRODUCT_DEFINITION_RELATIONSHIP.' +
  'RELATED_PRODUCT_DEFINITION'));
REPEAT j := 1 TO HIINDEX(pdrel_set) BY 1;
  pr2_set := pr2_set + bag_to_set (USEDIN (pdrel_set[j],
  'PRODUCT_PROPERTY_DEFINITION_SCHEMA.PROPERTY_DEFINITION.' +
  'DEFINITION'));
END_REPEAT;
REPEAT i := 1 TO HIINDEX(pr2_set) BY 1;
  sdr_set := sdr_set + QUERY (pdr <*
  bag_to_set (USEDIN(pr2_set[i],
  'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.' +
  'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) |
  'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.' +
  'SHAPE_DEFINITION_REPRESENTATION' IN TYPEOF(pdr));
END_REPEAT;

-- Gather all instances of shape_definition_representation where
-- the assembly part has a representation defined for it.

pr1_set := bag_to_set (USEDIN (assy.relating_product_definition,
  'PRODUCT_PROPERTY_DEFINITION_SCHEMA.PROPERTY_DEFINITION.' +
  'DEFINITION'));
REPEAT i := 1 TO HIINDEX(pr1_set) BY 1;
  sdr_a_set := sdr_a_set + QUERY (pdr <*
  bag_to_set (USEDIN (pr1_set[i],
  'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.' +
  'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) |
  'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.' +
  'SHAPE_DEFINITION_REPRESENTATION' IN TYPEOF(pdr));
END_REPEAT;

-- If there is a representation defined for the component part
-- and for the assembly part

IF ( (SIZEOF (sdr_set) > 0) AND (SIZEOF (sdr_a_set)) > 0) ) THEN

  -- For each representation of the shape of the component part
  -- gather all instances of shape_representation_relationship the
  -- where representation of component part is related to
  -- another representation.

  REPEAT i := 1 TO HIINDEX (sdr_set);
```



```
      srr_set := QUERY (rr <* bag_to_set (USEDIN (sdr_set[i]\
       property_definition_representation.used_representation,
      'REPRESENTATION_SCHEMA.REPRESENTATION_RELATIONSHIP.REP_2')) |
      'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.' +
      'SHAPE_REPRESENTATION_RELATIONSHIP' IN TYPEOF (rr));

      -- If there is a shape_representation_relationship
      -- where the component part's shape_representation
      -- is related to another shape_representation.

      IF SIZEOF (srr_set) > 0 THEN

        -- For each shape_representation_relationship in that set

        REPEAT j := 1 TO HIINDEX (srr_set);
          -- If the other shape_representation in the
          -- shape_representation_relationship is the
          -- shape_representation of the assembly
          -- product_definition in at least one instance.

          IF SIZEOF (QUERY (pdr <* bag_to_set (
            USEDIN (srr_set[j]\representation_relationship.rep_1,
            'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.' +
            'PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTATION')) |
            'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.' +
            'SHAPE_DEFINITION_REPRESENTATION' IN TYPEOF (pdr)) *
            sdr_a_set) >= 1 THEN

            -- If the shape_representation_relationship and the
            -- product_definition_relationship of each occurrence
            -- of the component and assembly relationship is not given
            -- via the context_dependent_shape_representation then
            -- return FALSE

            IF SIZEOF (QUERY (cdsr <* bag_to_set (USEDIN (srr_set[j],
              'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.' +
              'CONTEXT_DEPENDENT_SHAPE_REPRESENTATION.' +
              'REPRESENTATION_RELATION')) | NOT (cdsr.
               represented_product_relation.definition :=: assy) )) > 0
              THEN RETURN (FALSE);
            END_IF;
          END_IF;
        END_REPEAT;
      END_IF;
    END_REPEAT;
  END_IF;

  -- If the shape of the component is not specified
  -- or there are no violations then return TRUE
  RETURN (TRUE);
END_FUNCTION;
(*
```

Argument definitions:

assy: identifies the input **next_assembly_usage_occurrence** for which the relationships are to be checked.

5.2.6.6 derive_angle

The **derive_angle** function accepts as input two **axis2_placement_3d** with a common 3 direction and calculates the angle of rotation between them.

EXPRESS specification:

```
*)
FUNCTION derive_angle (placement_1 : axis2_placement_3d;
                       placement_2 : axis2_placement_3d) :
                       plane_angle_measure;
  LOCAL
    v1     : direction;
    v2     : direction;
    mag_v1 : REAL;
    mag_v2 : REAL;
    theta  : plane_angle_measure;
  END_LOCAL;

  v1 := placement_1.p[1];
  v2 := placement_2.p[1];

  mag_v1 := SQRT (v1.direction_ratios[1]*v1.direction_ratios[1] +
                  v1.direction_ratios[2]*v1.direction_ratios[2]);

  mag_v2 := SQRT (v2.direction_ratios[1]*v2.direction_ratios[1] +
                  v2.direction_ratios[2]*v2.direction_ratios[2]);

  IF ((mag_v1 = 0.0) OR (mag_v2 = 0.0)) THEN
    theta := 0.0;
    RETURN (theta);
  END_IF;

  theta := ACOS ((v1.direction_ratios[1]*v2.direction_ratios[1] +
                  v1.direction_ratios[2]*v2.direction_ratios[2]) /
                 (mag_v1*mag_v2));
  RETURN (theta);
END_FUNCTION;
(*
```

Argument definitions:

placement_1: the first of the two **axis2_placement_3d**s with a common direction.

placement_2: the second of the two **axis2_placement_3d**s with a common direction.

5.2.6.7 unique_version_change_order

The **unique_version_change_order** boolean function accepts an **action_directive** as input. It will return true if the **versioned_action_request**s incorporated by the **action_directive** are assigned to **applied_-action_request_assigment** entities that reference different **product_definition_formation** entities and those **product_definition_formation** entities reference different **product** entities. This function will be true if the product versions updated by an **action_directive** belong to different parts. A single **action_directive** may not change different versions of the same part.

EXPRESS specification:

```
*)
FUNCTION unique_version_change_order
 (ad : action_directive) : BOOLEAN;

  LOCAL
    varset : SET OF versioned_action_request := ad.requests;
```



```
    aara   : SET OF applied_action_request_assignment := [];
    vers   : SET OF product_definition_formation := [];
    i,k    : INTEGER;
  END_LOCAL;

  -- build the set of applied_action_request_assignments
  -- associated with the versioned_action_requests
  -- that are incorporated by the action_directive

  REPEAT i := 1 TO SIZEOF (varset);
    aara := aara + QUERY (ara <* bag_to_set (USEDIN (varset[i],
     'MANAGEMENT_RESOURCES_SCHEMA.' +
     'ACTION_REQUEST_ASSIGNMENT.ASSIGNED_ACTION_REQUEST')) |
     'STRUCTURAL_ANALYSIS_DESIGN.' +
     'APPLIED_ACTION_REQUEST_ASSIGNMENT' IN TYPEOF (ara));
  END_REPEAT;

  -- gather the product_definition_formations that are
  -- referenced by the applied_action_request_assignments
  REPEAT k := 1 TO SIZEOF (aara);
    vers := vers + bag_to_set (QUERY (aarai <* aara[k].items |
     'PRODUCT_DEFINITION_SCHEMA.PRODUCT_DEFINITION_FORMATION'
      IN TYPEOF (aarai)));
  END_REPEAT;

  -- check that no product_definition_formation
  -- reference the same instance of product
  RETURN (SIZEOF (QUERY (ver <* vers |
          NOT (SIZEOF (QUERY (other_ver <* vers - ver |
          ver.of_product :=: other_ver.of_product)) = 0 ))) = 0);
END_FUNCTION;
(*
```

Argument definitions:

ad: identifies the input **action_directive** to be checked.

```
*)
END_SCHEMA; --structural_analysis_design
(*
```

6 Conformance Requirements

Conformance to this part of ISO 10303 includes satisfying the requirements stated in this part, the requirements of the implementation method(s) supported, and the relevant requirements of the normative references.

An implementation shall support at least one of the following implementation methods: ISO 10303-21 or ISO 10303-22.

Requirements with respect to implementation methods are specified in annex C.

The Protocol Information Conformance Statement (PICS) proforma lists the options or the combination of options that may be included in the implementation. The PICS proforma is provided in annex D.

This part of ISO 10303 provides for a number of options that may be supported by an implementation. These options have been grouped into ten conformance classes that are described later in this clause. Support for a particular conformance class requires support of all the elements specified in that class.

Conformance to this part of ISO 10303 requires conformance to at least one of the primary conformance classes 7 through 10.

Classes 2 through 6 are the shape representation conformance classes that may be used for ISO 10303-209. One or more shape representation conformance classes may be selected by an implementation and combined with the primary conformance classes 7 through 10.

Class 7 is for composite part representation, including composite constituents, structure, and shape information. Class 7 needs to be used with Class 8, 9, or 10 when representing composite materials or parts in a finite element model.

Conformance to a particular conformance class requires that all AIM entities, types, and associated constraints defined as a part of that class be supported. Table 22 specifies which UoF are in each conformance class. The AIM entities for a conformance class include all the entities specified in the AIM element column of the mapping table (see clause 5.1) sections of each of the UoF in a conformance class. All entities, types and associated constraints that are required as attributes of the AIM entities of a conformance class shall also be supported.

The conformance classes of this part of ISO 10303 are characterized with combinations of Units of Functionality (see clause 4.1) as follows:

Primary conformance classes:

Class 1: Support for configuration control without shape information.

Class 2: Support for Class 1 plus shapes represented by non_topological_surface_and_wireframe.

Class 3: Support for Class 1 plus shapes represented by wireframe_with_topology.

Class 4: Support for Class 1 plus shapes represented by manifold_surface_with_topology.

Class 5: Support for Class 1 plus shapes represented by faceted_boundary_representation.

Class 6: Support for Class 1 plus shapes represented by advanced_boundary_representation.

Class 7: Support for part_identification, part_shape, material, part_composite_constituents, composite_-constituent_representation, part_laminate_table, and zone_composite_constituents_and_their_represent-ation.

Class 8: Support for part_identification, part_shape, material, fea_model, and analysis_report.

Class 9: Support for Class 8, plus fe_analysis_control.

Class 10: Support for Class 9, plus fe_analysis_results.

Table 22 - Conformance Classes

Units of functionality	Conformance Class									
	1	2	3	4	5	6	7	8	9	10
activity_control	X	X	X	X	X	X				
advanced_boundary_representation						X				
analysis_report								X	X	X
assembly	X	X	X	X	X	X				
authorization	X	X	X	X	X	X				
composite_constituents_representation							X			
effectivity	X	X	X	X	X	X				
end_item_identification	X	X	X	X	X	X				
faceted_boundary_representation					X					
fe_analysis_control									X	X
fe_analysis_results										X
fea_model								X	X	X
manifold_surface_with_topology				X						
material							X	X	X	X
non_topological_surface_and_wireframe		X								
part_composite_constituents							X			
part_identification	X	X	X	X	X	X	X	X	X	X
part_laminate_table							X			
part_shape		X	X	X	X	X	X	X	X	X
wireframe_with_topology			X							
zone_composite_constituents_and_their_-representation							X			

Annex A
(normative)

AIM EXPRESS expanded listing

The following EXPRESS is the expanded form of the short form schema given in 5.2. In the event of any discrepancy between the short form and this expanded listing, the expanded listing shall be used.

```
*)
SCHEMA structural_analysis_design;

  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_definition,
     product_definition_formation,
     product_definition_relationship);
  END_TYPE; -- action_item

  TYPE action_request_item = SELECT
    (product_definition,
     product_definition_formation,
     product_definition_relationship);
  END_TYPE; -- action_request_item

  TYPE action_type = ENUMERATION OF
    (applied_loads,
     residual_loads);
  END_TYPE; -- action_type

  TYPE aggregated_angular_variable = ENUMERATION OF
    (total_applied_moment,
     application_defined_aggregated_angular_scalar_variable);
  END_TYPE; -- aggregated_angular_variable

  TYPE aggregated_scalar_variable = ENUMERATION OF
    (total_strain_energy,
     mass,
     volume);
  END_TYPE; -- aggregated_scalar_variable

  TYPE aggregated_tensor2_3d_variable = ENUMERATION OF
    (rotational_inertia);
  END_TYPE; -- aggregated_tensor2_3d_variable

  TYPE aggregated_vector_3d_variable = ENUMERATION OF
    (total_applied_force,
     centre_of_mass_offset);
  END_TYPE; -- aggregated_vector_3d_variable

  TYPE ahead_or_behind = ENUMERATION OF
    (ahead,
```



```
   behind);
END_TYPE; -- ahead_or_behind

TYPE amount_of_substance_measure = REAL;
END_TYPE; -- amount_of_substance_measure

TYPE angular_value = context_dependent_measure;
END_TYPE; -- angular_value

TYPE anisotropic_symmetric_tensor2_2d = ARRAY [1:3] OF
                context_dependent_measure;
END_TYPE; -- anisotropic_symmetric_tensor2_2d

TYPE anisotropic_symmetric_tensor2_3d = ARRAY [1:6] OF
                context_dependent_measure;
END_TYPE; -- anisotropic_symmetric_tensor2_3d

TYPE anisotropic_symmetric_tensor4_2d = ARRAY [1:6] OF
                context_dependent_measure;
END_TYPE; -- anisotropic_symmetric_tensor4_2d

TYPE anisotropic_symmetric_tensor4_3d = ARRAY [1:21] OF
                context_dependent_measure;
END_TYPE; -- anisotropic_symmetric_tensor4_3d

TYPE application_defined_degree_of_freedom = STRING;
END_TYPE; -- application_defined_degree_of_freedom

TYPE application_defined_element_purpose = STRING;
END_TYPE; -- application_defined_element_purpose

TYPE application_defined_matrix_property_type = STRING;
END_TYPE; -- application_defined_matrix_property_type

TYPE application_defined_scalar_variable = STRING;
END_TYPE; -- application_defined_scalar_variable

TYPE application_defined_tensor2_2d_variable = STRING;
END_TYPE; -- application_defined_tensor2_2d_variable

TYPE application_defined_tensor2_3d_variable = STRING;
END_TYPE; -- application_defined_tensor2_3d_variable

TYPE application_defined_vector_2d_variable = STRING;
END_TYPE; -- application_defined_vector_2d_variable

TYPE application_defined_vector_3d_variable = STRING;
END_TYPE; -- application_defined_vector_3d_variable

TYPE approval_item = SELECT
  (action_directive,
   certification,
   configuration_item,
   configuration_effectivity,
   contract,
   control,
   fea_model,
   product_definition,
   product_definition_formation,
   result,
   retention,
```

```
   security_classification,
   versioned_action_request);
END_TYPE; -- approval_item

TYPE area_measure = REAL;
END_TYPE; -- area_measure

TYPE axi_or_plane = ENUMERATION OF
  (axisymmetric,
   planar);
END_TYPE; -- axi_or_plane

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 boundary_aggregated_variable = SELECT
  (aggregated_vector_3d_variable,
   application_defined_vector_3d_variable);
END_TYPE; -- boundary_aggregated_variable

TYPE boundary_curve_scalar_variable = ENUMERATION OF
  (normal_force_per_unit_length);
END_TYPE; -- boundary_curve_scalar_variable

TYPE boundary_curve_vector_3d_variable = ENUMERATION OF
  (applied_force_per_unit_length,
   applied_moment_per_unit_length);
END_TYPE; -- boundary_curve_vector_3d_variable

TYPE boundary_edge_variable = SELECT
  (boundary_curve_scalar_variable,
   boundary_curve_vector_3d_variable,
```

```
   application_defined_scalar_variable,
   application_defined_vector_3d_variable);
END_TYPE; -- boundary_edge_variable

TYPE boundary_surface_scalar_variable = ENUMERATION OF
  (pressure);
END_TYPE; -- boundary_surface_scalar_variable

TYPE boundary_surface_vector_3d_variable = ENUMERATION OF
  (applied_force_per_unit_area,
   applied_moment_per_unit_area);
END_TYPE; -- boundary_surface_vector_3d_variable

TYPE boundary_variable = SELECT
  (boundary_surface_scalar_variable,
   boundary_surface_vector_3d_variable,
   application_defined_scalar_variable,
   application_defined_vector_3d_variable);
END_TYPE; -- boundary_variable

TYPE certification_item = SELECT
  (product_definition_relationship);
END_TYPE; -- certification_item

TYPE characterized_definition = SELECT
  (characterized_object,
   characterized_product_definition,
   shape_definition);
END_TYPE; -- characterized_definition

TYPE characterized_material_property = SELECT
  (material_property_representation,
   product_material_composition_relationship);
END_TYPE; -- characterized_material_property

TYPE characterized_product_definition = SELECT
  (product_definition,
   product_definition_relationship);
END_TYPE; -- characterized_product_definition

TYPE context_dependent_measure = REAL;
END_TYPE; -- context_dependent_measure

TYPE contract_item = SELECT
  (product_definition_formation);
END_TYPE; -- contract_item

TYPE coordinate_system_type = ENUMERATION OF
  (cartesian,
   cylindrical,
   spherical);
END_TYPE; -- coordinate_system_type

TYPE count_measure = NUMBER;
END_TYPE; -- count_measure

TYPE curve_2d_element_descriptor = SELECT
  (axisymmetric_curve_2d_element_descriptor,
   plane_curve_2d_element_descriptor);
END_TYPE; -- curve_2d_element_descriptor
```

```
TYPE curve_2d_element_output_reference = SELECT
  (curve_2d_element_representation,
   curve_2d_element_descriptor,
   curve_2d_element_group,
   curve_2d_substructure_element_reference,
   analysis_item_within_representation);
END_TYPE; -- curve_2d_element_output_reference

TYPE curve_2d_element_representation = SELECT
  (axisymmetric_curve_2d_element_representation,
   plane_curve_2d_element_representation);
END_TYPE; -- curve_2d_element_representation

TYPE curve_2d_state_coordinate_system = SELECT
  (fea_axis2_placement_3d,
   curve_2d_element_coordinate_system);
END_TYPE; -- curve_2d_state_coordinate_system

TYPE curve_3d_element_coordinate_system = SELECT
  (aligned_curve_3d_element_coordinate_system,
   parametric_curve_3d_element_coordinate_system);
END_TYPE; -- curve_3d_element_coordinate_system

TYPE curve_3d_element_length_integration = SELECT
  (element_integration_algebraic,
   curve_3d_element_length_integration_rule,
   curve_3d_element_length_integration_explicit);
END_TYPE; -- curve_3d_element_length_integration

TYPE curve_3d_element_output_reference = SELECT
  (curve_3d_element_representation,
   curve_3d_element_descriptor,
   curve_3d_element_group,
   curve_3d_substructure_element_reference,
   analysis_item_within_representation);
END_TYPE; -- curve_3d_element_output_reference

TYPE curve_3d_state_coordinate_system = SELECT
  (fea_axis2_placement_3d,
   curve_3d_element_coordinate_system);
END_TYPE; -- curve_3d_state_coordinate_system

TYPE curve_edge = ENUMERATION OF
  (element_edge);
END_TYPE; -- curve_edge

TYPE curve_element_end_coordinate_system = SELECT
  (fea_axis2_placement_3d,
   curve_3d_element_coordinate_system);
END_TYPE; -- curve_element_end_coordinate_system

TYPE curve_element_freedom = SELECT
  (enumerated_curve_element_freedom,
   application_defined_degree_of_freedom);
END_TYPE; -- curve_element_freedom

TYPE curve_element_purpose = SELECT
  (enumerated_curve_element_purpose,
   application_defined_element_purpose);
END_TYPE; -- curve_element_purpose
```



```
TYPE curve_element_variable = SELECT
  (volume_variable,
   curve_scalar_variable,
   curve_vector_2d_variable,
   application_defined_vector_2d_variable,
   curve_vector_3d_variable);
END_TYPE; -- curve_element_variable

TYPE curve_matrix_property_type = SELECT
  (enumerated_curve_matrix_property_type,
   application_defined_matrix_property_type);
END_TYPE; -- curve_matrix_property_type

TYPE curve_on_surface = SELECT
  (pcurve,
   surface_curve,
   composite_curve_on_surface);
END_TYPE; -- curve_on_surface

TYPE curve_scalar_variable = ENUMERATION OF
  (curve_axial_force,
   curve_axial_strain,
   torque,
   curve_warping,
   bi_moment,
   twist);
END_TYPE; -- curve_scalar_variable

TYPE curve_vector_2d_variable = ENUMERATION OF
  (curve_shear_force,
   curve_bending_moment,
   curve_element_curvature,
   curve_thermal_gradient,
   reference_curve_thermal_gradient);
END_TYPE; -- curve_vector_2d_variable

TYPE curve_vector_3d_variable = ENUMERATION OF
  (applied_force_per_unit_length,
   applied_moment_per_unit_length);
END_TYPE; -- curve_vector_3d_variable

TYPE cylindrical_harmonic_number = INTEGER;
WHERE
  wr1: (SELF >= 0);
END_TYPE; -- cylindrical_harmonic_number

TYPE date_item = SELECT
  (action_directive,
   approval_person_organization,
   certification,
   contract,
   product_definition,
   retention,
   security_classification,
   versioned_action_request);
END_TYPE; -- date_item

TYPE date_time_item = SELECT
  (action_directive,
   approval_person_organization,
   certification,
```

```
   contract,
   product_definition,
   retention,
   security_classification,
   versioned_action_request);
END_TYPE; -- date_time_item

TYPE date_time_select = SELECT
  (date,
   local_time,
   date_and_time);
END_TYPE; -- date_time_select

TYPE day_in_month_number = INTEGER;
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;
END_TYPE; -- day_in_year_number

TYPE degree_of_freedom = SELECT
  (enumerated_degree_of_freedom,
   application_defined_degree_of_freedom);
END_TYPE; -- degree_of_freedom

TYPE descriptive_measure = STRING;
END_TYPE; -- descriptive_measure

TYPE dimension_count = INTEGER;
WHERE
  wr1: (SELF > 0);
END_TYPE; -- dimension_count

TYPE dimensional_characteristic = SELECT
  (dimensional_size);
END_TYPE; -- dimensional_characteristic

TYPE directionally_explicit_element_coordinate_system = SELECT
  (directionally_explicit_element_coordinate_system_arbitrary,
   directionally_explicit_element_coordinate_system_aligned);
END_TYPE; -- directionally_explicit_element_coordinate_system

TYPE document_reference_item = SELECT
  (product_definition,
   shape_aspect);
END_TYPE; -- document_reference_item

TYPE electric_current_measure = REAL;
END_TYPE; -- electric_current_measure

TYPE element_2d_shape = ENUMERATION OF
  (quadrilateral,
   triangle);
END_TYPE; -- element_2d_shape

TYPE element_aspect = SELECT
  (element_volume,
```



```
   volume_3d_face,
   volume_2d_face,
   volume_3d_edge,
   volume_2d_edge,
   surface_3d_face,
   surface_2d_face,
   surface_3d_edge,
   surface_2d_edge,
   curve_edge);
END_TYPE; -- element_aspect

TYPE element_integration_algebraic = ENUMERATION OF
  (algebraic);
END_TYPE; -- element_integration_algebraic

TYPE element_or_element_group = SELECT
  (element_representation,
   element_group);
END_TYPE; -- element_or_element_group

TYPE element_order = ENUMERATION OF
  (linear,
   quadratic,
   cubic);
END_TYPE; -- element_order

TYPE element_volume = ENUMERATION OF
  (volume);
END_TYPE; -- element_volume

TYPE enumerated_curve_element_freedom = ENUMERATION OF
  (x_translation,
   y_translation,
   z_translation,
   x_rotation,
   y_rotation,
   z_rotation,
   warp,
   none);
END_TYPE; -- enumerated_curve_element_freedom

TYPE enumerated_curve_element_purpose = ENUMERATION OF
  (axial,
   y_y_bending,
   z_z_bending,
   torsion,
   x_y_shear,
   x_z_shear,
   warping);
END_TYPE; -- enumerated_curve_element_purpose

TYPE enumerated_curve_matrix_property_type = ENUMERATION OF
  (axial,
   y_y_bending,
   z_z_bending,
   torsion,
   x_y_shear,
   x_z_shear,
   warping,
   axial_mass,
   y_y_bending_mass,
```

```
   z_z_bending_mass,
   torsion_mass,
   x_y_shear_mass,
   x_z_shear_mass,
   warping_mass,
   mass);
END_TYPE; -- enumerated_curve_matrix_property_type

TYPE enumerated_degree_of_freedom = ENUMERATION OF
  (x_translation,
   y_translation,
   z_translation,
   x_rotation,
   y_rotation,
   z_rotation,
   warp);
END_TYPE; -- enumerated_degree_of_freedom

TYPE enumerated_matrix_property_type = ENUMERATION OF
  (stiffness,
   mass,
   damping);
END_TYPE; -- enumerated_matrix_property_type

TYPE enumerated_plane_2d_element_purpose = ENUMERATION OF
  (plane_stress,
   plane_strain);
END_TYPE; -- enumerated_plane_2d_element_purpose

TYPE enumerated_surface_element_purpose = ENUMERATION OF
  (membrane_direct,
   membrane_shear,
   bending_direct,
   bending_torsion,
   normal_to_plane_shear);
END_TYPE; -- enumerated_surface_element_purpose

TYPE enumerated_surface_matrix_property_type = ENUMERATION OF
  (membrane_direct,
   membrane_shear,
   bending_direct,
   bending_torsion,
   normal_to_plane_shear,
   membrane_direct_mass,
   membrane_shear_mass,
   bending_direct_mass,
   bending_torsion_mass,
   normal_to_plane_shear_mass,
   mass);
END_TYPE; -- enumerated_surface_matrix_property_type

TYPE enumerated_volume_element_purpose = ENUMERATION OF
  (stress_displacement);
END_TYPE; -- enumerated_volume_element_purpose

TYPE fea_column_normalised_monoclinic_symmetric_tensor4_3d = ARRAY [1:13]
            OF context_dependent_measure;
END_TYPE; -- fea_column_normalised_monoclinic_symmetric_tensor4_3d

TYPE fea_column_normalised_orthotropic_symmetric_tensor4_3d = ARRAY [1:9]
            OF context_dependent_measure;
```



```
END_TYPE; -- fea_column_normalised_orthotropic_symmetric_tensor4_3d

TYPE fea_iso_orthotropic_symmetric_tensor4_3d = ARRAY [1:3] OF
            context_dependent_measure;
END_TYPE; -- fea_iso_orthotropic_symmetric_tensor4_3d

TYPE fea_isotropic_symmetric_tensor4_3d = ARRAY [1:2] OF
            context_dependent_measure;
END_TYPE; -- fea_isotropic_symmetric_tensor4_3d

TYPE fea_transverse_isotropic_symmetric_tensor4_3d = ARRAY [1:5] OF
            context_dependent_measure;
END_TYPE; -- fea_transverse_isotropic_symmetric_tensor4_3d

TYPE field_value = SELECT
  (unspecified_value,
   scalar,
   tensor1_2d,
   tensor1_3d,
   anisotropic_symmetric_tensor2_2d,
   isotropic_symmetric_tensor2_3d,
   orthotropic_symmetric_tensor2_3d,
   anisotropic_symmetric_tensor2_3d);
END_TYPE; -- field_value

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 hour_in_day = INTEGER;
WHERE
  wr1: ((0 <= SELF) AND (SELF < 24));
END_TYPE; -- hour_in_day

TYPE identifier = STRING;
END_TYPE; -- identifier

TYPE integration_rule = ENUMERATION OF
  (gaussian,
   simpson);
END_TYPE; -- integration_rule

TYPE isotropic_symmetric_tensor2_3d = context_dependent_measure;
END_TYPE; -- isotropic_symmetric_tensor2_3d

TYPE knot_type = ENUMERATION OF
  (uniform_knots,
   unspecified,
   quasi_uniform_knots,
   piecewise_bezier_knots);
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 luminous_intensity_measure = REAL;
END_TYPE; -- luminous_intensity_measure

TYPE mass_measure = REAL;
END_TYPE; -- mass_measure

TYPE matrix_property_type = SELECT
  (enumerated_matrix_property_type,
   application_defined_matrix_property_type);
END_TYPE; -- matrix_property_type

TYPE matrix_symmetry = ENUMERATION OF
  (symmetric,
   diagonal);
END_TYPE; -- matrix_symmetry

TYPE measure_or_unspecified_value = SELECT
  (context_dependent_measure,
   unspecified_value);
END_TYPE; -- measure_or_unspecified_value

TYPE measure_value = SELECT
  (length_measure,
   mass_measure,
   time_measure,
   electric_current_measure,
   thermodynamic_temperature_measure,
   amount_of_substance_measure,
   luminous_intensity_measure,
   plane_angle_measure,
   solid_angle_measure,
   area_measure,
   volume_measure,
   ratio_measure,
   parameter_value,
   numeric_measure,
   context_dependent_measure,
   descriptive_measure,
   positive_length_measure,
   positive_plane_angle_measure,
   positive_ratio_measure,
   count_measure);
END_TYPE; -- measure_value

TYPE message_level = ENUMERATION OF
  (error,
   warning,
   note);
END_TYPE; -- message_level

TYPE minute_in_hour = INTEGER;
WHERE
  wr1: ((0 <= SELF) AND (SELF <= 59));
END_TYPE; -- minute_in_hour
```



```
TYPE model_or_control_element = SELECT
  (element_representation,
   constraint_element);
END_TYPE; -- model_or_control_element

TYPE month_in_year_number = INTEGER;
WHERE
  wr1: ((1 <= SELF) AND (SELF <= 12));
END_TYPE; -- month_in_year_number

TYPE name_item = SELECT
  (expression_conversion_based_unit);
END_TYPE; -- name_item

TYPE node_or_node_group = SELECT
  (node_representation,
   node_group);
END_TYPE; -- node_or_node_group

TYPE node_output_reference = SELECT
  (node_representation,
   node_group,
   substructure_node_reference,
   analysis_item_within_representation);
END_TYPE; -- node_output_reference

TYPE numeric_measure = NUMBER;
END_TYPE; -- numeric_measure

TYPE organization_item = SELECT
  (action_directive,
   configuration_item,
   contract,
   product,
   product_definition,
   product_definition_formation,
   security_classification,
   versioned_action_request);
END_TYPE; -- organization_item

TYPE orthotropic_symmetric_tensor2_3d = ARRAY [1:3] OF
            context_dependent_measure;
END_TYPE; -- orthotropic_symmetric_tensor2_3d

TYPE parameter_value = REAL;
END_TYPE; -- parameter_value

TYPE pcurve_or_surface = SELECT
  (pcurve,
   surface);
END_TYPE; -- pcurve_or_surface

TYPE person_organization_item = SELECT
  (action_directive,
   configuration_item,
   contract,
   product,
   product_definition,
   product_definition_formation,
   security_classification,
   versioned_action_request);
```

```
END_TYPE; -- person_organization_item

TYPE person_organization_select = SELECT
  (person,
   organization,
   person_and_organization);
END_TYPE; -- person_organization_select

TYPE plane_2d_element_purpose = SELECT
  (enumerated_plane_2d_element_purpose,
   application_defined_element_purpose);
END_TYPE; -- plane_2d_element_purpose

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 positive_ratio_measure = ratio_measure;
WHERE
  wr1: (SELF > 0);
END_TYPE; -- positive_ratio_measure

TYPE preferred_surface_curve_representation = ENUMERATION OF
  (curve_3d,
   pcurve_s1,
   pcurve_s2);
END_TYPE; -- preferred_surface_curve_representation

TYPE ratio_measure = REAL;
END_TYPE; -- ratio_measure

TYPE retention_item = SELECT
  (control,
   document,
   material_property,
   product,
   product_definition,
   product_definition_formation,
   product_definition_relationship,
   representation,
   result);
END_TYPE; -- retention_item

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 scalar = context_dependent_measure;
END_TYPE; -- scalar

TYPE second_in_minute = REAL;
WHERE
  wr1: ((0 <= SELF) AND (SELF < 60));
END_TYPE; -- second_in_minute

TYPE security_classification_item = SELECT
  (product_definition,
   product_definition_formation,
   product_definition_usage);
END_TYPE; -- security_classification_item

TYPE set_of_reversible_topology_item = SET [0:?] OF
            reversible_topology_item;
END_TYPE; -- set_of_reversible_topology_item

TYPE shape_definition = SELECT
  (product_definition_shape,
   shape_aspect,
   shape_aspect_relationship);
END_TYPE; -- shape_definition

TYPE shape_function = ENUMERATION OF
  (lagrangian,
   serendipity,
   hermitian,
   unspecified);
END_TYPE; -- shape_function

TYPE shell = SELECT
  (vertex_shell,
   wire_shell,
   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 = ENUMERATION OF
  (made,
   bought,
   not_known);
END_TYPE; -- source

TYPE supported_item = SELECT
  (action_directive,
   action,
   action_method);
END_TYPE; -- supported_item

TYPE surface_2d_edge = INTEGER;
WHERE
  wr1: ((SELF >= 1) AND (SELF <= 2));
END_TYPE; -- surface_2d_edge

TYPE surface_2d_element_coordinate_system = SELECT
  (aligned_surface_2d_element_coordinate_system,
   parametric_surface_2d_element_coordinate_system);
END_TYPE; -- surface_2d_element_coordinate_system

TYPE surface_2d_element_descriptor = SELECT
  (axisymmetric_surface_2d_element_descriptor,
   plane_surface_2d_element_descriptor);
```



```
END_TYPE; -- surface_2d_element_descriptor

TYPE surface_2d_element_length_integration = SELECT
  (element_integration_algebraic,
   surface_2d_element_length_integration_rule,
   surface_2d_element_length_integration_explicit);
END_TYPE; -- surface_2d_element_length_integration

TYPE surface_2d_element_output_reference = SELECT
  (surface_2d_element_representation,
   surface_2d_element_descriptor,
   surface_2d_element_group,
   surface_2d_substructure_element_reference,
   analysis_item_within_representation);
END_TYPE; -- surface_2d_element_output_reference

TYPE surface_2d_element_representation = SELECT
  (axisymmetric_surface_2d_element_representation,
   plane_surface_2d_element_representation);
END_TYPE; -- surface_2d_element_representation

TYPE surface_2d_face = INTEGER;
WHERE
  wr1: ((SELF >= 1) AND (SELF <= 2));
END_TYPE; -- surface_2d_face

TYPE surface_2d_state_coordinate_system = SELECT
  (fea_axis2_placement_3d,
   surface_2d_element_coordinate_system);
END_TYPE; -- surface_2d_state_coordinate_system

TYPE surface_3d_edge = INTEGER;
WHERE
  wr1: ((SELF >= 1) AND (SELF <= 4));
END_TYPE; -- surface_3d_edge

TYPE surface_3d_element_coordinate_system = SELECT
  (aligned_surface_3d_element_coordinate_system,
   parametric_surface_3d_element_coordinate_system,
   constant_surface_3d_element_coordinate_system);
END_TYPE; -- surface_3d_element_coordinate_system

TYPE surface_3d_element_field_integration = SELECT
  (element_integration_algebraic,
   surface_3d_element_field_integration_rule,
   surface_3d_element_field_integration_explicit);
END_TYPE; -- surface_3d_element_field_integration

TYPE surface_3d_element_output_reference = SELECT
  (surface_3d_element_representation,
   surface_3d_element_descriptor,
   surface_3d_element_group,
   surface_3d_substructure_element_reference,
   analysis_item_within_representation);
END_TYPE; -- surface_3d_element_output_reference

TYPE surface_3d_face = INTEGER;
WHERE
  wr1: ((SELF >= 1) AND (SELF <= 2));
END_TYPE; -- surface_3d_face
```

```
TYPE surface_3d_state_coordinate_system = SELECT
  (fea_axis2_placement_3d,
   surface_3d_element_coordinate_system);
END_TYPE; -- surface_3d_state_coordinate_system

TYPE surface_element_purpose = SELECT
  (enumerated_surface_element_purpose,
   application_defined_element_purpose);
END_TYPE; -- surface_element_purpose

TYPE surface_element_variable = SELECT
  (volume_variable,
   surface_scalar_variable,
   surface_vector_2d_variable,
   surface_vector_3d_variable,
   surface_tensor2_2d_variable,
   application_defined_tensor2_2d_variable);
END_TYPE; -- surface_element_variable

TYPE surface_matrix_property_type = SELECT
  (enumerated_surface_matrix_property_type,
   application_defined_matrix_property_type);
END_TYPE; -- surface_matrix_property_type

TYPE surface_model = SELECT
  (shell_based_surface_model);
END_TYPE; -- surface_model

TYPE surface_scalar_variable = ENUMERATION OF
  (thickness,
   surface_thermal_gradient,
   reference_surface_thermal_gradient);
END_TYPE; -- surface_scalar_variable

TYPE surface_section_integration = SELECT
  (element_integration_algebraic,
   surface_section_integration_rule,
   surface_section_integration_explicit);
END_TYPE; -- surface_section_integration

TYPE surface_tensor2_2d_variable = ENUMERATION OF
  (surface_membrane_force,
   surface_membrane_strain,
   surface_bending_moment,
   surface_curvature);
END_TYPE; -- surface_tensor2_2d_variable

TYPE surface_vector_2d_variable = ENUMERATION OF
  (surface_out_of_plane_shear_force,
   surface_out_of_plane_shear_strain);
END_TYPE; -- surface_vector_2d_variable

TYPE surface_vector_3d_variable = ENUMERATION OF
  (applied_force_per_unit_area,
   applied_moment_per_unit_area);
END_TYPE; -- surface_vector_3d_variable

TYPE symmetric_tensor2_2d = SELECT
  (anisotropic_symmetric_tensor2_2d);
END_TYPE; -- symmetric_tensor2_2d
```



```
TYPE symmetric_tensor2_3d = SELECT
  (isotropic_symmetric_tensor2_3d,
   orthotropic_symmetric_tensor2_3d,
   anisotropic_symmetric_tensor2_3d);
END_TYPE; -- symmetric_tensor2_3d

TYPE symmetric_tensor4_2d = SELECT
  (anisotropic_symmetric_tensor4_2d);
END_TYPE; -- symmetric_tensor4_2d

TYPE symmetric_tensor4_3d = SELECT
  (anisotropic_symmetric_tensor4_3d,
   fea_isotropic_symmetric_tensor4_3d,
   fea_iso_orthotropic_symmetric_tensor4_3d,
   fea_transverse_isotropic_symmetric_tensor4_3d,
   fea_column_normalised_orthotropic_symmetric_tensor4_3d,
   fea_column_normalised_monoclinic_symmetric_tensor4_3d);
END_TYPE; -- symmetric_tensor4_3d

TYPE tensor1 = SELECT
  (tensor1_2d,
   tensor1_3d);
END_TYPE; -- tensor1

TYPE tensor1_2d = ARRAY [1:2] OF context_dependent_measure;
END_TYPE; -- tensor1_2d

TYPE tensor1_3d = ARRAY [1:3] OF context_dependent_measure;
END_TYPE; -- tensor1_3d

TYPE tensor_type = SELECT
  (scalar,
   angular_value,
   tensor1_2d,
   tensor1_3d,
   anisotropic_symmetric_tensor2_2d,
   isotropic_symmetric_tensor2_3d,
   orthotropic_symmetric_tensor2_3d,
   anisotropic_symmetric_tensor2_3d,
   anisotropic_symmetric_tensor4_2d,
   anisotropic_symmetric_tensor4_3d,
   fea_isotropic_symmetric_tensor4_3d,
   fea_iso_orthotropic_symmetric_tensor4_3d,
   fea_transverse_isotropic_symmetric_tensor4_3d,
   fea_column_normalised_orthotropic_symmetric_tensor4_3d,
   fea_column_normalised_monoclinic_symmetric_tensor4_3d);
END_TYPE; -- tensor_type

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_preference = ENUMERATION OF
  (cartesian,
   parameter,
   unspecified);
END_TYPE; -- trimming_preference

TYPE trimming_select = SELECT
  (cartesian_point,
   parameter_value);
END_TYPE; -- trimming_select

TYPE unit = SELECT
  (named_unit,
   derived_unit);
END_TYPE; -- unit

TYPE unspecified_value = ENUMERATION OF
  (unspecified);
END_TYPE; -- unspecified_value

TYPE value_qualifier = SELECT
  (precision_qualifier,
   type_qualifier,
   uncertainty_qualifier);
END_TYPE; -- value_qualifier

TYPE vector_or_direction = SELECT
  (vector,
   direction);
END_TYPE; -- vector_or_direction

TYPE volume_2d_edge = INTEGER;
WHERE
  wr1: ((SELF >= 1) AND (SELF <= 4));
END_TYPE; -- volume_2d_edge

TYPE volume_2d_element_coordinate_system = SELECT
  (arbitrary_volume_2d_element_coordinate_system,
   parametric_volume_2d_element_coordinate_system);
END_TYPE; -- volume_2d_element_coordinate_system

TYPE volume_2d_element_descriptor = SELECT
  (axisymmetric_volume_2d_element_descriptor,
   plane_volume_2d_element_descriptor);
END_TYPE; -- volume_2d_element_descriptor

TYPE volume_2d_element_field_integration = SELECT
  (element_integration_algebraic,
   volume_2d_element_field_integration_rule,
   volume_2d_element_field_integration_explicit);
END_TYPE; -- volume_2d_element_field_integration

TYPE volume_2d_element_output_reference = SELECT
  (volume_2d_element_representation,
```



```
   volume_2d_element_descriptor,
   volume_2d_element_group,
   volume_2d_substructure_element_reference,
   analysis_item_within_representation);
END_TYPE; -- volume_2d_element_output_reference

TYPE volume_2d_element_representation = SELECT
  (axisymmetric_volume_2d_element_representation,
   plane_volume_2d_element_representation);
END_TYPE; -- volume_2d_element_representation

TYPE volume_2d_face = INTEGER;
WHERE
  wr1: ((SELF >= 1) AND (SELF <= 4));
END_TYPE; -- volume_2d_face

TYPE volume_3d_edge = INTEGER;
WHERE
  wr1: ((SELF >= 1) AND (SELF <= 12));
END_TYPE; -- volume_3d_edge

TYPE volume_3d_element_coordinate_system = SELECT
  (arbitrary_volume_3d_element_coordinate_system,
   parametric_volume_3d_element_coordinate_system);
END_TYPE; -- volume_3d_element_coordinate_system

TYPE volume_3d_element_field_integration = SELECT
  (element_integration_algebraic,
   volume_3d_element_field_integration_rule,
   volume_3d_element_field_integration_explicit);
END_TYPE; -- volume_3d_element_field_integration

TYPE volume_3d_element_output_reference = SELECT
  (volume_3d_element_representation,
   volume_3d_element_descriptor,
   volume_3d_element_group,
   volume_3d_substructure_element_reference,
   analysis_item_within_representation);
END_TYPE; -- volume_3d_element_output_reference

TYPE volume_3d_element_shape = ENUMERATION OF
  (hexahedron,
   wedge,
   tetrahedron,
   pyramid);
END_TYPE; -- volume_3d_element_shape

TYPE volume_3d_face = INTEGER;
WHERE
  wr1: ((SELF >= 1) AND (SELF <= 6));
END_TYPE; -- volume_3d_face

TYPE volume_aggregated_variable = SELECT
  (aggregated_scalar_variable,
   aggregated_angular_variable,
   aggregated_vector_3d_variable,
   aggregated_tensor2_3d_variable,
   application_defined_scalar_variable,
   application_defined_vector_3d_variable,
   application_defined_tensor2_3d_variable);
END_TYPE; -- volume_aggregated_variable
```

```
TYPE volume_angular_variable = ENUMERATION OF
  (constant_angular_acceleration,
   application_defined_angular_scalar_variable);
END_TYPE; -- volume_angular_variable

TYPE volume_element_purpose = SELECT
  (enumerated_volume_element_purpose,
   application_defined_element_purpose);
END_TYPE; -- volume_element_purpose

TYPE volume_measure = REAL;
END_TYPE; -- volume_measure

TYPE volume_scalar_variable = ENUMERATION OF
  (temperature,
   moisture,
   reference_temperature,
   strain_energy_per_unit_volume);
END_TYPE; -- volume_scalar_variable

TYPE volume_tensor2_3d_variable = ENUMERATION OF
  (total_strain,
   stress);
END_TYPE; -- volume_tensor2_3d_variable

TYPE volume_variable = SELECT
  (volume_scalar_variable,
   volume_angular_variable,
   volume_vector_3d_variable,
   volume_tensor2_3d_variable,
   application_defined_scalar_variable,
   application_defined_vector_3d_variable,
   application_defined_tensor2_3d_variable);
END_TYPE; -- volume_variable

TYPE volume_vector_3d_variable = ENUMERATION OF
  (position,
   applied_force_per_unit_volume,
   applied_moment_per_unit_volume,
   displacement,
   infinitesimal_rotation,
   acceleration);
END_TYPE; -- volume_vector_3d_variable

TYPE week_in_year_number = INTEGER;
WHERE
  wr1: ((1 <= SELF) AND (SELF <= 53));
END_TYPE; -- week_in_year_number

TYPE wireframe_model = SELECT
  (shell_based_wireframe_model,
   edge_based_wireframe_model);
END_TYPE; -- wireframe_model

TYPE year_number = INTEGER;
END_TYPE; -- year_number

ENTITY action;
    name          : label;
    description   : text;
    chosen_method : action_method;
```

```
END_ENTITY; -- action

ENTITY action_assignment
  ABSTRACT SUPERTYPE;
    assigned_action : action;
END_ENTITY; -- action_assignment

ENTITY action_directive;
    name        : label;
    description : text;
    analysis    : text;
    comment     : text;
    requests    : SET [1:?] OF versioned_action_request;
END_ENTITY; -- action_directive

ENTITY action_method;
    name        : label;
    description : text;
    consequence : text;
    purpose     : text;
END_ENTITY; -- action_method

ENTITY action_request_assignment
  ABSTRACT SUPERTYPE;
    assigned_action_request : versioned_action_request;
END_ENTITY; -- action_request_assignment

ENTITY action_request_solution;
    method  : action_method;
    request : versioned_action_request;
END_ENTITY; -- action_request_solution

ENTITY action_request_status;
    status           : label;
    assigned_request : versioned_action_request;
END_ENTITY; -- action_request_status

ENTITY action_status;
    status          : label;
    assigned_action : executed_action;
END_ENTITY; -- action_status

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;
  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_brep_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    wr1: (SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
             'STRUCTURAL_ANALYSIS_DESIGN.MANIFOLD_SOLID_BREP',
             'STRUCTURAL_ANALYSIS_DESIGN.FACETED_BREP',
             'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM',
             'STRUCTURAL_ANALYSIS_DESIGN.AXIS2_PLACEMENT_3D'] * TYPEOF(it))
             = 1)) )) = 0);
    wr2: (SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
             'STRUCTURAL_ANALYSIS_DESIGN.MANIFOLD_SOLID_BREP',
             'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM'] * TYPEOF(it)) = 1)
))
             > 0);
    wr3: (SIZEOF(QUERY ( msb <* QUERY ( it <* SELF.items | (
             'STRUCTURAL_ANALYSIS_DESIGN.MANIFOLD_SOLID_BREP' IN TYPEOF(
             it)) ) | (NOT (SIZEOF(QUERY ( csh <* msb_shells(msb) | (NOT
             (SIZEOF(QUERY ( fcs <* csh\connected_face_set.cfs_faces | (
             NOT ('STRUCTURAL_ANALYSIS_DESIGN.ADVANCED_FACE' IN TYPEOF(
             fcs))) )) = 0)) )) = 0)) )) = 0);
    wr4: (SIZEOF(QUERY ( msb <* QUERY ( it <* items | (
             'STRUCTURAL_ANALYSIS_DESIGN.MANIFOLD_SOLID_BREP' IN TYPEOF(
             it)) ) | ('STRUCTURAL_ANALYSIS_DESIGN.ORIENTED_CLOSED_SHELL'
             IN TYPEOF(msb\manifold_solid_brep.outer)) )) = 0);
    wr5: (SIZEOF(QUERY ( brv <* QUERY ( it <* items | (
             'STRUCTURAL_ANALYSIS_DESIGN.BREP_WITH_VOIDS' IN TYPEOF(it)) )
              | (NOT (SIZEOF(QUERY ( csh <* brv\brep_with_voids.voids |
             csh\oriented_closed_shell.orientation )) = 0)) )) = 0);
    wr6: (SIZEOF(QUERY ( mi <* QUERY ( it <* items | (
             'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM' IN TYPEOF(it)) ) |
             (NOT
('STRUCTURAL_ANALYSIS_DESIGN.ADVANCED_BREP_SHAPE_REPRESENTATION'
             IN TYPEOF(mi\mapped_item.mapping_source.
             mapped_representation))) )) = 0);
END_ENTITY; -- advanced_brep_shape_representation

ENTITY advanced_face
  SUBTYPE OF (face_surface);
  WHERE
    wr1 : (SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.ELEMENTARY_SURFACE',
              'STRUCTURAL_ANALYSIS_DESIGN.B_SPLINE_SURFACE',
              'STRUCTURAL_ANALYSIS_DESIGN.SWEPT_SURFACE'] * TYPEOF(
              face_geometry)) = 1);
    wr2 : (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
              'STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP' IN TYPEOF(bnds.bound))
)
               | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
              edge_list | (NOT ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_CURVE'
              IN TYPEOF(oe\oriented_edge.edge_element))) )) = 0)) )) = 0);
    wr3 : (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
              'STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP' IN TYPEOF(bnds.bound))
)
               | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
              edge_list | (NOT (SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.LINE',
              'STRUCTURAL_ANALYSIS_DESIGN.CONIC',
              'STRUCTURAL_ANALYSIS_DESIGN.POLYLINE',
              'STRUCTURAL_ANALYSIS_DESIGN.SURFACE_CURVE',
              'STRUCTURAL_ANALYSIS_DESIGN.B_SPLINE_CURVE'] * TYPEOF(oe.
              edge_element\edge_curve.edge_geometry)) = 1)) )) = 0)) )) =
```

```
                0);
      wr4 : (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
                'STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP' IN TYPEOF(bnds.bound))
)
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list | (NOT ((
                'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_POINT' IN TYPEOF(oe\edge
                .edge_start)) AND (
                'STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT' IN TYPEOF(oe\
                edge.edge_start\vertex_point.vertex_geometry)) AND (
                'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_POINT' IN TYPEOF(oe\edge
                .edge_end)) AND (
                'STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT' IN TYPEOF(oe\
                edge.edge_end\vertex_point.vertex_geometry)))) )) = 0)) ))
                = 0);
      wr5 : (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
                'STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP' IN TYPEOF(bnds.bound))
)
                 | ('STRUCTURAL_ANALYSIS_DESIGN.ORIENTED_PATH' IN TYPEOF(
                elp_fbnds.bound)) )) = 0);
      wr6 : ((NOT ('STRUCTURAL_ANALYSIS_DESIGN.SWEPT_SURFACE' IN TYPEOF(
                face_geometry))) OR (SIZEOF([
                'STRUCTURAL_ANALYSIS_DESIGN.LINE',
                'STRUCTURAL_ANALYSIS_DESIGN.CONIC',
                'STRUCTURAL_ANALYSIS_DESIGN.POLYLINE',
                'STRUCTURAL_ANALYSIS_DESIGN.B_SPLINE_CURVE'] * TYPEOF(
                face_geometry\swept_surface.swept_curve)) = 1));
      wr7 : (SIZEOF(QUERY ( vlp_fbnds <* QUERY ( bnds <* bounds | (
                'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_LOOP' IN TYPEOF(bnds.
                bound)) ) | (NOT ((
                'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_POINT' IN TYPEOF(
                vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex)) AND (
                'STRUCTURAL_ANALYSIS_DESIGN.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([
                'STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP',
                'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_LOOP'] * TYPEOF(bnd.
                bound)) = 1)) )) = 0);
      wr9 : (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* bounds | (
                'STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP' IN TYPEOF(bnds.bound))
)
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list | (('STRUCTURAL_ANALYSIS_DESIGN.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 (
                'STRUCTURAL_ANALYSIS_DESIGN.PCURVE' IN TYPEOF(sc_ag))) )) =
                 0))) )) = 0)) )) = 0);
      wr10: (((NOT ('STRUCTURAL_ANALYSIS_DESIGN.SWEPT_SURFACE' IN TYPEOF(
                face_geometry))) OR (NOT (
                'STRUCTURAL_ANALYSIS_DESIGN.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 | ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP' IN TYPEOF(
                bnds.bound)) ) | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound
                \path.edge_list | (('STRUCTURAL_ANALYSIS_DESIGN.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 aligned_axis_tolerance;
      model_ref : fea_model;
      tolerance : context_dependent_measure;
  END_ENTITY; -- aligned_axis_tolerance

  ENTITY aligned_curve_3d_element_coordinate_system
    SUBTYPE OF (fea_representation_item);
      coordinate_system : fea_axis2_placement_3d;
  END_ENTITY; -- aligned_curve_3d_element_coordinate_system

  ENTITY aligned_surface_2d_element_coordinate_system
    SUBTYPE OF (fea_representation_item);
      orientation : direction;
    WHERE
      wr1: (SELF\geometric_representation_item.dim = 2);
  END_ENTITY; -- aligned_surface_2d_element_coordinate_system

  ENTITY aligned_surface_3d_element_coordinate_system
    SUBTYPE OF (fea_representation_item);
      coordinate_system : fea_axis2_placement_3d;
  END_ENTITY; -- aligned_surface_3d_element_coordinate_system

  ENTITY alternate_product_relationship;
      name       : label;
      definition : text;
      alternate  : product;
      base       : product;
      basis      : text;
    UNIQUE
      ur1 : alternate, base;
    WHERE
      wr1: (alternate :<>: base);
  END_ENTITY; -- alternate_product_relationship

  ENTITY amount_of_substance_measure_with_unit
    SUBTYPE OF (measure_with_unit);
    WHERE
      wr1: ('STRUCTURAL_ANALYSIS_DESIGN.AMOUNT_OF_SUBSTANCE_UNIT' IN
                TYPEOF(SELF\measure_with_unit.unit_component));
  END_ENTITY; -- amount_of_substance_measure_with_unit

  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 analysis_item_within_representation;
      name        : label;
```



```
    description : text;
    item        : representation_item;
    rep         : representation;
  WHERE
    wr1: (SIZEOF(QUERY ( tmp <* using_representations(item) | (tmp :=:
             rep) )) = 1);
END_ENTITY; -- analysis_item_within_representation

ENTITY analysis_message
  SUPERTYPE OF (ONEOF (whole_model_analysis_message,
      element_analysis_message,node_analysis_message,
      element_group_analysis_message))
  SUBTYPE OF (state_definition);
    level        : message_level;
    message_text : text;
END_ENTITY; -- analysis_message

ENTITY analysis_step
  SUPERTYPE OF (ONEOF (control_analysis_step,result_analysis_step));
    analysis_control : control;
END_ENTITY; -- analysis_step

ENTITY application_context;
    application : text;
  INVERSE
    context_elements : SET [1:?] OF application_context_element FOR
                       frame_of_reference;
END_ENTITY; -- application_context

ENTITY application_context_element
  SUPERTYPE OF (ONEOF (product_context,product_definition_context,
      product_concept_context));
    name               : label;
    frame_of_reference : application_context;
END_ENTITY; -- application_context_element

ENTITY application_protocol_definition;
    status                                    : label;
    application_interpreted_model_schema_name : label;
    application_protocol_year                 : year_number;
    application                               : application_context;
END_ENTITY; -- application_protocol_definition

ENTITY applied_action_assignment
  SUBTYPE OF (action_assignment);
    items : SET [1:?] OF action_item;
END_ENTITY; -- applied_action_assignment

ENTITY applied_action_request_assignment
  SUBTYPE OF (action_request_assignment);
    items : SET [1:?] OF action_request_item;
END_ENTITY; -- applied_action_request_assignment

ENTITY applied_approval_assignment
  SUBTYPE OF (approval_assignment);
    items : SET [1:?] OF approval_item;
END_ENTITY; -- applied_approval_assignment

ENTITY applied_certification_assignment
  SUBTYPE OF (certification_assignment);
    items : SET [1:?] OF certification_item;
```

```
END_ENTITY; -- applied_certification_assignment

ENTITY applied_contract_assignment
  SUBTYPE OF (contract_assignment);
    items : SET [1:?] OF contract_item;
END_ENTITY; -- applied_contract_assignment

ENTITY applied_date_and_time_assignment
  SUBTYPE OF (date_and_time_assignment);
    items : SET [1:?] OF date_time_item;
  WHERE
    wr1: applied_date_time_correlation(SELF);
END_ENTITY; -- applied_date_and_time_assignment

ENTITY applied_date_assignment
  SUBTYPE OF (date_assignment);
    items : SET [1:?] OF date_item;
  WHERE
    wr1: applied_date_correlation(SELF);
END_ENTITY; -- applied_date_assignment

ENTITY applied_document_reference
  SUBTYPE OF (document_reference);
    items : SET [1:?] OF document_reference_item;
END_ENTITY; -- applied_document_reference

ENTITY applied_name_assignment
  SUBTYPE OF (name_assignment);
    items : SET [1:?] OF name_item;
END_ENTITY; -- applied_name_assignment

ENTITY applied_organization_assignment
  SUBTYPE OF (organization_assignment);
    items : SET [1:?] OF organization_item;
  WHERE
    wr1: applied_organization_correlation(SELF);
END_ENTITY; -- applied_organization_assignment

ENTITY applied_person_and_organization_assignment
  SUBTYPE OF (person_and_organization_assignment);
    items : SET [1:?] OF person_organization_item;
  WHERE
    wr1: applied_person_and_organization_correlation(SELF);
END_ENTITY; -- applied_person_and_organization_assignment

ENTITY applied_security_classification_assignment
  SUBTYPE OF (security_classification_assignment);
    items : SET [1:?] OF security_classification_item;
END_ENTITY; -- applied_security_classification_assignment

ENTITY approval;
    status : approval_status;
    level  : label;
END_ENTITY; -- approval

ENTITY approval_assignment
  ABSTRACT SUPERTYPE;
    assigned_approval : approval;
END_ENTITY; -- approval_assignment
```



```
ENTITY approval_date_time;
    date_time      : date_time_select;
    dated_approval : approval;
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_relationship;
    name              : label;
    description       : text;
    relating_approval : approval;
    related_approval  : approval;
END_ENTITY; -- approval_relationship

ENTITY approval_role;
    role : label;
END_ENTITY; -- approval_role

ENTITY approval_status;
    name : label;
END_ENTITY; -- approval_status

ENTITY arbitrary_volume_2d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
    orientation : direction;
  WHERE
    wr1: (SELF\geometric_representation_item.dim = 2);
END_ENTITY; -- arbitrary_volume_2d_element_coordinate_system

ENTITY arbitrary_volume_3d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
    coordinate_system : fea_axis2_placement_3d;
END_ENTITY; -- arbitrary_volume_3d_element_coordinate_system

ENTITY area_measure_with_unit
  SUBTYPE OF (measure_with_unit);
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.AREA_UNIT' IN TYPEOF(SELF\
             measure_with_unit.unit_component));
END_ENTITY; -- area_measure_with_unit

ENTITY area_unit
  SUBTYPE OF (named_unit);
  WHERE
    wr1: ((SELF\named_unit.dimensions.length_exponent = 2) 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; -- area_unit

ENTITY assembly_component_usage
  SUPERTYPE OF (ONEOF (next_assembly_usage_occurrence,
      specified_higher_usage_occurrence,promissory_usage_occurrence))
```

```
  SUBTYPE OF (product_definition_usage);
    reference_designator : OPTIONAL identifier;
END_ENTITY; -- assembly_component_usage

ENTITY assembly_component_usage_substitute;
    name       : label;
    definition : text;
    base       : assembly_component_usage;
    substitute : assembly_component_usage;
  UNIQUE
    ur1 : base, substitute;
  WHERE
    wr1: (base.relating_product_definition :=: substitute.
             relating_product_definition);
    wr2: (base :<>: substitute);
END_ENTITY; -- assembly_component_usage_substitute

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 axisymmetric_2d_element_property;
    angle : plane_angle_measure;
END_ENTITY; -- axisymmetric_2d_element_property

ENTITY axisymmetric_curve_2d_element_descriptor
  SUBTYPE OF (element_descriptor);
    purpose : SET [1:?] OF SET [1:?] OF curve_element_purpose;
END_ENTITY; -- axisymmetric_curve_2d_element_descriptor

ENTITY axisymmetric_curve_2d_element_representation
  SUBTYPE OF (element_representation);
    model_ref          : fea_model_2d;
```



```
    element_descriptor : axisymmetric_curve_2d_element_descriptor;
    property           : curve_2d_element_property;
    angle_property     : axisymmetric_2d_element_property;
    material           : element_material;
  UNIQUE
    ur1 : model_ref, name;
  WHERE
    wr1: (model_ref.type_of_2d_analysis = axisymmetric);
    wr2: (SIZEOF(QUERY ( item <* SELF\representation.items | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'CURVE_2D_ELEMENT_COORDINATE_SYSTEM') IN TYPEOF(item)) )) =
             1);
    wr3: (SIZEOF(QUERY ( item1 <* material.properties | (SIZEOF(
             QUERY ( item2 <* item1\property_definition_representation.
             used_representation.items | (SIZEOF([
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_LINEAR_ELASTICITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MASS_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_AREA_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MOISTURE_ABSORPTION'] *
              TYPEOF(item2)) = 1) )) = 1) )) >= 1);
    wr4: ('STRUCTURAL_ANALYSIS_DESIGN.PARAMETRIC_REPRESENTATION_CONTEXT'
             IN TYPEOF(SELF\representation.context_of_items));
    fu1: required_0d_nodes(SELF\element_representation.node_list);
END_ENTITY; -- axisymmetric_curve_2d_element_representation

ENTITY axisymmetric_surface_2d_element_descriptor
  SUBTYPE OF (element_descriptor);
    purpose : SET [1:?] OF SET [1:?] OF surface_element_purpose;
END_ENTITY; -- axisymmetric_surface_2d_element_descriptor

ENTITY axisymmetric_surface_2d_element_representation
  SUBTYPE OF (element_representation);
    model_ref          : fea_model_2d;
    element_descriptor : axisymmetric_surface_2d_element_descriptor;
    property           : surface_element_property;
    angle_property     : axisymmetric_2d_element_property;
    material           : element_material;
  UNIQUE
    ur1 : model_ref, name;
  WHERE
    wr1: (model_ref.type_of_2d_analysis = axisymmetric);
    wr2: ((SIZEOF(QUERY ( item <* SELF\representation.items | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'PARAMETRIC_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM') IN
             TYPEOF(item)) )) + SIZEOF(QUERY ( item <* SELF\
             representation.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
             'ALIGNED_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM') IN TYPEOF(
             item)) ))) = 1);
    wr3: (SIZEOF(QUERY ( item1 <* material.properties | (SIZEOF(
             QUERY ( item2 <* item1\property_definition_representation.
             used_representation.items | (SIZEOF([
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_LINEAR_ELASTICITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MASS_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_AREA_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
```

```
              'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
              'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MOISTURE_ABSORPTION',
              'STRUCTURAL_ANALYSIS_DESIGN.' +
              'FEA_SHELL_MEMBRANE_STIFFNESS','STRUCTURAL_ANALYSIS_DESIGN.'
              + 'FEA_SHELL_BENDING_STIFFNESS',
              'STRUCTURAL_ANALYSIS_DESIGN.' +
              'FEA_SHELL_MEMBRANE_BENDING_COUPLING_STIFFNESS',
              'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_SHELL_SHEAR_STIFFNESS']
              * TYPEOF(item2)) = 1) )) = 1) )) >= 1);
    wr4: ('STRUCTURAL_ANALYSIS_DESIGN.PARAMETRIC_REPRESENTATION_CONTEXT'
              IN TYPEOF(SELF\representation.context_of_items));
    fu1: required_1d_nodes(SELF\element_representation.node_list,
              element_descriptor\element_descriptor.topology_order);
END_ENTITY; -- axisymmetric_surface_2d_element_representation

ENTITY axisymmetric_volume_2d_element_descriptor
  SUBTYPE OF (element_descriptor);
    purpose : SET [1:?] OF SET [1:?] OF volume_element_purpose;
    shape   : element_2d_shape;
END_ENTITY; -- axisymmetric_volume_2d_element_descriptor

ENTITY axisymmetric_volume_2d_element_representation
  SUBTYPE OF (element_representation);
    model_ref          : fea_model_2d;
    element_descriptor : axisymmetric_volume_2d_element_descriptor;
    angle_property     : axisymmetric_2d_element_property;
    material           : element_material;
  UNIQUE
    ur1 : model_ref, name;
  WHERE
    wr1: (model_ref.type_of_2d_analysis = axisymmetric);
    wr2: ((SIZEOF(QUERY ( item <* SELF\representation.items | ((
              'STRUCTURAL_ANALYSIS_DESIGN.' +
              'PARAMETRIC_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM') IN TYPEOF(
              item)) )) + SIZEOF(QUERY ( item <* SELF\representation.items
               | (('STRUCTURAL_ANALYSIS_DESIGN.' +
              'ARBITRARY_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM') IN TYPEOF(
              item)) ))) = 1);
    wr3: (SIZEOF(QUERY ( item1 <* material.properties | (SIZEOF(
              QUERY ( item2 <* item1\property_definition_representation.
              used_representation.items | (SIZEOF([
              'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_LINEAR_ELASTICITY',
              'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MASS_DENSITY',
              'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_AREA_DENSITY',
              'STRUCTURAL_ANALYSIS_DESIGN.' +
              'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
              'STRUCTURAL_ANALYSIS_DESIGN.' +
              'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
              'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MOISTURE_ABSORPTION'] *
               TYPEOF(item2)) = 1) )) = 1) )) >= 1);
    wr4: ('STRUCTURAL_ANALYSIS_DESIGN.PARAMETRIC_REPRESENTATION_CONTEXT'
              IN TYPEOF(SELF\representation.context_of_items));
    fu1: required_2d_nodes(SELF\element_representation.node_list,
              element_descriptor.shape,element_descriptor\
              element_descriptor.topology_order);
END_ENTITY; -- axisymmetric_volume_2d_element_representation

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: (('STRUCTURAL_ANALYSIS_DESIGN.UNIFORM_CURVE' IN TYPEOF(SELF))
           OR ('STRUCTURAL_ANALYSIS_DESIGN.QUASI_UNIFORM_CURVE' IN
           TYPEOF(SELF)) OR ('STRUCTURAL_ANALYSIS_DESIGN.BEZIER_CURVE'
           IN TYPEOF(SELF)) OR (
           'STRUCTURAL_ANALYSIS_DESIGN.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: (('STRUCTURAL_ANALYSIS_DESIGN.UNIFORM_SURFACE' IN TYPEOF(SELF))
           OR ('STRUCTURAL_ANALYSIS_DESIGN.QUASI_UNIFORM_SURFACE' IN
           TYPEOF(SELF)) OR (
           'STRUCTURAL_ANALYSIS_DESIGN.BEZIER_SURFACE' IN TYPEOF(SELF))
           OR ('STRUCTURAL_ANALYSIS_DESIGN.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 beveled_sheet_representation
  SUBTYPE OF (shape_representation);
END_ENTITY; -- beveled_sheet_representation

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 binary_generic_expression
  ABSTRACT SUPERTYPE
  SUBTYPE OF (generic_expression);
    operands : LIST [2:2] OF generic_expression;
END_ENTITY; -- binary_generic_expression

ENTITY binary_numeric_expression
  ABSTRACT SUPERTYPE OF (ONEOF (minus_expression,div_expression))
  SUBTYPE OF (numeric_expression, binary_generic_expression);
    SELF\binary_generic_expression.operands : LIST [2:2] OF
                                                  numeric_expression;
END_ENTITY; -- binary_numeric_expression

ENTITY boolean_expression
  ABSTRACT SUPERTYPE OF (ONEOF (multiple_arity_boolean_expression,
      comparison_expression,interval_expression))
  SUBTYPE OF (expression);
END_ENTITY; -- boolean_expression

ENTITY boundary_curve
  SUBTYPE OF (composite_curve_on_surface);
  WHERE
    wr1: SELF\composite_curve.closed_curve;
END_ENTITY; -- boundary_curve

ENTITY bounded_curve
  SUPERTYPE OF (ONEOF (polyline,b_spline_curve,trimmed_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: ('STRUCTURAL_ANALYSIS_DESIGN.BOUNDED_CURVE' IN TYPEOF(SELF\
             pcurve.reference_to_curve.items[1]));
END_ENTITY; -- bounded_pcurve

ENTITY bounded_surface
  SUPERTYPE OF (ONEOF (b_spline_surface,rectangular_trimmed_surface,
      curve_bounded_surface,rectangular_composite_surface))
  SUBTYPE OF (surface);
END_ENTITY; -- bounded_surface

ENTITY bounded_surface_curve
  SUBTYPE OF (surface_curve, bounded_curve);
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.BOUNDED_CURVE' IN TYPEOF(SELF\
             surface_curve.curve_3d));
END_ENTITY; -- bounded_surface_curve

ENTITY brep_with_voids
  SUBTYPE OF (manifold_solid_brep);
    voids : SET [1:?] OF oriented_closed_shell;
END_ENTITY; -- brep_with_voids

ENTITY calculated_state
  SUBTYPE OF (state);
END_ENTITY; -- calculated_state

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
  SUPERTYPE OF (ONEOF (cylindrical_point,spherical_point))
  SUBTYPE OF (point);
    coordinates : LIST [1:3] OF length_measure;
END_ENTITY; -- cartesian_point

ENTITY cartesian_transformation_operator
  SUPERTYPE OF (ONEOF (cartesian_transformation_operator_2d,
      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_2d
  SUBTYPE OF (cartesian_transformation_operator);
  DERIVE
    u : LIST [2:2] OF direction := base_axis(2,SELF\
           cartesian_transformation_operator.axis1,SELF\
           cartesian_transformation_operator.axis2,?);
  WHERE
    wr1: (SELF\geometric_representation_item.dim = 2);
END_ENTITY; -- cartesian_transformation_operator_2d

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 certification;
    name    : label;
    purpose : text;
    kind    : certification_type;
END_ENTITY; -- certification

ENTITY certification_assignment
  ABSTRACT SUPERTYPE;
    assigned_certification : certification;
END_ENTITY; -- certification_assignment

ENTITY certification_type;
    description : label;
END_ENTITY; -- certification_type

ENTITY characterized_object;
    name        : label;
    description : text;
END_ENTITY; -- characterized_object

ENTITY circle
  SUBTYPE OF (conic);
    radius : positive_length_measure;
END_ENTITY; -- circle

ENTITY closed_shell
  SUBTYPE OF (connected_face_set);
END_ENTITY; -- closed_shell

ENTITY comparison_equal
  SUBTYPE OF (comparison_expression);
END_ENTITY; -- comparison_equal

ENTITY comparison_expression
  ABSTRACT SUPERTYPE OF (comparison_equal)
  SUBTYPE OF (boolean_expression, binary_generic_expression);
    SELF\binary_generic_expression.operands : LIST [2:2] OF expression;
  WHERE
    wr1:  ((('STRUCTURAL_ANALYSIS_DESIGN.NUMERIC_EXPRESSION' IN TYPEOF(
             SELF\binary_generic_expression.operands[1])) AND (
```



```
            'STRUCTURAL_ANALYSIS_DESIGN.NUMERIC_EXPRESSION' IN TYPEOF(
            SELF\binary_generic_expression.operands[2]))) OR ((
            'STRUCTURAL_ANALYSIS_DESIGN.BOOLEAN_EXPRESSION' IN TYPEOF(
            SELF\binary_generic_expression.operands[1])) AND (
            'STRUCTURAL_ANALYSIS_DESIGN.BOOLEAN_EXPRESSION' IN TYPEOF(
            SELF\binary_generic_expression.operands[2]))) OR ((
            'STRUCTURAL_ANALYSIS_DESIGN.STRING_EXPRESSION' IN TYPEOF(
            SELF\binary_generic_expression.operands[1])) AND (
            'STRUCTURAL_ANALYSIS_DESIGN.STRING_EXPRESSION' IN TYPEOF(
            SELF\binary_generic_expression.operands[2]))));
END_ENTITY; -- comparison_expression

ENTITY composite_assembly_definition
  SUBTYPE OF (product_definition);
  WHERE
    wr1: (SIZEOF(QUERY ( pdr <* USEDIN(SELF,
            'STRUCTURAL_ANALYSIS_DESIGN.' +
            'PRODUCT_DEFINITION_RELATIONSHIP.' +
            'RELATING_PRODUCT_DEFINITION') | ((
            'STRUCTURAL_ANALYSIS_DESIGN.' +
            'NEXT_ASSEMBLY_USAGE_OCCURRENCE') IN TYPEOF(pdr)) )) = 1);
END_ENTITY; -- composite_assembly_definition

ENTITY composite_assembly_sequence_definition
  SUBTYPE OF (product_definition);
  WHERE
    wr1: (SIZEOF(QUERY ( pdr <* USEDIN(SELF,
            'STRUCTURAL_ANALYSIS_DESIGN.' +
            'PRODUCT_DEFINITION_RELATIONSHIP.' +
            'RELATING_PRODUCT_DEFINITION') | ((
            'STRUCTURAL_ANALYSIS_DESIGN.' +
            'NEXT_ASSEMBLY_USAGE_OCCURRENCE') IN TYPEOF(pdr)) )) > 0);
END_ENTITY; -- composite_assembly_sequence_definition

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
  SUPERTYPE OF (boundary_curve)
  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: ('STRUCTURAL_ANALYSIS_DESIGN.BOUNDED_CURVE' IN TYPEOF(
             parent_curve));
END_ENTITY; -- composite_curve_segment

ENTITY composite_sheet_representation
  SUBTYPE OF (shape_representation);
  WHERE
    wr1: (SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.' +
             'GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'MANIFOLD_SURFACE_SHAPE_REPRESENTATION'] * TYPEOF(SELF)) = 1);
END_ENTITY; -- composite_sheet_representation

ENTITY configuration_design;
    configuration : configuration_item;
    design        : product_definition_formation;
  UNIQUE
    ur1 : configuration, design;
END_ENTITY; -- configuration_design

ENTITY configuration_effectivity
  SUBTYPE OF (product_definition_effectivity);
    configuration : configuration_design;
  UNIQUE
    ur1 : configuration, usage, id;
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.PRODUCT_DEFINITION_USAGE' IN
             TYPEOF(SELF\product_definition_effectivity.usage));
END_ENTITY; -- configuration_effectivity

ENTITY configuration_item;
    id           : identifier;
    name         : label;
    description  : OPTIONAL text;
    item_concept : product_concept;
    purpose      : OPTIONAL label;
  UNIQUE
    ur1 : id;
END_ENTITY; -- configuration_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 constant_surface_3d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
    axis  : INTEGER;
    angle : plane_angle_measure;
  WHERE
    wr1: ((axis >= 1) AND (axis <= 2));
END_ENTITY; -- constant_surface_3d_element_coordinate_system

ENTITY constraint_element
  SUPERTYPE OF (ONEOF (single_point_constraint_element,
      linear_constraint_equation_element,nodal_dof_reduction,
      point_constraint,curve_constraint,surface_constraint,
      solid_constraint));
    element_id : identifier;
    steps      : SET [1:?] OF control_analysis_step;
END_ENTITY; -- constraint_element

ENTITY context_dependent_shape_representation;
    representation_relation      : shape_representation_relationship;
    represented_product_relation : product_definition_shape;
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.PRODUCT_DEFINITION_RELATIONSHIP'
             IN TYPEOF(SELF.represented_product_relation.definition));
END_ENTITY; -- context_dependent_shape_representation

ENTITY context_dependent_unit
  SUBTYPE OF (named_unit);
    name : label;
END_ENTITY; -- context_dependent_unit

ENTITY contract;
    name    : label;
    purpose : text;
    kind    : contract_type;
END_ENTITY; -- contract

ENTITY contract_assignment
  ABSTRACT SUPERTYPE;
    assigned_contract : contract;
END_ENTITY; -- contract_assignment

ENTITY contract_type;
    description : label;
END_ENTITY; -- contract_type

ENTITY control;
    model_ref              : fea_model;
    control_id             : identifier;
    creating_software      : text;
    description            : text;
```

```
    user_defined_control   : SET [1:?] OF text;
    intended_analysis_code : SET [1:?] OF text;
  UNIQUE
    ur1 : model_ref, control_id;
END_ENTITY; -- control

ENTITY control_analysis_step
  SUPERTYPE OF (ONEOF (control_linear_static_analysis_step,
      control_linear_modes_and_frequencies_analysis_step))
  SUBTYPE OF (analysis_step);
    step_id       : identifier;
    sequence      : INTEGER;
    initial_state : state;
    description   : text;
  UNIQUE
    ur1 : analysis_control, sequence;
    ur2 : analysis_control, step_id;
END_ENTITY; -- control_analysis_step

ENTITY control_linear_modes_and_frequencies_analysis_step
  SUBTYPE OF (control_analysis_step);
    process         : control_linear_modes_and_frequencies_process;
    number_of_modes : count_measure;
    frequency_range : ARRAY [1:2] OF context_dependent_measure;
END_ENTITY; -- control_linear_modes_and_frequencies_analysis_step

ENTITY control_linear_modes_and_frequencies_process
  SUBTYPE OF (control_process);
    final_input_state : state;
END_ENTITY; -- control_linear_modes_and_frequencies_process

ENTITY control_linear_static_analysis_step
  SUBTYPE OF (control_analysis_step);
    process : control_linear_static_load_increment_process;
END_ENTITY; -- control_linear_static_analysis_step

ENTITY control_linear_static_analysis_step_with_harmonic
  SUBTYPE OF (control_linear_static_analysis_step);
    symmetry : cylindrical_symmetry_control;
END_ENTITY; -- control_linear_static_analysis_step_with_harmonic

ENTITY control_linear_static_load_increment_process
  SUBTYPE OF (control_process);
    final_input_state : state;
END_ENTITY; -- control_linear_static_load_increment_process

ENTITY control_process
  SUPERTYPE OF (ONEOF (control_linear_static_load_increment_process,
      control_linear_modes_and_frequencies_process));
    process_id  : identifier;
    description : text;
END_ENTITY; -- control_process

ENTITY control_result_relationship;
    control : control_analysis_step;
    result  : result_analysis_step;
END_ENTITY; -- control_result_relationship

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   : hour_in_day;
    minute_offset : OPTIONAL minute_in_hour;
    sense         : ahead_or_behind;
END_ENTITY; -- coordinated_universal_time_offset

ENTITY curve
  SUPERTYPE OF (ONEOF (line,conic,pcurve,surface_curve,offset_curve_2d,
      offset_curve_3d,curve_replica))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- curve

ENTITY curve_2d_element_basis;
    descriptor : curve_2d_element_descriptor;
    variable   : curve_element_variable;
END_ENTITY; -- curve_2d_element_basis

ENTITY curve_2d_element_constant_specified_variable_value
  SUBTYPE OF (curve_2d_element_field_variable_definition);
    simple_value      : field_value;
    variable          : surface_element_variable;
    coordinate_system : OPTIONAL curve_2d_element_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- curve_2d_element_constant_specified_variable_value

ENTITY curve_2d_element_constant_specified_volume_variable_value
  SUBTYPE OF (curve_2d_element_field_variable_definition);
    simple_value      : field_value;
    variable          : volume_variable;
    coordinate_system : OPTIONAL curve_2d_element_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- curve_2d_element_constant_specified_volume_variable_value

ENTITY curve_2d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
    orientation : direction;
  WHERE
    wr1: (SELF\geometric_representation_item.dim = 2);
END_ENTITY; -- curve_2d_element_coordinate_system

ENTITY curve_2d_element_field_variable_definition
  SUPERTYPE OF (ONEOF (
      curve_2d_element_location_point_volume_variable_values,
      curve_2d_element_location_point_variable_values,
      curve_2d_whole_element_variable_value,
      curve_2d_element_constant_specified_variable_value,
      curve_2d_element_constant_specified_volume_variable_value))
  SUBTYPE OF (field_variable_element_definition);
```

```
    element : curve_2d_element_output_reference;
END_ENTITY; -- curve_2d_element_field_variable_definition

ENTITY curve_2d_element_group
  SUBTYPE OF (element_group);
  WHERE
    wr1: (SIZEOF(QUERY ( item <* elements | ((NOT ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'AXISYMMETRIC_CURVE_2D_ELEMENT_REPRESENTATION') IN TYPEOF(
             item))) AND (NOT (('STRUCTURAL_ANALYSIS_DESIGN.' +
             'PLANE_CURVE_2D_ELEMENT_REPRESENTATION') IN TYPEOF(item)))) ))
             = 0);
END_ENTITY; -- curve_2d_element_group

ENTITY curve_2d_element_integrated_matrix;
    descriptor              : curve_2d_element_descriptor;
    property_type           : curve_matrix_property_type;
    integration_description : text;
END_ENTITY; -- curve_2d_element_integrated_matrix

ENTITY curve_2d_element_integrated_matrix_with_definition
  SUBTYPE OF (curve_2d_element_integrated_matrix);
    integration_definition : curve_2d_element_integration;
END_ENTITY; -- curve_2d_element_integrated_matrix_with_definition

ENTITY curve_2d_element_integration;
    section : LIST [1:?] OF curve_section_element_location;
END_ENTITY; -- curve_2d_element_integration

ENTITY curve_2d_element_location_point_variable_values
  SUBTYPE OF (curve_2d_element_field_variable_definition);
    basis                : BOOLEAN;
    values_and_locations : SET [1:?] OF
                               curve_2d_element_value_and_location;
    variable             : curve_element_variable;
  WHERE
    wr1: consistent_set_values(values_and_locations,variable);
    wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
             SELF\state_definition.defined_state));
END_ENTITY; -- curve_2d_element_location_point_variable_values

ENTITY curve_2d_element_location_point_volume_variable_values
  SUBTYPE OF (curve_2d_element_field_variable_definition);
    basis                : BOOLEAN;
    values_and_locations : SET [1:?] OF
curve_2d_element_value_and_volume_location;
    variable             : curve_element_variable;
  WHERE
    wr1: consistent_set_values(values_and_locations,variable);
    wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
             SELF\state_definition.defined_state));
END_ENTITY; -- curve_2d_element_location_point_volume_variable_values

ENTITY curve_2d_element_property;
    property_id : identifier;
    description : text;
    section     : curve_element_section_definition;
END_ENTITY; -- curve_2d_element_property

ENTITY curve_2d_element_value_and_location;
    simple_value      : field_value;
```



```
    location           : curve_section_element_location;
    coordinate_system : OPTIONAL curve_2d_state_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
END_ENTITY; -- curve_2d_element_value_and_location

ENTITY curve_2d_element_value_and_volume_location;
    simple_value       : field_value;
    location           : curve_volume_element_location;
    coordinate_system : OPTIONAL curve_2d_state_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
END_ENTITY; -- curve_2d_element_value_and_volume_location

ENTITY curve_2d_node_field_aggregated_variable_values
  SUBTYPE OF (curve_2d_node_field_variable_definition);
    simple_value       : field_value;
    variable           : volume_aggregated_variable;
    coordinate_system : OPTIONAL curve_2d_state_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- curve_2d_node_field_aggregated_variable_values

ENTITY curve_2d_node_field_section_variable_values
  SUBTYPE OF (curve_2d_node_field_variable_definition);
    simple_value       : field_value;
    variable           : curve_element_variable;
    location           : curve_section_element_location;
    coordinate_system : OPTIONAL curve_2d_state_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- curve_2d_node_field_section_variable_values

ENTITY curve_2d_node_field_variable_definition
  SUPERTYPE OF (ONEOF (curve_2d_node_field_section_variable_values,
      curve_2d_node_field_aggregated_variable_values))
  SUBTYPE OF (field_variable_node_definition);
END_ENTITY; -- curve_2d_node_field_variable_definition

ENTITY curve_2d_substructure_element_reference;
    substructure_element_ref : substructure_element_representation;
    element_ref              : curve_2d_element_representation;
END_ENTITY; -- curve_2d_substructure_element_reference

ENTITY curve_2d_whole_element_variable_value
  SUBTYPE OF (curve_2d_element_field_variable_definition);
    simple_value       : field_value;
    variable           : volume_aggregated_variable;
    coordinate_system : OPTIONAL curve_2d_element_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
```

```
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- curve_2d_whole_element_variable_value

ENTITY curve_3d_element_basis;
    descriptor               : curve_3d_element_descriptor;
    variable                 : curve_element_variable;
    variable_order           : element_order;
    variable_shape_function : shape_function;
    evaluation_points        : LIST [1:?] OF curve_element_location;
END_ENTITY; -- curve_3d_element_basis

ENTITY curve_3d_element_constant_specified_variable_value
  SUBTYPE OF (curve_3d_element_field_variable_definition);
    simple_value      : field_value;
    variable          : curve_element_variable;
    coordinate_system : OPTIONAL curve_3d_element_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- curve_3d_element_constant_specified_variable_value

ENTITY curve_3d_element_constant_specified_volume_variable_value
  SUBTYPE OF (curve_3d_element_field_variable_definition);
    simple_value      : field_value;
    variable          : volume_variable;
    coordinate_system : OPTIONAL curve_3d_element_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- curve_3d_element_constant_specified_volume_variable_value

ENTITY curve_3d_element_descriptor
  SUBTYPE OF (element_descriptor);
    purpose : SET [1:?] OF SET [1:?] OF curve_element_purpose;
END_ENTITY; -- curve_3d_element_descriptor

ENTITY curve_3d_element_field_variable_definition
  SUPERTYPE OF (ONEOF (
      curve_3d_element_location_point_volume_variable_values,
      curve_3d_element_location_point_variable_values,
      curve_3d_whole_element_variable_value,
      curve_3d_element_constant_specified_variable_value,
      curve_3d_element_constant_specified_volume_variable_value,
      curve_3d_element_nodal_specified_variable_values))
  SUBTYPE OF (field_variable_element_definition);
    element : curve_3d_element_output_reference;
END_ENTITY; -- curve_3d_element_field_variable_definition

ENTITY curve_3d_element_group
  SUBTYPE OF (element_group);
  WHERE
    wr1: (SIZEOF(QUERY ( item <* elements | (NOT ((
```



```
                'STRUCTURAL_ANALYSIS_DESIGN.' +
                'CURVE_3D_ELEMENT_REPRESENTATION') IN TYPEOF(item))) )) = 0);
  END_ENTITY; -- curve_3d_element_group

  ENTITY curve_3d_element_integrated_matrix;
      descriptor             : curve_3d_element_descriptor;
      property_type          : curve_matrix_property_type;
      integration_description : text;
  END_ENTITY; -- curve_3d_element_integrated_matrix

  ENTITY curve_3d_element_integrated_matrix_with_definition
    SUBTYPE OF (curve_3d_element_integrated_matrix);
      integration_definition : curve_3d_element_integration;
  END_ENTITY; -- curve_3d_element_integrated_matrix_with_definition

  ENTITY curve_3d_element_integration;
      element_length : curve_3d_element_length_integration;
      section        : curve_section_integration_explicit;
  END_ENTITY; -- curve_3d_element_integration

  ENTITY curve_3d_element_length_integration_explicit;
      integration_positions_and_weights : SET [1:?] OF
curve_3d_element_position_weight;
  END_ENTITY; -- curve_3d_element_length_integration_explicit

  ENTITY curve_3d_element_length_integration_rule;
      integration_method : integration_rule;
      integration_order  : INTEGER;
  END_ENTITY; -- curve_3d_element_length_integration_rule

  ENTITY curve_3d_element_location_point_variable_values
    SUBTYPE OF (curve_3d_element_field_variable_definition);
      basis                : BOOLEAN;
      values_and_locations : SET [1:?] OF
                                 curve_3d_element_value_and_location;
      variable             : curve_element_variable;
    WHERE
      wr1: consistent_set_values(values_and_locations,variable);
      wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
                SELF\state_definition.defined_state));
  END_ENTITY; -- curve_3d_element_location_point_variable_values

  ENTITY curve_3d_element_location_point_volume_variable_values
    SUBTYPE OF (curve_3d_element_field_variable_definition);
      basis                : BOOLEAN;
      values_and_locations : SET [1:?] OF
curve_3d_element_value_and_volume_location;
      variable             : volume_variable;
    WHERE
      wr1: consistent_set_values(values_and_locations,variable);
      wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
                SELF\state_definition.defined_state));
  END_ENTITY; -- curve_3d_element_location_point_volume_variable_values

  ENTITY curve_3d_element_nodal_specified_variable_values
    SUBTYPE OF (curve_3d_element_field_variable_definition);
      values                 : LIST [1:?] OF field_value;
      additional_node_values : BOOLEAN;
      variable               : curve_element_variable;
    WHERE
      wr1: consistent_list_values(values,variable);
```

```
    wr2: appropriate_list_value_existence(values,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- curve_3d_element_nodal_specified_variable_values

ENTITY curve_3d_element_position_weight;
    integration_position : curve_volume_element_location;
    integration_weight   : context_dependent_measure;
END_ENTITY; -- curve_3d_element_position_weight

ENTITY curve_3d_element_property;
    property_id          : identifier;
    description          : text;
    interval_definitions : LIST [1:?] OF curve_element_interval;
    end_offsets          : ARRAY [1:2] OF curve_element_end_offset;
    end_releases         : ARRAY [1:2] OF curve_element_end_release;
END_ENTITY; -- curve_3d_element_property

ENTITY curve_3d_element_representation
  SUBTYPE OF (element_representation);
    model_ref          : fea_model_3d;
    element_descriptor : curve_3d_element_descriptor;
    property           : curve_3d_element_property;
    material           : element_material;
  UNIQUE
    ur1 : model_ref, name;
  WHERE
    wr1: ((SIZEOF(QUERY ( item <* SELF\representation.items | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'PARAMETRIC_CURVE_3D_ELEMENT_COORDINATE_SYSTEM') IN TYPEOF(
             item)) )) + SIZEOF(QUERY ( item <* SELF\representation.items
              | (('STRUCTURAL_ANALYSIS_DESIGN.' +
             'ALIGNED_CURVE_3D_ELEMENT_COORDINATE_SYSTEM') IN TYPEOF(item))
)))
             = 1);
    wr2: (SIZEOF(QUERY ( item1 <* material.properties | (SIZEOF(
             QUERY ( item2 <* item1\property_definition_representation.
             used_representation.items | (SIZEOF([
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_LINEAR_ELASTICITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MASS_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_AREA_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MOISTURE_ABSORPTION'] *
              TYPEOF(item2)) = 1) )) = 1) )) >= 1);
    wr3: ('STRUCTURAL_ANALYSIS_DESIGN.PARAMETRIC_REPRESENTATION_CONTEXT'
             IN TYPEOF(SELF\representation.context_of_items));
    fu1: required_1d_nodes(SELF\element_representation.node_list,
             element_descriptor\element_descriptor.topology_order);
END_ENTITY; -- curve_3d_element_representation

ENTITY curve_3d_element_value_and_location;
    simple_value      : field_value;
    location          : curve_element_location;
    coordinate_system : OPTIONAL curve_3d_state_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
END_ENTITY; -- curve_3d_element_value_and_location
```



```
ENTITY curve_3d_element_value_and_volume_location;
    simple_value      : field_value;
    location          : curve_volume_element_location;
    coordinate_system : OPTIONAL curve_3d_state_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
END_ENTITY; -- curve_3d_element_value_and_volume_location

ENTITY curve_3d_node_field_aggregated_variable_values
  SUBTYPE OF (curve_3d_node_field_variable_definition);
    simple_value      : field_value;
    variable          : volume_aggregated_variable;
    coordinate_system : OPTIONAL curve_3d_state_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- curve_3d_node_field_aggregated_variable_values

ENTITY curve_3d_node_field_section_variable_values
  SUBTYPE OF (curve_3d_node_field_variable_definition);
    simple_value      : field_value;
    variable          : curve_element_variable;
    location          : curve_section_element_location;
    coordinate_system : OPTIONAL curve_3d_state_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- curve_3d_node_field_section_variable_values

ENTITY curve_3d_node_field_variable_definition
  SUPERTYPE OF (ONEOF (curve_3d_node_field_section_variable_values,
      curve_3d_node_field_aggregated_variable_values))
  SUBTYPE OF (field_variable_node_definition);
END_ENTITY; -- curve_3d_node_field_variable_definition

ENTITY curve_3d_substructure_element_reference;
    substructure_element_ref : substructure_element_representation;
    element_ref              : curve_3d_element_representation;
END_ENTITY; -- curve_3d_substructure_element_reference

ENTITY curve_3d_whole_element_variable_value
  SUBTYPE OF (curve_3d_element_field_variable_definition);
    simple_value      : field_value;
    variable          : volume_aggregated_variable;
    coordinate_system : OPTIONAL curve_3d_element_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- curve_3d_whole_element_variable_value
```

```
ENTITY curve_bounded_surface
  SUBTYPE OF (bounded_surface);
    basis_surface  : surface;
    boundaries     : SET [1:?] OF boundary_curve;
    implicit_outer : BOOLEAN;
  WHERE
    wr1: ((NOT implicit_outer) OR (SIZEOF(QUERY ( temp <* boundaries | (
             'STRUCTURAL_ANALYSIS_DESIGN.OUTER_BOUNDARY_CURVE' IN TYPEOF(
             temp)) )) = 0));
    wr2: ((NOT implicit_outer) OR (
             'STRUCTURAL_ANALYSIS_DESIGN.BOUNDED_SURFACE' IN TYPEOF(
             basis_surface)));
    wr3: (SIZEOF(QUERY ( temp <* boundaries | (
             'STRUCTURAL_ANALYSIS_DESIGN.OUTER_BOUNDARY_CURVE' IN TYPEOF(
             temp)) )) <= 1);
    wr4: (SIZEOF(QUERY ( temp <* boundaries | (temp\
             composite_curve_on_surface.basis_surface[1] <> SELF.
             basis_surface) )) = 0);
END_ENTITY; -- curve_bounded_surface

ENTITY curve_constraint
  SUBTYPE OF (constraint_element);
    required_curve             : analysis_item_within_representation;
    coordinate_system          : fea_axis2_placement_3d;
    freedoms_and_coefficients  : SET [1:?] OF freedom_and_coefficient;
    description                : text;
  WHERE
    wr1: (('STRUCTURAL_ANALYSIS_DESIGN.CURVE' IN TYPEOF(required_curve.
             item)) OR ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_CURVE' IN
             TYPEOF(required_curve.item)));
END_ENTITY; -- curve_constraint

ENTITY curve_element_end_offset;
    coordinate_system : curve_element_end_coordinate_system;
    offset_vector     : ARRAY [1:3] OF context_dependent_measure;
END_ENTITY; -- curve_element_end_offset

ENTITY curve_element_end_release;
    coordinate_system : curve_element_end_coordinate_system;
    releases          : LIST [1:?] OF curve_element_end_release_packet;
END_ENTITY; -- curve_element_end_release

ENTITY curve_element_end_release_packet;
    release_freedom   : curve_element_freedom;
    release_stiffness : context_dependent_measure;
END_ENTITY; -- curve_element_end_release_packet

ENTITY curve_element_interval
  SUPERTYPE OF (ONEOF (curve_element_interval_constant,
      curve_element_interval_linearly_varying));
    finish_position : curve_element_location;
    eu_angles       : euler_angles;
END_ENTITY; -- curve_element_interval

ENTITY curve_element_interval_constant
  SUBTYPE OF (curve_element_interval);
    section : curve_element_section_definition;
END_ENTITY; -- curve_element_interval_constant
```

```
ENTITY curve_element_interval_linearly_varying
  SUBTYPE OF (curve_element_interval);
    sections : ARRAY [1:2] OF curve_element_section_definition;
END_ENTITY; -- curve_element_interval_linearly_varying

ENTITY curve_element_location;
    coordinate : fea_parametric_point;
END_ENTITY; -- curve_element_location

ENTITY curve_element_section_definition
  SUPERTYPE OF (curve_element_section_derived_definitions);
    description   : text;
    section_angle : plane_angle_measure;
END_ENTITY; -- curve_element_section_definition

ENTITY curve_element_section_derived_definitions
  SUBTYPE OF (curve_element_section_definition);
    cross_sectional_area           : context_dependent_measure;
    shear_area                     : ARRAY [1:2] OF
                                         measure_or_unspecified_value;
    second_moment_of_area          : ARRAY [1:3] OF
                                         context_dependent_measure;
    torsional_constant             : context_dependent_measure;
    warping_constant               : measure_or_unspecified_value;
    location_of_centroid           : ARRAY [1:2] OF
                                         measure_or_unspecified_value;
    location_of_shear_centre       : ARRAY [1:2] OF
                                         measure_or_unspecified_value;
    location_of_non_structural_mass : ARRAY [1:2] OF
                                         measure_or_unspecified_value;
    non_structural_mass            : measure_or_unspecified_value;
    polar_moment                   : measure_or_unspecified_value;
END_ENTITY; -- curve_element_section_derived_definitions

ENTITY curve_freedom_action_definition
  SUBTYPE OF (curve_freedom_and_value_definition);
    action : action_type;
END_ENTITY; -- curve_freedom_action_definition

ENTITY curve_freedom_and_value_definition
  SUPERTYPE OF (ONEOF (curve_freedom_values,
      curve_freedom_action_definition))
  SUBTYPE OF (state_definition);
    required_curve     : analysis_item_within_representation;
    coordinate_system  : fea_axis2_placement_3d;
    degrees_of_freedom : freedoms_list;
    values             : LIST [1:?] OF measure_or_unspecified_value;
  WHERE
    wr1: (SIZEOF(degrees_of_freedom.freedoms) = SIZEOF(values));
    wr2: (('STRUCTURAL_ANALYSIS_DESIGN.CURVE' IN TYPEOF(required_curve.
             item)) OR ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_CURVE' IN
             TYPEOF(required_curve.item)));
END_ENTITY; -- curve_freedom_and_value_definition

ENTITY curve_freedom_values
  SUBTYPE OF (curve_freedom_and_value_definition);
END_ENTITY; -- curve_freedom_values

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 curve_section_element_location;
    offsets : ARRAY [1:2] OF context_dependent_measure;
END_ENTITY; -- curve_section_element_location

ENTITY curve_section_integration_explicit;
    integration_positions : SET [1:?] OF curve_section_element_location;
END_ENTITY; -- curve_section_integration_explicit

ENTITY curve_volume_element_location;
    field_location   : curve_element_location;
    section_location : curve_section_element_location;
END_ENTITY; -- curve_volume_element_location

ENTITY cylindrical_point
  SUBTYPE OF (cartesian_point);
    r     : length_measure;
    theta : plane_angle_measure;
    z     : length_measure;
  DERIVE
    SELF\cartesian_point.coordinates : LIST [1:3] OF length_measure := [
                                     r * COS(theta),r * SIN(theta),z];
  WHERE
    wr1: (r >= 0);
END_ENTITY; -- cylindrical_point

ENTITY cylindrical_surface
  SUBTYPE OF (elementary_surface);
    radius : positive_length_measure;
END_ENTITY; -- cylindrical_surface

ENTITY cylindrical_symmetry_control
  SUBTYPE OF (symmetry_control);
    harmonic : cylindrical_harmonic_number;
    phase    : measure_or_unspecified_value;
END_ENTITY; -- cylindrical_symmetry_control

ENTITY data_environment;
    name        : label;
    description : text;
    elements    : SET [1:?] OF property_definition_representation;
END_ENTITY; -- data_environment

ENTITY data_environment_relationship;
    name                        : label;
    description                 : text;
    relating_data_environment : data_environment;
    related_data_environment  : data_environment;
END_ENTITY; -- data_environment_relationship

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_assignment
  ABSTRACT SUPERTYPE;
    assigned_date : date;
    role          : date_role;
END_ENTITY; -- date_assignment

ENTITY date_role;
    name : label;
END_ENTITY; -- date_role

ENTITY date_time_role;
    name : label;
END_ENTITY; -- date_time_role

ENTITY dated_effectivity
  SUBTYPE OF (effectivity);
    effectivity_start_date : date_and_time;
    effectivity_end_date   : OPTIONAL date_and_time;
END_ENTITY; -- dated_effectivity

ENTITY definitional_representation
  SUBTYPE OF (representation);
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.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: ('STRUCTURAL_ANALYSIS_DESIGN.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;
  WHERE
    wr1: ((SIZEOF(elements) > 1) OR ((SIZEOF(elements) = 1) AND (
```

```
             elements[1].exponent <> 1)));
END_ENTITY; -- derived_unit

ENTITY derived_unit_element;
    unit     : named_unit;
    exponent : REAL;
END_ENTITY; -- derived_unit_element

ENTITY descriptive_representation_item
  SUBTYPE OF (representation_item);
    description : text;
END_ENTITY; -- descriptive_representation_item

ENTITY design_make_from_relationship
  SUBTYPE OF (product_definition_relationship);
END_ENTITY; -- design_make_from_relationship

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 dimensional_size;
    applies_to : shape_aspect;
    name       : label;
  WHERE
    wr1: (applies_to.product_definitional = TRUE);
END_ENTITY; -- dimensional_size

ENTITY directed_action
  SUBTYPE OF (executed_action);
    directive : action_directive;
END_ENTITY; -- directed_action

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 direction_node
  SUBTYPE OF (direction);
    node_1 : node_representation;
    node_2 : node_representation;
  DERIVE
    SELF\direction.direction_ratios : LIST [2:3] OF REAL :=
                                     build_direction_node(node_1,
                                     node_2);
  WHERE
    wr1: (SIZEOF(QUERY ( item <* node_1\representation.items | (
             'STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT' IN TYPEOF(item))
))
             = 1);
    wr2: (SIZEOF(QUERY ( item <* node_2\representation.items | (
```



```
              'STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT' IN TYPEOF(item))
))
              = 1);
      wr3: (NOT ((direction_ratios[1] = 0) AND (direction_ratios[2] = 0)
              AND (direction_ratios[3] = 0)));
  END_ENTITY; -- direction_node

  ENTITY directionally_explicit_element_coefficient;
      property_type : matrix_property_type;
      coefficient   : context_dependent_measure;
  END_ENTITY; -- directionally_explicit_element_coefficient

  ENTITY directionally_explicit_element_coordinate_system_aligned
    SUBTYPE OF (fea_representation_item);
      aligned_system : curve_3d_element_coordinate_system;
  END_ENTITY; -- directionally_explicit_element_coordinate_system_aligned

  ENTITY directionally_explicit_element_coordinate_system_arbitrary
    SUBTYPE OF (fea_representation_item);
      arbitrary_system : fea_axis2_placement_3d;
  END_ENTITY; -- directionally_explicit_element_coordinate_system_arbitrary

  ENTITY directionally_explicit_element_representation
    SUBTYPE OF (element_representation);
      model_ref            : fea_model;
      systems_and_freedoms : LIST [2:2] OF system_and_freedom;
      description          : text;
      coefficient          : directionally_explicit_element_coefficient;
    UNIQUE
      ur1 : model_ref, name;
    WHERE
      fu1: required_1d_nodes(SELF\element_representation.node_list,linear);
  END_ENTITY; -- directionally_explicit_element_representation

  ENTITY div_expression
    SUBTYPE OF (binary_numeric_expression);
  END_ENTITY; -- div_expression

  ENTITY document;
      id          : identifier;
      name        : label;
      description : text;
      kind        : document_type;
    UNIQUE
      ur1 : id;
  END_ENTITY; -- document

  ENTITY document_reference
    ABSTRACT SUPERTYPE;
      assigned_document : document;
      source            : label;
  END_ENTITY; -- document_reference

  ENTITY document_relationship;
      name              : label;
      description       : text;
      relating_document : document;
      related_document  : document;
  END_ENTITY; -- document_relationship
```

```
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 document_with_class
  SUBTYPE OF (document);
    class : identifier;
END_ENTITY; -- document_with_class

ENTITY draped_defined_transformation
  SUBTYPE OF (item_defined_transformation);
END_ENTITY; -- draped_defined_transformation

ENTITY dummy_node
  SUBTYPE OF (node_representation);
END_ENTITY; -- dummy_node

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([
             'STRUCTURAL_ANALYSIS_DESIGN.EDGE_BASED_WIREFRAME_MODEL',
             'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM',
             'STRUCTURAL_ANALYSIS_DESIGN.AXIS2_PLACEMENT_3D'] * TYPEOF(it))
             = 1)) )) = 0);
    wr2: (SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
             'STRUCTURAL_ANALYSIS_DESIGN.EDGE_BASED_WIREFRAME_MODEL',
             'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM'] * TYPEOF(it)) = 1)
))
             >= 1);
    wr3: (SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.items | (
             'STRUCTURAL_ANALYSIS_DESIGN.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 (
             'STRUCTURAL_ANALYSIS_DESIGN.EDGE_CURVE' IN TYPEOF(edges))) ))
             = 0)) )) = 0)) )) = 0);
    wr4: (SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.items | (
             'STRUCTURAL_ANALYSIS_DESIGN.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 | (
             'STRUCTURAL_ANALYSIS_DESIGN.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 | (
            'STRUCTURAL_ANALYSIS_DESIGN.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 ((
            'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_POINT' IN TYPEOF(edges.
            edge_start)) AND ('STRUCTURAL_ANALYSIS_DESIGN.VERTEX_POINT'
            IN TYPEOF(edges.edge_end)))) )) = 0)) )) = 0)) )) = 0);
    wr6: (SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.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 | (
            'STRUCTURAL_ANALYSIS_DESIGN.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 | (
            'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM' IN TYPEOF(it)) ) |
            (NOT (('STRUCTURAL_ANALYSIS_DESIGN.' +
            '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 (ONEOF (serial_numbered_effectivity,dated_effectivity,
      lot_effectivity));
    id : identifier;
END_ENTITY; -- effectivity

ENTITY element_analysis_message
  SUBTYPE OF (analysis_message);
    element : element_representation;
END_ENTITY; -- element_analysis_message
```

```
ENTITY element_definition
  SUBTYPE OF (shape_aspect);
END_ENTITY; -- element_definition

ENTITY element_descriptor
  SUPERTYPE OF (ONEOF (volume_3d_element_descriptor,
      axisymmetric_volume_2d_element_descriptor,
      plane_volume_2d_element_descriptor,surface_3d_element_descriptor,
      axisymmetric_surface_2d_element_descriptor,
      plane_surface_2d_element_descriptor,curve_3d_element_descriptor,
      axisymmetric_curve_2d_element_descriptor,
      plane_curve_2d_element_descriptor));
    topology_order : element_order;
    description    : text;
END_ENTITY; -- element_descriptor

ENTITY element_geometric_relationship;
    element_ref : element_or_element_group;
    item        : analysis_item_within_representation;
    aspect      : element_aspect;
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_REPRESENTATION_ITEM' IN
             TYPEOF(item.item));
    wr2: consistent_geometric_reference(aspect,item.item);
    wr3: consistent_element_or_group_reference(aspect,element_ref);
END_ENTITY; -- element_geometric_relationship

ENTITY element_group
  SUBTYPE OF (fea_group);
    elements : SET [1:?] OF element_representation;
END_ENTITY; -- element_group

ENTITY element_group_analysis_message
  SUBTYPE OF (analysis_message);
    group : element_group;
END_ENTITY; -- element_group_analysis_message

ENTITY element_material;
    material_id : identifier;
    description : text;
    properties  : SET [1:?] OF material_property_representation;
END_ENTITY; -- element_material

ENTITY element_nodal_freedom_actions
  SUBTYPE OF (state_definition);
    element      : model_or_control_element;
    nodal_action : LIST [1:?] OF element_nodal_freedom_terms;
END_ENTITY; -- element_nodal_freedom_actions

ENTITY element_nodal_freedom_terms;
    coordinate_system  : fea_axis2_placement_3d;
    degrees_of_freedom : freedoms_list;
    values             : LIST [1:?] OF measure_or_unspecified_value;
  WHERE
    wr1: (SIZEOF(degrees_of_freedom.freedoms) = SIZEOF(values));
END_ENTITY; -- element_nodal_freedom_terms

ENTITY element_representation
  SUPERTYPE OF (ONEOF (volume_3d_element_representation,
      axisymmetric_volume_2d_element_representation,
      plane_volume_2d_element_representation,
```



```
      surface_3d_element_representation,
      axisymmetric_surface_2d_element_representation,
      plane_surface_2d_element_representation,
      curve_3d_element_representation,
      axisymmetric_curve_2d_element_representation,
      plane_curve_2d_element_representation,point_element_representation,
      directionally_explicit_element_representation,
      explicit_element_representation,substructure_element_representation))
  SUBTYPE OF (representation);
    node_list : LIST [1:?] OF node_representation;
  WHERE
    wr1: (SIZEOF(QUERY ( item <* node_list | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'GEOMETRIC_NODE') IN TYPEOF(
             item)) )) = 0);
END_ENTITY; -- element_representation

ENTITY element_sequence;
    order_id    : identifier;
    control_ref : control;
    purpose     : text;
    elements    : LIST [1:?] OF model_or_control_element;
  UNIQUE
    ur1 : order_id, control_ref;
END_ENTITY; -- element_sequence

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 environment;
    syntactic_representation : generic_variable;
    semantics                : variable_semantics;
END_ENTITY; -- environment

ENTITY euler_angles;
    angles : ARRAY [1:3] OF plane_angle_measure;
END_ENTITY; -- euler_angles

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 expanded_uncertainty
  SUBTYPE OF (standard_uncertainty);
    coverage_factor : REAL;
END_ENTITY; -- expanded_uncertainty
```

```
ENTITY explicit_element_matrix;
    property_type : matrix_property_type;
    symmetry      : matrix_symmetry;
    node_dof_list : LIST [1:?] OF LIST [1:?] OF degree_of_freedom;
    matrix_values : LIST [1:?] OF context_dependent_measure;
  WHERE
    wr1: (SIZEOF(matrix_values) = number_of_terms(node_dof_list,symmetry));
END_ENTITY; -- explicit_element_matrix

ENTITY explicit_element_representation
  SUBTYPE OF (element_representation);
    model_ref   : fea_model;
    description : text;
    matrix      : explicit_element_matrix;
  UNIQUE
    ur1 : model_ref, name;
  WHERE
    wr1: (SIZEOF(QUERY ( item <* SELF\representation.items | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_AXIS2_PLACEMENT_3D') IN
              TYPEOF(item)) )) = 1);
    wr2: (SIZEOF(matrix.node_dof_list) = SIZEOF(SELF\
             element_representation.node_list));
END_ENTITY; -- explicit_element_representation

ENTITY expression
  ABSTRACT SUPERTYPE OF (ONEOF (numeric_expression,boolean_expression))
  SUBTYPE OF (generic_expression);
END_ENTITY; -- expression

ENTITY expression_conversion_based_unit
  SUBTYPE OF (named_unit, variable_semantics);
  INVERSE
    associated_variable_environment : environment FOR semantics;
END_ENTITY; -- expression_conversion_based_unit

ENTITY face
  SUPERTYPE OF (ONEOF (face_surface,oriented_face))
  SUBTYPE OF (topological_representation_item);
    bounds : SET [1:?] OF face_bound;
  WHERE
    wr1: (NOT mixed_loop_type_set(list_to_set(list_face_loops(SELF))));
    wr2: (SIZEOF(QUERY ( temp <* bounds | (
             'STRUCTURAL_ANALYSIS_DESIGN.FACE_OUTER_BOUND' IN TYPEOF(temp))
))
             <= 1);
END_ENTITY; -- face

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;
```



```
  END_ENTITY; -- face_surface

  ENTITY faceted_brep
    SUBTYPE OF (manifold_solid_brep);
  END_ENTITY; -- faceted_brep

  ENTITY faceted_brep_shape_representation
    SUBTYPE OF (shape_representation);
    WHERE
      wr1: (SIZEOF(QUERY ( it <* items | (NOT (SIZEOF([
               'STRUCTURAL_ANALYSIS_DESIGN.FACETED_BREP',
               'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM',
               'STRUCTURAL_ANALYSIS_DESIGN.AXIS2_PLACEMENT_3D'] * TYPEOF(it))
               = 1)) )) = 0);
      wr2: (SIZEOF(QUERY ( it <* items | (SIZEOF([
               'STRUCTURAL_ANALYSIS_DESIGN.FACETED_BREP',
               'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM'] * TYPEOF(it)) = 1)
))
               > 0);
      wr3: (SIZEOF(QUERY ( fbrep <* QUERY ( it <* items | (
               'STRUCTURAL_ANALYSIS_DESIGN.FACETED_BREP' IN TYPEOF(it)) )
                | (NOT (SIZEOF(QUERY ( csh <* msb_shells(fbrep) | (NOT (
               SIZEOF(QUERY ( fcs <* csh\connected_face_set.cfs_faces | (
               NOT (('STRUCTURAL_ANALYSIS_DESIGN.FACE_SURFACE' IN TYPEOF(
               fcs)) AND ('STRUCTURAL_ANALYSIS_DESIGN.PLANE' IN TYPEOF(fcs\
               face_surface.face_geometry)) AND (
               'STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT' IN TYPEOF(fcs\
               face_surface.face_geometry\elementary_surface.position.
               location)))) )) = 0)) )) = 0)) )) = 0);
      wr4: (SIZEOF(QUERY ( fbrep <* QUERY ( it <* items | (
               'STRUCTURAL_ANALYSIS_DESIGN.FACETED_BREP' IN TYPEOF(it)) )
                | (NOT (SIZEOF(QUERY ( csh <* msb_shells(fbrep) | (NOT (
               SIZEOF(QUERY ( fcs <* csh\connected_face_set.cfs_faces | (
               NOT (SIZEOF(QUERY ( bnds <* fcs.bounds | (
               'STRUCTURAL_ANALYSIS_DESIGN.FACE_OUTER_BOUND' IN TYPEOF(bnds))
))
               = 1)) )) = 0)) )) = 0)) )) = 0);
      wr5: (SIZEOF(QUERY ( msb <* QUERY ( it <* items | (
               'STRUCTURAL_ANALYSIS_DESIGN.MANIFOLD_SOLID_BREP' IN TYPEOF(
               it)) ) | ('STRUCTURAL_ANALYSIS_DESIGN.ORIENTED_CLOSED_SHELL'
               IN TYPEOF(msb\manifold_solid_brep.outer)) )) = 0);
      wr6: (SIZEOF(QUERY ( brv <* QUERY ( it <* items | (
               'STRUCTURAL_ANALYSIS_DESIGN.BREP_WITH_VOIDS' IN TYPEOF(it)) )
                | (NOT (SIZEOF(QUERY ( csh <* brv\brep_with_voids.voids |
               csh\oriented_closed_shell.orientation )) = 0)) )) = 0);
      wr7: (SIZEOF(QUERY ( mi <* QUERY ( it <* items | (
               'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM' IN TYPEOF(it)) ) |
               (NOT
('STRUCTURAL_ANALYSIS_DESIGN.FACETED_BREP_SHAPE_REPRESENTATION'
               IN TYPEOF(mi\mapped_item.mapping_source.
               mapped_representation))) )) = 0);
  END_ENTITY; -- faceted_brep_shape_representation

  ENTITY fea_area_density
    SUBTYPE OF (fea_material_property_representation_item);
      fea_constant : scalar;
  END_ENTITY; -- fea_area_density

  ENTITY fea_axis2_placement_2d
    SUBTYPE OF (axis2_placement_2d);
      system_type : coordinate_system_type;
```

```
    description : text;
END_ENTITY; -- fea_axis2_placement_2d

ENTITY fea_axis2_placement_3d
  SUBTYPE OF (axis2_placement_3d);
    system_type : coordinate_system_type;
    description : text;
END_ENTITY; -- fea_axis2_placement_3d

ENTITY fea_curve_section_geometric_relationship;
    section_ref : curve_element_section_definition;
    item        : analysis_item_within_representation;
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_REPRESENTATION_ITEM' IN
             TYPEOF(item.item));
END_ENTITY; -- fea_curve_section_geometric_relationship

ENTITY fea_group
  SUPERTYPE OF (ONEOF (element_group,node_group))
  SUBTYPE OF (group);
    model_ref : fea_model;
END_ENTITY; -- fea_group

ENTITY fea_group_relation
  SUBTYPE OF (group_relationship);
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.FEA_GROUP' IN TYPEOF(SELF\
             group_relationship.relating_group));
    wr2: ('STRUCTURAL_ANALYSIS_DESIGN.FEA_GROUP' IN TYPEOF(SELF\
             group_relationship.related_group));
END_ENTITY; -- fea_group_relation

ENTITY fea_linear_elasticity
  SUBTYPE OF (fea_material_property_representation_item);
    fea_constants : symmetric_tensor4_3d;
END_ENTITY; -- fea_linear_elasticity

ENTITY fea_mass_density
  SUBTYPE OF (fea_material_property_representation_item);
    fea_constant : scalar;
END_ENTITY; -- fea_mass_density

ENTITY fea_material_property_geometric_relationship;
    material_ref : fea_material_property_representation;
    item         : analysis_item_within_representation;
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_REPRESENTATION_ITEM' IN
             TYPEOF(item.item));
END_ENTITY; -- fea_material_property_geometric_relationship

ENTITY fea_material_property_representation
  SUBTYPE OF (material_property_representation);
  WHERE
    wr1: (SIZEOF(QUERY ( item <* SELF\property_definition_representation
             .used_representation.items | (SIZEOF([
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_LINEAR_ELASTICITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MASS_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_AREA_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
```

```
              'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
              'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MOISTURE_ABSORPTION',
              'STRUCTURAL_ANALYSIS_DESIGN.' +
              'FEA_SHELL_MEMBRANE_STIFFNESS','STRUCTURAL_ANALYSIS_DESIGN.'
              + 'FEA_SHELL_BENDING_STIFFNESS',
              'STRUCTURAL_ANALYSIS_DESIGN.' +
              'FEA_SHELL_MEMBRANE_BENDING_COUPLING_STIFFNESS',
              'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_SHELL_SHEAR_STIFFNESS']
              * TYPEOF(item)) = 1) )) = 1);
    wr2: ('STRUCTURAL_ANALYSIS_DESIGN.MATERIAL_PROPERTY' IN TYPEOF(SELF\
              property_definition_representation.definition));
END_ENTITY; -- fea_material_property_representation

ENTITY fea_material_property_representation_item
  SUPERTYPE OF (ONEOF (fea_linear_elasticity,fea_mass_density,
      fea_area_density,
      fea_tangential_coefficient_of_linear_thermal_expansion,
      fea_secant_coefficient_of_linear_thermal_expansion,
      fea_moisture_absorption,fea_shell_membrane_stiffness,
      fea_shell_bending_stiffness,
      fea_shell_membrane_bending_coupling_stiffness,
      fea_shell_shear_stiffness))
  SUBTYPE OF (representation_item);
END_ENTITY; -- fea_material_property_representation_item

ENTITY fea_model
  SUPERTYPE OF (ONEOF (fea_model_2d,fea_model_3d))
  SUBTYPE OF (representation);
    creating_software      : text;
    intended_analysis_code : SET [1:?] OF text;
    description            : text;
    analysis_type          : text;
  UNIQUE
    ur1 : name;
END_ENTITY; -- fea_model

ENTITY fea_model_2d
  SUBTYPE OF (fea_model);
    type_of_2d_analysis : axi_or_plane;
  WHERE
    wr1: (SELF\representation.context_of_items\
              geometric_representation_context.coordinate_space_dimension
              = 2);
END_ENTITY; -- fea_model_2d

ENTITY fea_model_3d
  SUBTYPE OF (fea_model);
  WHERE
    wr1: (SELF\representation.context_of_items\
              geometric_representation_context.coordinate_space_dimension
              = 3);
END_ENTITY; -- fea_model_3d

ENTITY fea_model_definition
  SUBTYPE OF (shape_aspect);
END_ENTITY; -- fea_model_definition

ENTITY fea_moisture_absorption
  SUBTYPE OF (fea_material_property_representation_item);
    fea_constants : symmetric_tensor2_3d;
END_ENTITY; -- fea_moisture_absorption
```

```
ENTITY fea_parametric_point
  SUBTYPE OF (point);
    coordinates : LIST [1:3] OF parameter_value;
  WHERE
    wr1: valid_parametric_coordinate(coordinates);
    wr2: (SIZEOF(TYPEOF(SELF) * [
             'STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT',
             'STRUCTURAL_ANALYSIS_DESIGN.POINT_ON_CURVE',
             'STRUCTURAL_ANALYSIS_DESIGN.POINT_ON_SURFACE',
             'STRUCTURAL_ANALYSIS_DESIGN.DEGENERATE_PCURVE',
             'STRUCTURAL_ANALYSIS_DESIGN.POINT_REPLICA',
             'STRUCTURAL_ANALYSIS_DESIGN.SPHERICAL_POINT',
             'STRUCTURAL_ANALYSIS_DESIGN.CYLINDRICAL_POINT']) = 0);
END_ENTITY; -- fea_parametric_point

ENTITY fea_representation_item
  SUPERTYPE OF (ONEOF (arbitrary_volume_3d_element_coordinate_system,
      parametric_volume_3d_element_coordinate_system,
      arbitrary_volume_2d_element_coordinate_system,
      parametric_volume_2d_element_coordinate_system,
      aligned_surface_3d_element_coordinate_system,
      parametric_surface_3d_element_coordinate_system,
      constant_surface_3d_element_coordinate_system,
      aligned_surface_2d_element_coordinate_system,
      parametric_surface_2d_element_coordinate_system,
      aligned_curve_3d_element_coordinate_system,
      parametric_curve_3d_element_coordinate_system,
      parametric_curve_3d_element_coordinate_direction,
      curve_2d_element_coordinate_system,
      directionally_explicit_element_coordinate_system_arbitrary,
      directionally_explicit_element_coordinate_system_aligned))
  SUBTYPE OF (representation_item);
END_ENTITY; -- fea_representation_item

ENTITY fea_secant_coefficient_of_linear_thermal_expansion
  SUBTYPE OF (fea_material_property_representation_item);
    fea_constants         : symmetric_tensor2_3d;
    reference_temperature : thermodynamic_temperature_measure;
END_ENTITY; -- fea_secant_coefficient_of_linear_thermal_expansion

ENTITY fea_shell_bending_stiffness
  SUBTYPE OF (fea_material_property_representation_item);
    fea_constants : symmetric_tensor4_2d;
END_ENTITY; -- fea_shell_bending_stiffness

ENTITY fea_shell_membrane_bending_coupling_stiffness
  SUBTYPE OF (fea_material_property_representation_item);
    fea_constants : symmetric_tensor4_2d;
END_ENTITY; -- fea_shell_membrane_bending_coupling_stiffness

ENTITY fea_shell_membrane_stiffness
  SUBTYPE OF (fea_material_property_representation_item);
    fea_constants : symmetric_tensor4_2d;
END_ENTITY; -- fea_shell_membrane_stiffness

ENTITY fea_shell_shear_stiffness
  SUBTYPE OF (fea_material_property_representation_item);
    fea_constants : symmetric_tensor2_2d;
END_ENTITY; -- fea_shell_shear_stiffness

ENTITY fea_surface_section_geometric_relationship;
```



```
    section_ref : surface_section;
    item        : analysis_item_within_representation;
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_REPRESENTATION_ITEM' IN
             TYPEOF(item.item));
END_ENTITY; -- fea_surface_section_geometric_relationship

ENTITY fea_tangential_coefficient_of_linear_thermal_expansion
  SUBTYPE OF (fea_material_property_representation_item);
    fea_constants : symmetric_tensor2_3d;
END_ENTITY; -- fea_tangential_coefficient_of_linear_thermal_expansion

ENTITY field_variable_definition
  SUPERTYPE OF (ONEOF (field_variable_element_definition,
      field_variable_element_group_value,field_variable_whole_model_value,
      field_variable_node_definition))
  SUBTYPE OF (state_definition);
END_ENTITY; -- field_variable_definition

ENTITY field_variable_element_definition
  SUPERTYPE OF (ONEOF (volume_3d_element_field_variable_definition,
      volume_2d_element_field_variable_definition,
      surface_3d_element_field_variable_definition,
      surface_2d_element_field_variable_definition,
      curve_3d_element_field_variable_definition,
      curve_2d_element_field_variable_definition))
  SUBTYPE OF (field_variable_definition);
END_ENTITY; -- field_variable_element_definition

ENTITY field_variable_element_group_value
  SUBTYPE OF (field_variable_definition);
    group             : element_group;
    simple_value      : field_value;
    variable          : volume_aggregated_variable;
    coordinate_system : OPTIONAL fea_axis2_placement_3d;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- field_variable_element_group_value

ENTITY field_variable_node_definition
  SUPERTYPE OF (ONEOF (volume_3d_node_field_variable_definition,
      volume_2d_node_field_variable_definition,
      surface_3d_node_field_variable_definition,
      surface_2d_node_field_variable_definition,
      curve_3d_node_field_variable_definition,
      curve_2d_node_field_variable_definition))
  SUBTYPE OF (field_variable_definition);
    node  : node_output_reference;
    group : OPTIONAL element_group;
END_ENTITY; -- field_variable_node_definition

ENTITY field_variable_whole_model_value
  SUBTYPE OF (field_variable_definition);
    simple_value      : field_value;
    variable          : volume_aggregated_variable;
    coordinate_system : OPTIONAL fea_axis2_placement_3d;
  WHERE
```

```
    wr1: necessary_value_coordinate_system(simple_value,
              coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
              state_definition.defined_state));
END_ENTITY; -- field_variable_whole_model_value

ENTITY flat_pattern_ply_representation_relationship
  SUBTYPE OF (shape_representation_relationship);
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.SHAPE_REPRESENTATION' IN (TYPEOF(
              SELF\representation_relationship.rep_1) * TYPEOF(SELF\
              representation_relationship.rep_2)));
    wr2: (SELF\representation_relationship.rep_1.context_of_items\
              geometric_representation_context.coordinate_space_dimension
              = 3);
END_ENTITY; -- flat_pattern_ply_representation_relationship

ENTITY founded_item;
END_ENTITY; -- founded_item

ENTITY freedom_and_coefficient;
    freedom : degree_of_freedom;
    a       : measure_or_unspecified_value;
END_ENTITY; -- freedom_and_coefficient

ENTITY freedoms_list;
    freedoms : LIST [1:?] OF degree_of_freedom;
END_ENTITY; -- freedoms_list

ENTITY functionally_defined_transformation;
    name        : label;
    description : text;
END_ENTITY; -- functionally_defined_transformation

ENTITY generic_expression
  ABSTRACT SUPERTYPE OF (ONEOF (simple_generic_expression,
      unary_generic_expression,binary_generic_expression,
      multiple_arity_generic_expression));
  WHERE
    wr1: is_acyclic(SELF);
END_ENTITY; -- generic_expression

ENTITY generic_literal
  ABSTRACT SUPERTYPE
  SUBTYPE OF (simple_generic_expression);
END_ENTITY; -- generic_literal

ENTITY generic_variable
  ABSTRACT SUPERTYPE
  SUBTYPE OF (simple_generic_expression);
  INVERSE
    interpretation : environment FOR syntactic_representation;
END_ENTITY; -- generic_variable

ENTITY geometric_curve_set
  SUBTYPE OF (geometric_set);
  WHERE
    wr1: (SIZEOF(QUERY ( temp <* SELF\geometric_set.elements | (
              'STRUCTURAL_ANALYSIS_DESIGN.SURFACE' IN TYPEOF(temp)) )) = 0);
END_ENTITY; -- geometric_curve_set
```



```
  ENTITY geometric_node
    SUBTYPE OF (node_representation);
    WHERE
      wr1: (SIZEOF(QUERY ( item <* SELF\representation.items | (
                 'STRUCTURAL_ANALYSIS_DESIGN.POINT' IN TYPEOF(item)) )) = 1);
  END_ENTITY; -- geometric_node

  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,
        shell_based_surface_model,shell_based_wireframe_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
('STRUCTURAL_ANALYSIS_DESIGN.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 geometrically_bounded_surface_shape_representation
    SUBTYPE OF (shape_representation);
    WHERE
      wr1: (SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
                 'STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_SET',
                 'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM',
                 'STRUCTURAL_ANALYSIS_DESIGN.AXIS2_PLACEMENT_3D'] * TYPEOF(it))
                 = 1)) )) = 0);
      wr2: (SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
                 'STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_SET',
                 'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM'] * TYPEOF(it)) = 1)
))
                 > 0);
      wr3: (SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
                 'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM' IN TYPEOF(it)) ) |
                 (NOT ((('STRUCTURAL_ANALYSIS_DESIGN.' +
                 'GEOMETRICALLY_BOUNDED_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 | (
                 'STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_SET' IN TYPEOF(mr_it))
))
                 > 0))) )) = 0);
      wr4: (SIZEOF(QUERY ( gs <* QUERY ( it <* SELF.items | (
                 'STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_SET' IN TYPEOF(it)) )
                  | (NOT (SIZEOF(QUERY ( pnt <* QUERY ( gsel <* gs\
                 geometric_set.elements | ('STRUCTURAL_ANALYSIS_DESIGN.POINT'
```

```
            IN TYPEOF(gsel)) ) | (NOT gbsf_check_point(pnt)) )) = 0)) ))
            = 0);
    wr5: (SIZEOF(QUERY ( gs <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_SET' IN TYPEOF(it)) )
             | (NOT (SIZEOF(QUERY ( cv <* QUERY ( gsel <* gs\
            geometric_set.elements | ('STRUCTURAL_ANALYSIS_DESIGN.CURVE'
            IN TYPEOF(gsel)) ) | (NOT gbsf_check_curve(cv)) )) = 0)) ))
            = 0);
    wr6: (SIZEOF(QUERY ( gs <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_SET' IN TYPEOF(it)) )
             | (NOT (SIZEOF(QUERY ( sf <* QUERY ( gsel <* gs\
            geometric_set.elements | (
            'STRUCTURAL_ANALYSIS_DESIGN.SURFACE' IN TYPEOF(gsel)) ) | (
            NOT gbsf_check_surface(sf)) )) = 0)) )) = 0);
    wr7: (SIZEOF(QUERY ( gs <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_SET' IN TYPEOF(it)) )
             | (SIZEOF(QUERY ( gsel <* gs\geometric_set.elements | (
            'STRUCTURAL_ANALYSIS_DESIGN.SURFACE' IN TYPEOF(gsel)) )) > 0)
))
            > 0);
  END_ENTITY; -- geometrically_bounded_surface_shape_representation

  ENTITY geometrically_bounded_wireframe_shape_representation
    SUBTYPE OF (shape_representation);
    WHERE
      wr1: (SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF(TYPEOF(it) * [
            'STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_CURVE_SET',
            'STRUCTURAL_ANALYSIS_DESIGN.AXIS2_PLACEMENT_3D',
            'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM']) = 1)) )) = 0);
      wr2: (SIZEOF(QUERY ( it <* SELF.items | (SIZEOF(TYPEOF(it) * [
            'STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_CURVE_SET',
            'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM']) = 1) )) >= 1);
      wr3: (SIZEOF(QUERY ( gcs <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_CURVE_SET' IN TYPEOF(
            it)) ) | (NOT (SIZEOF(QUERY ( crv <* QUERY ( elem <* gcs\
            geometric_set.elements | ('STRUCTURAL_ANALYSIS_DESIGN.CURVE'
            IN TYPEOF(elem)) ) | (NOT
            valid_geometrically_bounded_wf_curve(crv)) )) = 0)) )) = 0);
      wr4: (SIZEOF(QUERY ( gcs <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_CURVE_SET' IN TYPEOF(
            it)) ) | (NOT (SIZEOF(QUERY ( pnts <* QUERY ( elem <* gcs\
            geometric_set.elements | ('STRUCTURAL_ANALYSIS_DESIGN.POINT'
            IN TYPEOF(elem)) ) | (NOT
            valid_geometrically_bounded_wf_point(pnts)) )) = 0)) )) = 0);
      wr5: (SIZEOF(QUERY ( gcs <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_CURVE_SET' IN TYPEOF(
            it)) ) | (NOT (SIZEOF(QUERY ( cnc <* QUERY ( elem <* gcs\
            geometric_set.elements | ('STRUCTURAL_ANALYSIS_DESIGN.CONIC'
            IN TYPEOF(elem)) ) | (NOT (
            'STRUCTURAL_ANALYSIS_DESIGN.AXIS2_PLACEMENT_3D' IN TYPEOF(
            cnc\conic.position))) )) = 0)) )) = 0);
      wr6: (SIZEOF(QUERY ( gcs <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_CURVE_SET' IN TYPEOF(
            it)) ) | (NOT (SIZEOF(QUERY ( pline <* QUERY ( elem <* gcs\
            geometric_set.elements | (
            'STRUCTURAL_ANALYSIS_DESIGN.POLYLINE' IN TYPEOF(elem)) ) | (
            NOT (SIZEOF(pline\polyline.points) > 2)) )) = 0)) )) = 0);
      wr7: (SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM' IN TYPEOF(it)) ) |
            (NOT (('STRUCTURAL_ANALYSIS_DESIGN.' +
            'GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_REPRESENTATION') IN
```



```
               TYPEOF(mi\mapped_item.mapping_source.mapped_representation)))
))
               = 0);
  END_ENTITY; -- geometrically_bounded_wireframe_shape_representation

  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 grounded_damper
    SUBTYPE OF (point_element_matrix);
      damping_coefficients : ARRAY [1:6] OF context_dependent_measure;
  END_ENTITY; -- grounded_damper

  ENTITY grounded_spring
    SUBTYPE OF (point_element_matrix);
      stiffness_coefficients : ARRAY [1:6] OF context_dependent_measure;
  END_ENTITY; -- grounded_spring

  ENTITY group;
      name        : label;
      description : text;
  END_ENTITY; -- group

  ENTITY group_relationship;
      name           : label;
      description    : 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 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 interval_expression
    SUBTYPE OF (boolean_expression, multiple_arity_generic_expression);
    DERIVE
      interval_low  : generic_expression := SELF\
                         multiple_arity_generic_expression.operands[1];
      interval_item : generic_expression := SELF\
                         multiple_arity_generic_expression.operands[2];
      interval_high : generic_expression := SELF\
                         multiple_arity_generic_expression.operands[3];
```

```
  WHERE
    wr1: (('STRUCTURAL_ANALYSIS_DESIGN.EXPRESSION' IN TYPEOF(
             interval_low)) AND ('STRUCTURAL_ANALYSIS_DESIGN.EXPRESSION'
             IN TYPEOF(interval_item)) AND (
             'STRUCTURAL_ANALYSIS_DESIGN.EXPRESSION' IN TYPEOF(
             interval_high)));
    wr2: ((('STRUCTURAL_ANALYSIS_DESIGN.STRING_EXPRESSION' IN TYPEOF(
             SELF.interval_low)) AND (
             'STRUCTURAL_ANALYSIS_DESIGN.STRING_EXPRESSION' IN TYPEOF(
             SELF.interval_high)) AND (
             'STRUCTURAL_ANALYSIS_DESIGN.STRING_EXPRESSION' IN TYPEOF(
             SELF.interval_item))) OR ((
             'STRUCTURAL_ANALYSIS_DESIGN.STRING_EXPRESSION' IN TYPEOF(
             SELF.interval_low)) AND (
             'STRUCTURAL_ANALYSIS_DESIGN.NUMERIC_EXPRESSION' IN TYPEOF(
             SELF.interval_item)) AND (
             'STRUCTURAL_ANALYSIS_DESIGN.NUMERIC_EXPRESSION' IN TYPEOF(
             SELF.interval_high))));
END_ENTITY; -- interval_expression

ENTITY item_defined_transformation;
    name             : label;
    description      : text;
    transform_item_1 : representation_item;
    transform_item_2 : representation_item;
END_ENTITY; -- item_defined_transformation

ENTITY laid_defined_transformation
  SUBTYPE OF (item_defined_transformation);
END_ENTITY; -- laid_defined_transformation

ENTITY length_measure_with_unit
  SUBTYPE OF (measure_with_unit);
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.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 linear_constraint_equation_element
  SUBTYPE OF (constraint_element);
```



```
      freedoms_and_coefficients : SET [1:?] OF
linear_constraint_equation_nodal_term;
      description               : text;
  END_ENTITY; -- linear_constraint_equation_element

  ENTITY linear_constraint_equation_element_value
    SUBTYPE OF (state_definition);
      element : linear_constraint_equation_element;
      b       : measure_or_unspecified_value;
  END_ENTITY; -- linear_constraint_equation_element_value

  ENTITY linear_constraint_equation_nodal_term;
      node                          : node_representation;
      coordinate_system             : fea_axis2_placement_3d;
      freedom_and_coefficient_term : freedom_and_coefficient;
      dependent                     : LOGICAL;
  END_ENTITY; -- linear_constraint_equation_nodal_term

  ENTITY linearly_superimposed_state
    SUBTYPE OF (state);
    INVERSE
      components : SET [1:?] OF state_component FOR state;
  END_ENTITY; -- linearly_superimposed_state

  ENTITY literal_number
    ABSTRACT SUPERTYPE OF (real_literal)
    SUBTYPE OF (simple_numeric_expression, generic_literal);
      the_value : NUMBER;
  END_ENTITY; -- literal_number

  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 lot_effectivity
    SUBTYPE OF (effectivity);
      effectivity_lot_id   : identifier;
      effectivity_lot_size : measure_with_unit;
  END_ENTITY; -- lot_effectivity

  ENTITY make_from_usage_option
    SUBTYPE OF (product_definition_usage);
      ranking           : INTEGER;
      ranking_rationale : text;
      quantity          : measure_with_unit;
    WHERE
      wr1: (ranking > 0);
  END_ENTITY; -- make_from_usage_option

  ENTITY manifold_solid_brep
    SUBTYPE OF (solid_model);
```

```
    outer : closed_shell;
END_ENTITY; -- manifold_solid_brep

ENTITY manifold_surface_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    wr1 : (SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
            'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_SURFACE_MODEL',
            'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM',
            'STRUCTURAL_ANALYSIS_DESIGN.AXIS2_PLACEMENT_3D'] * TYPEOF(
            it)) = 1)) )) = 0);
    wr2 : (SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
            'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_SURFACE_MODEL',
            'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM'] * TYPEOF(it)) = 1)
))
            > 0);
    wr3 : (SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM' IN TYPEOF(it)) )
             | (NOT
(('STRUCTURAL_ANALYSIS_DESIGN.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 | (
            'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_SURFACE_MODEL' IN
            TYPEOF(mr_it)) )) > 0))) )) = 0);
    wr4 : (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_SURFACE_MODEL' IN
            TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( sh <* sbsm\
            shell_based_surface_model.sbsm_boundary | (NOT (SIZEOF([
            'STRUCTURAL_ANALYSIS_DESIGN.OPEN_SHELL',
            'STRUCTURAL_ANALYSIS_DESIGN.ORIENTED_CLOSED_SHELL',
            'STRUCTURAL_ANALYSIS_DESIGN.CLOSED_SHELL'] * TYPEOF(sh)) =
            1)) )) = 0)) )) = 0);
    wr5 : (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_SURFACE_MODEL' IN
            TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* sbsm\
            shell_based_surface_model.sbsm_boundary | (NOT (SIZEOF(
            QUERY ( fa <* cfs\connected_face_set.cfs_faces | (NOT (
            'STRUCTURAL_ANALYSIS_DESIGN.FACE_SURFACE' IN TYPEOF(fa))) ))
            = 0)) )) = 0)) )) = 0);
    wr6 : (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_SURFACE_MODEL' IN
            TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* sbsm\
            shell_based_surface_model.sbsm_boundary | (NOT (SIZEOF(
            QUERY ( fa <* cfs\connected_face_set.cfs_faces | (NOT ((
            'STRUCTURAL_ANALYSIS_DESIGN.ADVANCED_FACE' IN TYPEOF(fa))
            OR msf_surface_check(fa\face_surface.face_geometry))) )) =
            0)) )) = 0)) )) = 0);
    wr7 : (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_SURFACE_MODEL' IN
            TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* sbsm\
            shell_based_surface_model.sbsm_boundary | (NOT (SIZEOF(
            QUERY ( fa <* cfs\connected_face_set.cfs_faces | (NOT ((
            'STRUCTURAL_ANALYSIS_DESIGN.ADVANCED_FACE' IN TYPEOF(fa))
            OR (SIZEOF(QUERY ( bnds <* fa.bounds | (NOT (SIZEOF([
            'STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP',
            'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_LOOP'] * TYPEOF(bnds.
            bound)) = 1)) )) = 0))) )) = 0)) )) = 0)) )) = 0);
    wr8 : (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_SURFACE_MODEL' IN
            TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* sbsm\
```



```
        shell_based_surface_model.sbsm_boundary | (NOT (SIZEOF(
        QUERY ( fa <* cfs\connected_face_set.cfs_faces | (NOT ((
        'STRUCTURAL_ANALYSIS_DESIGN.ADVANCED_FACE' IN TYPEOF(fa))
        OR (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* fa.bounds
         | ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP' IN TYPEOF(bnds.
        bound)) ) | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds\path.
        edge_list | (NOT ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_CURVE'
        IN TYPEOF(oe.edge_element))) )) = 0)) )) = 0))) )) = 0)) ))
        = 0)) )) = 0);
wr9 : (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
        'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_SURFACE_MODEL' IN
        TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* sbsm\
        shell_based_surface_model.sbsm_boundary | (NOT (SIZEOF(
        QUERY ( fa <* cfs\connected_face_set.cfs_faces | (NOT ((
        'STRUCTURAL_ANALYSIS_DESIGN.ADVANCED_FACE' IN TYPEOF(fa))
        OR (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* fa.bounds
         | ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP' IN TYPEOF(bnds.
        bound)) ) | (NOT (SIZEOF(QUERY ( oe_cv <* QUERY ( oe <*
        elp_fbnds\path.edge_list | (
        'STRUCTURAL_ANALYSIS_DESIGN.EDGE_CURVE' IN TYPEOF(oe.
        edge_element)) ) | (NOT (SIZEOF([
        'STRUCTURAL_ANALYSIS_DESIGN.B_SPLINE_CURVE',
        'STRUCTURAL_ANALYSIS_DESIGN.CONIC',
        'STRUCTURAL_ANALYSIS_DESIGN.CURVE_REPLICA',
        'STRUCTURAL_ANALYSIS_DESIGN.LINE',
        'STRUCTURAL_ANALYSIS_DESIGN.OFFSET_CURVE_3D',
        'STRUCTURAL_ANALYSIS_DESIGN.PCURVE',
        'STRUCTURAL_ANALYSIS_DESIGN.POLYLINE',
        'STRUCTURAL_ANALYSIS_DESIGN.SURFACE_CURVE'] * TYPEOF(oe_cv.
        edge_element\edge_curve.edge_geometry)) = 1)) )) = 0)) )) =
         0))) )) = 0)) )) = 0)) )) = 0);
wr10: (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
        'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_SURFACE_MODEL' IN
        TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* sbsm\
        shell_based_surface_model.sbsm_boundary | (NOT (SIZEOF(
        QUERY ( fa <* cfs\connected_face_set.cfs_faces | (NOT ((
        'STRUCTURAL_ANALYSIS_DESIGN.ADVANCED_FACE' IN TYPEOF(fa))
        OR (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* fa.bounds
         | ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP' IN TYPEOF(bnds.
        bound)) ) | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds\path.
        edge_list | (NOT msf_curve_check(oe.edge_element\edge_curve
        .edge_geometry)) )) = 0)) )) = 0))) )) = 0)) )) = 0)) )) =
        0);
wr11: (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
        'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_SURFACE_MODEL' IN
        TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* sbsm\
        shell_based_surface_model.sbsm_boundary | (NOT (SIZEOF(
        QUERY ( fa <* cfs\connected_face_set.cfs_faces | (NOT ((
        'STRUCTURAL_ANALYSIS_DESIGN.ADVANCED_FACE' IN TYPEOF(fa))
        OR (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* fa.bounds
         | ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP' IN TYPEOF(bnds.
        bound)) ) | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds\path.
        edge_list | (NOT ((
        'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_POINT' IN TYPEOF(oe.
        edge_element.edge_start)) AND (
        'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_POINT' IN TYPEOF(oe.
        edge_element.edge_end)))) )) = 0)) )) = 0))) )) = 0)) )) =
        0)) )) = 0);
wr12: (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
        'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_SURFACE_MODEL' IN
        TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* sbsm\
```

```
                shell_based_surface_model.sbsm_boundary | (NOT (SIZEOF(
                QUERY ( fa <* cfs\connected_face_set.cfs_faces | (NOT ((
                'STRUCTURAL_ANALYSIS_DESIGN.ADVANCED_FACE' IN TYPEOF(fa))
                OR (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* fa.bounds
                 | ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP' IN TYPEOF(bnds.
                bound)) ) | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds\path.
                edge_list | (NOT ((SIZEOF([
                'STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT',
                'STRUCTURAL_ANALYSIS_DESIGN.DEGENERATE_PCURVE',
                'STRUCTURAL_ANALYSIS_DESIGN.POINT_ON_CURVE',
                'STRUCTURAL_ANALYSIS_DESIGN.POINT_ON_SURFACE'] * TYPEOF(oe.
                edge_element.edge_start\vertex_point.vertex_geometry)) = 1)
                AND (SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT',
                'STRUCTURAL_ANALYSIS_DESIGN.DEGENERATE_PCURVE',
                'STRUCTURAL_ANALYSIS_DESIGN.POINT_ON_CURVE',
                'STRUCTURAL_ANALYSIS_DESIGN.POINT_ON_SURFACE'] * TYPEOF(oe.
                edge_element.edge_end\vertex_point.vertex_geometry)) = 1)))
))
                = 0)) )) = 0))) )) = 0)) )) = 0)) )) = 0);
      wr13: (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
                'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_SURFACE_MODEL' IN
                TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* sbsm\
                shell_based_surface_model.sbsm_boundary | (NOT (SIZEOF(
                QUERY ( fa <* cfs\connected_face_set.cfs_faces | (NOT ((
                'STRUCTURAL_ANALYSIS_DESIGN.ADVANCED_FACE' IN TYPEOF(fa))
                OR (SIZEOF(QUERY ( vlp_fbnds <* QUERY ( bnds <* fa.bounds
                 | ('STRUCTURAL_ANALYSIS_DESIGN.VERTEX_LOOP' IN TYPEOF(bnds
                .bound)) ) | (NOT (
                'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_POINT' IN TYPEOF(
                vlp_fbnds\vertex_loop.loop_vertex))) )) = 0))) )) = 0)) ))
                = 0)) )) = 0);
      wr14: (SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
                'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_SURFACE_MODEL' IN
                TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( cfs <* sbsm\
                shell_based_surface_model.sbsm_boundary | (NOT (SIZEOF(
                QUERY ( fa <* cfs\connected_face_set.cfs_faces | (NOT ((
                'STRUCTURAL_ANALYSIS_DESIGN.ADVANCED_FACE' IN TYPEOF(fa))
                OR (SIZEOF(QUERY ( vlp_fbnds <* QUERY ( bnds <* fa.bounds
                 | ('STRUCTURAL_ANALYSIS_DESIGN.VERTEX_LOOP' IN TYPEOF(bnds
                .bound)) ) | (NOT (SIZEOF([
                'STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT',
                'STRUCTURAL_ANALYSIS_DESIGN.DEGENERATE_PCURVE',
                'STRUCTURAL_ANALYSIS_DESIGN.POINT_ON_CURVE',
                'STRUCTURAL_ANALYSIS_DESIGN.POINT_ON_SURFACE'] * TYPEOF(
                vlp_fbnds\vertex_loop.loop_vertex\vertex_point.
                vertex_geometry)) = 1)) )) = 0))) )) = 0)) )) = 0)) )) = 0);
  END_ENTITY; -- manifold_surface_shape_representation

  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: ('STRUCTURAL_ANALYSIS_DESIGN.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 material_designation;
    name        : label;
    definitions : SET [1:?] OF characterized_definition;
END_ENTITY; -- material_designation

ENTITY material_designation_characterization;
    name        : label;
    description : text;
    designation : material_designation;
    property    : characterized_material_property;
END_ENTITY; -- material_designation_characterization

ENTITY material_property
  SUBTYPE OF (property_definition);
  UNIQUE
    ur1 : name, definition;
  WHERE
    wr1: (('STRUCTURAL_ANALYSIS_DESIGN.CHARACTERIZED_OBJECT' IN TYPEOF(
             SELF\property_definition.definition)) OR (SIZEOF(bag_to_set(
             USEDIN(SELF,'STRUCTURAL_ANALYSIS_DESIGN.' +
             'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) -
             QUERY ( temp <* bag_to_set(USEDIN(SELF,
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'MATERIAL_PROPERTY_REPRESENTATION') IN TYPEOF(temp)) )) = 0));
END_ENTITY; -- material_property

ENTITY material_property_representation
  SUBTYPE OF (property_definition_representation);
    dependent_environment : data_environment;
END_ENTITY; -- material_property_representation

ENTITY measure_qualification;
    name              : label;
    description       : text;
    qualified_measure : measure_with_unit;
    qualifiers        : SET [1:?] OF value_qualifier;
  WHERE
    wr1: (SIZEOF(QUERY ( temp <* qualifiers | (
             'STRUCTURAL_ANALYSIS_DESIGN.PRECISION_QUALIFIER' IN TYPEOF(
             temp)) )) < 2);
END_ENTITY; -- measure_qualification

ENTITY measure_representation_item
  SUBTYPE OF (representation_item, measure_with_unit);
```

```
  END_ENTITY; -- measure_representation_item

  ENTITY measure_with_unit
    SUPERTYPE OF (ONEOF (length_measure_with_unit,mass_measure_with_unit,
        time_measure_with_unit,thermodynamic_temperature_measure_with_unit,
        amount_of_substance_measure_with_unit,plane_angle_measure_with_unit,
        solid_angle_measure_with_unit,area_measure_with_unit,
        volume_measure_with_unit));
      value_component : measure_value;
      unit_component  : unit;
    WHERE
      wr1: valid_units(SELF);
  END_ENTITY; -- measure_with_unit

  ENTITY minus_expression
    SUBTYPE OF (binary_numeric_expression);
  END_ENTITY; -- minus_expression

  ENTITY mult_expression
    SUBTYPE OF (multiple_arity_numeric_expression);
  END_ENTITY; -- mult_expression

  ENTITY multiple_arity_boolean_expression
    ABSTRACT SUPERTYPE
    SUBTYPE OF (boolean_expression, multiple_arity_generic_expression);
      SELF\multiple_arity_generic_expression.operands : LIST [2:?] OF
boolean_expression;
  END_ENTITY; -- multiple_arity_boolean_expression

  ENTITY multiple_arity_generic_expression
    ABSTRACT SUPERTYPE
    SUBTYPE OF (generic_expression);
      operands : LIST [2:?] OF generic_expression;
  END_ENTITY; -- multiple_arity_generic_expression

  ENTITY multiple_arity_numeric_expression
    ABSTRACT SUPERTYPE OF (ONEOF (plus_expression,mult_expression))
    SUBTYPE OF (numeric_expression, multiple_arity_generic_expression);
      SELF\multiple_arity_generic_expression.operands : LIST [2:?] OF
numeric_expression;
  END_ENTITY; -- multiple_arity_numeric_expression

  ENTITY name_assignment
    ABSTRACT SUPERTYPE;
      assigned_name : label;
  END_ENTITY; -- name_assignment

  ENTITY named_unit
    SUPERTYPE OF (ONEOF (si_unit,conversion_based_unit,
        context_dependent_unit) ANDOR ONEOF (length_unit,mass_unit,
        time_unit,thermodynamic_temperature_unit,amount_of_substance_unit,
        plane_angle_unit,solid_angle_unit,area_unit,volume_unit,ratio_unit));
      dimensions : dimensional_exponents;
  END_ENTITY; -- named_unit

  ENTITY named_unit_variable
    SUBTYPE OF (named_unit, variable_semantics);
    INVERSE
      associated_variable_environment : environment FOR semantics;
  END_ENTITY; -- named_unit_variable
```

```
ENTITY next_assembly_usage_occurrence
  SUBTYPE OF (assembly_component_usage);
END_ENTITY; -- next_assembly_usage_occurrence

ENTITY no_symmetry_control
  SUBTYPE OF (symmetry_control);
END_ENTITY; -- no_symmetry_control

ENTITY nodal_dof_reduction
  SUBTYPE OF (constraint_element);
    required_node     : node_output_reference;
    coordinate_system : fea_axis2_placement_3d;
    freedoms          : SET [1:?] OF degree_of_freedom;
    description       : text;
END_ENTITY; -- nodal_dof_reduction

ENTITY nodal_freedom_action_definition
  SUBTYPE OF (nodal_freedom_and_value_definition);
    action : action_type;
END_ENTITY; -- nodal_freedom_action_definition

ENTITY nodal_freedom_and_value_definition
  SUPERTYPE OF (ONEOF (nodal_freedom_values,
      nodal_freedom_action_definition))
  SUBTYPE OF (state_definition);
    node               : node_output_reference;
    coordinate_system  : fea_axis2_placement_3d;
    degrees_of_freedom : freedoms_list;
    values             : LIST [1:?] OF measure_or_unspecified_value;
  WHERE
    wr1: (SIZEOF(degrees_of_freedom.freedoms) = SIZEOF(values));
END_ENTITY; -- nodal_freedom_and_value_definition

ENTITY nodal_freedom_values
  SUBTYPE OF (nodal_freedom_and_value_definition);
END_ENTITY; -- nodal_freedom_values

ENTITY node
  SUPERTYPE OF (node_with_vector ANDOR
      node_with_solution_coordinate_system)
  SUBTYPE OF (node_representation);
  WHERE
    wr1: (SIZEOF(QUERY ( item <* SELF\representation.items | (
           'STRUCTURAL_ANALYSIS_DESIGN.POINT' IN TYPEOF(item)) )) = 1);
END_ENTITY; -- node

ENTITY node_analysis_message
  SUBTYPE OF (analysis_message);
    node : node_output_reference;
END_ENTITY; -- node_analysis_message

ENTITY node_definition
  SUBTYPE OF (shape_aspect);
END_ENTITY; -- node_definition

ENTITY node_geometric_relationship;
    node_ref : node_or_node_group;
    item     : analysis_item_within_representation;
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.GEOMETRIC_REPRESENTATION_ITEM' IN
           TYPEOF(item.item));
```

```
END_ENTITY; -- node_geometric_relationship

ENTITY node_group
  SUBTYPE OF (fea_group);
    nodes : SET [1:?] OF node_representation;
END_ENTITY; -- node_group

ENTITY node_representation
  SUPERTYPE OF (ONEOF (node,dummy_node,geometric_node))
  SUBTYPE OF (representation);
    model_ref : fea_model;
  UNIQUE
    ur1 : model_ref, name;
END_ENTITY; -- node_representation

ENTITY node_sequence;
    order_id    : identifier;
    control_ref : control;
    purpose     : text;
    nodes       : LIST [1:?] OF node_representation;
  UNIQUE
    ur1 : order_id, control_ref;
END_ENTITY; -- node_sequence

ENTITY node_set
  SUBTYPE OF (geometric_representation_item);
    nodes : SET [1:?] OF node_representation;
  WHERE
    wr1: (SIZEOF(QUERY ( tmp <* nodes | (tmp\representation.
             context_of_items :<>: nodes[1]\representation.
             context_of_items) )) = 0);
END_ENTITY; -- node_set

ENTITY node_with_solution_coordinate_system
  SUBTYPE OF (node);
  WHERE
    wr1: (((SIZEOF(QUERY ( item <* SELF\representation.items | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_AXIS2_PLACEMENT_3D') IN
              TYPEOF(item)) )) = 1) AND (SELF\representation.
             context_of_items\geometric_representation_context.
             coordinate_space_dimension = 3)) OR ((SIZEOF(
             QUERY ( item <* SELF\representation.items | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_AXIS2_PLACEMENT_2D') IN
              TYPEOF(item)) )) = 1) AND (SELF\representation.
             context_of_items\geometric_representation_context.
             coordinate_space_dimension = 2)));
END_ENTITY; -- node_with_solution_coordinate_system

ENTITY node_with_vector
  SUBTYPE OF (node);
  WHERE
    wr1: (SIZEOF(QUERY ( item <* SELF\representation.items | (
             'STRUCTURAL_ANALYSIS_DESIGN.DIRECTION' IN TYPEOF(item)) )) =
              1);
END_ENTITY; -- node_with_vector

ENTITY numeric_expression
  ABSTRACT SUPERTYPE OF (ONEOF (simple_numeric_expression,
      unary_numeric_expression,binary_numeric_expression,
      multiple_arity_numeric_expression))
  SUBTYPE OF (expression);
```



```
  DERIVE
    is_int       : BOOLEAN := is_int_expr(SELF);
    sql_mappable : BOOLEAN := is_sql_mappable(SELF);
END_ENTITY; -- numeric_expression

ENTITY numeric_variable
  SUPERTYPE OF (real_numeric_variable)
  SUBTYPE OF (simple_numeric_expression, variable);
  WHERE
    wr1: (('STRUCTURAL_ANALYSIS_DESIGN.INT_NUMERIC_VARIABLE' IN TYPEOF(
             SELF)) OR (
             'STRUCTURAL_ANALYSIS_DESIGN.REAL_NUMERIC_VARIABLE' IN
             TYPEOF(SELF)));
END_ENTITY; -- numeric_variable

ENTITY offset_curve_2d
  SUBTYPE OF (curve);
    basis_curve    : curve;
    distance       : length_measure;
    self_intersect : LOGICAL;
  WHERE
    wr1: (basis_curve.dim = 2);
END_ENTITY; -- offset_curve_2d

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 : text;
END_ENTITY; -- organization
```

```
ENTITY organization_assignment
  ABSTRACT SUPERTYPE;
    assigned_organization : organization;
    role                  : organization_role;
END_ENTITY; -- organization_assignment

ENTITY organization_relationship;
    name                  : label;
    description           : text;
    relating_organization : organization;
    related_organization  : organization;
END_ENTITY; -- organization_relationship

ENTITY organization_role;
    name : label;
END_ENTITY; -- organization_role

ENTITY organizational_address
  SUBTYPE OF (address);
    organizations : SET [1:?] OF organization;
    description   : text;
END_ENTITY; -- organizational_address

ENTITY organizational_project;
    name                        : label;
    description                 : text;
    responsible_organizations : SET [1:?] OF organization;
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 ('STRUCTURAL_ANALYSIS_DESIGN.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 ('STRUCTURAL_ANALYSIS_DESIGN.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 ('STRUCTURAL_ANALYSIS_DESIGN.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 ('STRUCTURAL_ANALYSIS_DESIGN.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 ('STRUCTURAL_ANALYSIS_DESIGN.ORIENTED_PATH' IN TYPEOF(SELF
             .path_element)));
END_ENTITY; -- oriented_path

ENTITY outer_boundary_curve
  SUBTYPE OF (boundary_curve);
END_ENTITY; -- outer_boundary_curve

ENTITY output_request_state
  SUBTYPE OF (state);
    steps : SET [1:?] OF control_analysis_step;
END_ENTITY; -- output_request_state

ENTITY parabola
  SUBTYPE OF (conic);
    focal_dist : length_measure;
  WHERE
    wr1: (focal_dist <> 0);
END_ENTITY; -- parabola

ENTITY parametric_curve_3d_element_coordinate_direction
  SUBTYPE OF (fea_representation_item);
    orientation : direction;
  WHERE
    wr1: (SELF\geometric_representation_item.dim = 3);
END_ENTITY; -- parametric_curve_3d_element_coordinate_direction

ENTITY parametric_curve_3d_element_coordinate_system
```

```
  SUBTYPE OF (fea_representation_item);
    direction : parametric_curve_3d_element_coordinate_direction;
END_ENTITY; -- parametric_curve_3d_element_coordinate_system

ENTITY parametric_representation_context
  SUBTYPE OF (representation_context);
END_ENTITY; -- parametric_representation_context

ENTITY parametric_surface_2d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
END_ENTITY; -- parametric_surface_2d_element_coordinate_system

ENTITY parametric_surface_3d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
    axis  : INTEGER;
    angle : plane_angle_measure;
  WHERE
    wr1: ((axis >= 1) AND (axis <= 2));
END_ENTITY; -- parametric_surface_3d_element_coordinate_system

ENTITY parametric_volume_2d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
    axis  : INTEGER;
    angle : plane_angle_measure;
  WHERE
    wr1: ((axis >= 1) AND (axis <= 2));
END_ENTITY; -- parametric_volume_2d_element_coordinate_system

ENTITY parametric_volume_3d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
    axis_1    : INTEGER;
    axis_2    : INTEGER;
    eu_angles : euler_angles;
  WHERE
    wr1: ((axis_1 >= 1) AND (axis_1 <= 3) AND (axis_2 >= 1) AND (axis_2
             <= 3) AND (NOT (axis_1 = axis_2)));
END_ENTITY; -- parametric_volume_3d_element_coordinate_system

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: ('STRUCTURAL_ANALYSIS_DESIGN.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 percentage_laminate_definition
  SUBTYPE OF (product_definition);
  WHERE
```



```
    wr1: (SIZEOF(QUERY ( pdr <* USEDIN(SELF,
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'PRODUCT_DEFINITION_RELATIONSHIP.' +
             'RELATING_PRODUCT_DEFINITION') | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'NEXT_ASSEMBLY_USAGE_OCCURRENCE') IN TYPEOF(pdr)) )) > 0);
END_ENTITY; -- percentage_laminate_definition

ENTITY percentage_ply_definition
  SUBTYPE OF (product_definition);
  WHERE
    wr1: (SIZEOF(QUERY ( pdr <* USEDIN(SELF,
             'STRUCTURAL_ANALYSIS_DESIGN.' +
'PRODUCT_DEFINITION_RELATIONSHIP.RELATING_PRODUCT_DEFINITION')
              | ((
             'STRUCTURAL_ANALYSIS_DESIGN.PERCENTAGE_LAMINATE_DEFINITION'
             IN TYPEOF(pdr.related_product_definition)) AND (pdr.name =
             'makeup and properties')) )) = 0);
END_ENTITY; -- percentage_ply_definition

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;
  UNIQUE
    ur1 : id;
  WHERE
    wr1: (EXISTS(last_name) OR EXISTS(first_name));
END_ENTITY; -- person

ENTITY person_and_organization;
    the_person       : person;
    the_organization : organization;
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;
END_ENTITY; -- person_and_organization_role

ENTITY personal_address
  SUBTYPE OF (address);
    people      : SET [1:?] OF person;
    description : 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_2d_element_property
  SUPERTYPE OF (simple_plane_2d_element_property);
    depth : context_dependent_measure;
END_ENTITY; -- plane_2d_element_property

ENTITY plane_angle_measure_with_unit
  SUBTYPE OF (measure_with_unit);
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.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 plane_curve_2d_element_descriptor
  SUBTYPE OF (element_descriptor);
    purpose    : SET [1:?] OF SET [1:?] OF curve_element_purpose;
    assumption : plane_2d_element_purpose;
END_ENTITY; -- plane_curve_2d_element_descriptor

ENTITY plane_curve_2d_element_representation
  SUBTYPE OF (element_representation);
    model_ref          : fea_model_2d;
    element_descriptor : plane_curve_2d_element_descriptor;
    property           : curve_2d_element_property;
    depth_property     : plane_2d_element_property;
    material           : element_material;
  UNIQUE
    ur1 : model_ref, name;
  WHERE
    wr1: (model_ref.type_of_2d_analysis = planar);
    wr2: (SIZEOF(QUERY ( item <* SELF\representation.items | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'CURVE_2D_ELEMENT_COORDINATE_SYSTEM') IN TYPEOF(item)) )) =
             1);
    wr3: (SIZEOF(QUERY ( item1 <* material.properties | (SIZEOF(
             QUERY ( item2 <* item1\property_definition_representation.
             used_representation.items | (SIZEOF([
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_LINEAR_ELASTICITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MASS_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_AREA_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MOISTURE_ABSORPTION'] *
```



```
             TYPEOF(item2)) = 1) )) = 1) )) >= 1);
    wr4: ('STRUCTURAL_ANALYSIS_DESIGN.PARAMETRIC_REPRESENTATION_CONTEXT'
             IN TYPEOF(SELF\representation.context_of_items));
    fu1: required_0d_nodes(SELF\element_representation.node_list);
END_ENTITY; -- plane_curve_2d_element_representation

ENTITY plane_surface_2d_element_descriptor
  SUBTYPE OF (element_descriptor);
    purpose    : SET [1:?] OF SET [1:?] OF surface_element_purpose;
    assumption : plane_2d_element_purpose;
END_ENTITY; -- plane_surface_2d_element_descriptor

ENTITY plane_surface_2d_element_representation
  SUBTYPE OF (element_representation);
    model_ref          : fea_model_2d;
    element_descriptor : plane_surface_2d_element_descriptor;
    property           : surface_element_property;
    depth_property     : plane_2d_element_property;
    material           : element_material;
  UNIQUE
    ur1 : model_ref, name;
  WHERE
    wr1: (model_ref.type_of_2d_analysis = planar);
    wr2: ((SIZEOF(QUERY ( item <* SELF\representation.items | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'PARAMETRIC_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM') IN
             TYPEOF(item)) )) + SIZEOF(QUERY ( item <* SELF\
             representation.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
             'ALIGNED_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM') IN TYPEOF(
             item)) ))) = 1);
    wr3: (SIZEOF(QUERY ( item1 <* material.properties | (SIZEOF(
             QUERY ( item2 <* item1\property_definition_representation.
             used_representation.items | (SIZEOF([
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_LINEAR_ELASTICITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MASS_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_AREA_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MOISTURE_ABSORPTION',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_SHELL_MEMBRANE_STIFFNESS','STRUCTURAL_ANALYSIS_DESIGN.'
             + 'FEA_SHELL_BENDING_STIFFNESS',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_SHELL_MEMBRANE_BENDING_COUPLING_STIFFNESS',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_SHELL_SHEAR_STIFFNESS']
             * TYPEOF(item2)) = 1) )) = 1) )) >= 1);
    wr4: ('STRUCTURAL_ANALYSIS_DESIGN.PARAMETRIC_REPRESENTATION_CONTEXT'
             IN TYPEOF(SELF\representation.context_of_items));
    fu1: required_1d_nodes(SELF\element_representation.node_list,
             element_descriptor\element_descriptor.topology_order);
END_ENTITY; -- plane_surface_2d_element_representation

ENTITY plane_volume_2d_element_descriptor
  SUBTYPE OF (element_descriptor);
    purpose    : SET [1:?] OF SET [1:?] OF volume_element_purpose;
    shape      : element_2d_shape;
    assumption : plane_2d_element_purpose;
END_ENTITY; -- plane_volume_2d_element_descriptor
```

```
ENTITY plane_volume_2d_element_representation
  SUBTYPE OF (element_representation);
    model_ref          : fea_model_2d;
    element_descriptor : plane_volume_2d_element_descriptor;
    depth_property     : plane_2d_element_property;
    material           : element_material;
  UNIQUE
    ur1 : model_ref, name;
  WHERE
    wr1: (model_ref.type_of_2d_analysis = planar);
    wr2: ((SIZEOF(QUERY ( item <* SELF\representation.items | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'PARAMETRIC_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM') IN TYPEOF(
             item)) )) + SIZEOF(QUERY ( item <* SELF\representation.items
              | (('STRUCTURAL_ANALYSIS_DESIGN.' +
             'ARBITRARY_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM') IN TYPEOF(
             item)) ))) = 1);
    wr3: (SIZEOF(QUERY ( item1 <* material.properties | (SIZEOF(
             QUERY ( item2 <* item1\property_definition_representation.
             used_representation.items | (SIZEOF([
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_LINEAR_ELASTICITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MASS_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_AREA_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MOISTURE_ABSORPTION'] *
              TYPEOF(item2)) = 1) )) = 1) )) >= 1);
    wr4: ('STRUCTURAL_ANALYSIS_DESIGN.PARAMETRIC_REPRESENTATION_CONTEXT'
             IN TYPEOF(SELF\representation.context_of_items));
    fu1: required_2d_nodes(SELF\element_representation.node_list,
             element_descriptor.shape,element_descriptor\
             element_descriptor.topology_order);
END_ENTITY; -- plane_volume_2d_element_representation

ENTITY plus_expression
  SUBTYPE OF (multiple_arity_numeric_expression);
END_ENTITY; -- plus_expression

ENTITY ply_laminate_definition
  SUBTYPE OF (product_definition);
  WHERE
    wr1: (SIZEOF(QUERY ( pdr <* USEDIN(SELF,
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'PRODUCT_DEFINITION_RELATIONSHIP.' +
             'RELATING_PRODUCT_DEFINITION') | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'NEXT_ASSEMBLY_USAGE_OCCURRENCE') IN TYPEOF(pdr)) )) = 1);
END_ENTITY; -- ply_laminate_definition

ENTITY ply_laminate_sequence_definition
  SUBTYPE OF (product_definition);
  WHERE
    wr1: (SIZEOF(QUERY ( pdr <* USEDIN(SELF,
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'PRODUCT_DEFINITION_RELATIONSHIP.' +
             'RELATING_PRODUCT_DEFINITION') | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'NEXT_ASSEMBLY_USAGE_OCCURRENCE') IN TYPEOF(pdr)) )) > 0);
END_ENTITY; -- ply_laminate_sequence_definition
```



```
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_and_vector
  SUBTYPE OF (shape_representation);
  WHERE
    wr1: (SIZEOF(SELF\representation.items) = 3);
    wr2: ((’STRUCTURAL_ANALYSIS_DESIGN.POINT’ IN TYPEOF(SELF\
             representation.items[1])) AND (
             ’STRUCTURAL_ANALYSIS_DESIGN.DIRECTION’ IN TYPEOF(SELF\
             representation.items[2])) AND (
             ’STRUCTURAL_ANALYSIS_DESIGN.DIRECTION’ IN TYPEOF(SELF\
             representation.items[3])));
END_ENTITY; -- point_and_vector

ENTITY point_constraint
  SUBTYPE OF (constraint_element);
    required_point            : analysis_item_within_representation;
    coordinate_system         : fea_axis2_placement_3d;
    freedoms_and_coefficients : SET [1:?] OF freedom_and_coefficient;
    description               : text;
  WHERE
    wr1: ((’STRUCTURAL_ANALYSIS_DESIGN.POINT’ IN TYPEOF(required_point.
             item)) OR (’STRUCTURAL_ANALYSIS_DESIGN.VERTEX_POINT’ IN
             TYPEOF(required_point.item)));
END_ENTITY; -- point_constraint

ENTITY point_element_matrix
  SUPERTYPE OF (ONEOF (stationary_mass,grounded_spring,grounded_damper));
END_ENTITY; -- point_element_matrix

ENTITY point_element_representation
  SUBTYPE OF (element_representation);
    model_ref   : fea_model;
    description : text;
    matrix_set  : SET [1:?] OF point_element_matrix;
  UNIQUE
    ur1 : model_ref, name;
  WHERE
    wr1: (SIZEOF(QUERY ( item <* SELF\representation.items | ((
             ’STRUCTURAL_ANALYSIS_DESIGN.’ + ’FEA_AXIS2_PLACEMENT_3D’) IN
              TYPEOF(item)) )) = 1);
    fu1: required_0d_nodes(SELF\element_representation.node_list);
END_ENTITY; -- point_element_representation

ENTITY point_freedom_action_definition
  SUBTYPE OF (point_freedom_and_value_definition);
    action : action_type;
END_ENTITY; -- point_freedom_action_definition

ENTITY point_freedom_and_value_definition
  SUPERTYPE OF (ONEOF (point_freedom_values,
      point_freedom_action_definition))
  SUBTYPE OF (state_definition);
    required_point     : analysis_item_within_representation;
    coordinate_system  : fea_axis2_placement_3d;
    degrees_of_freedom : freedoms_list;
    values             : LIST [1:?] OF measure_or_unspecified_value;
```

```
    WHERE
      wr1: (SIZEOF(degrees_of_freedom.freedoms) = SIZEOF(values));
      wr2: ((’STRUCTURAL_ANALYSIS_DESIGN.POINT’ IN TYPEOF(required_point.
               item)) OR (’STRUCTURAL_ANALYSIS_DESIGN.VERTEX_POINT’ IN
               TYPEOF(required_point.item)));
  END_ENTITY; -- point_freedom_and_value_definition

  ENTITY point_freedom_values
    SUBTYPE OF (point_freedom_and_value_definition);
  END_ENTITY; -- point_freedom_values

  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_path
    SUBTYPE OF (shape_representation);
    WHERE
      wr1: (SIZEOF(USEDIN(SELF,’STRUCTURAL_ANALYSIS_DESIGN.’ +
               ’REPRESENTATION_RELATIONSHIP.REP_1’)) = 1);
  END_ENTITY; -- point_path

  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 point_representation
    SUBTYPE OF (shape_representation);
    WHERE
      wr1: (SIZEOF(QUERY ( it <* SELF\representation.items | (NOT (SIZEOF(
               [’STRUCTURAL_ANALYSIS_DESIGN.NODE_SET’,
               ’STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM’,
               ’STRUCTURAL_ANALYSIS_DESIGN.’ + ’FEA_AXIS2_PLACEMENT_3D’] *
               TYPEOF(it)) = 1)) )) = 0);
      wr2: (SIZEOF(QUERY ( it <* SELF\representation.items | (SIZEOF([
               ’STRUCTURAL_ANALYSIS_DESIGN.NODE_SET’,
               ’STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM’] * TYPEOF(it)) = 1)
))
               >= 1);
  END_ENTITY; -- point_representation

  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 precision_qualifier;
    precision_value : INTEGER;
END_ENTITY; -- precision_qualifier

ENTITY product;
    id                 : identifier;
    name               : label;
    description        : text;
    frame_of_reference : SET [1:?] OF product_context;
  UNIQUE
    ur1 : id;
END_ENTITY; -- product

ENTITY product_category;
    name        : label;
    description : OPTIONAL text;
END_ENTITY; -- product_category

ENTITY product_category_relationship;
    name         : label;
    description  : 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_concept;
    id             : identifier;
    name           : label;
    description    : text;
    market_context : product_concept_context;
  UNIQUE
    ur1 : id;
END_ENTITY; -- product_concept

ENTITY product_concept_context
  SUBTYPE OF (application_context_element);
    market_segment_type : label;
END_ENTITY; -- product_concept_context

ENTITY product_context
  SUBTYPE OF (application_context_element);
    discipline_type : label;
END_ENTITY; -- product_context

ENTITY product_definition;
    id                 : identifier;
    description        : text;
    formation          : product_definition_formation;
    frame_of_reference : product_definition_context;
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_effectivity
  SUBTYPE OF (effectivity);
    usage : product_definition_relationship;
  UNIQUE
    ur1 : usage, id;
END_ENTITY; -- product_definition_effectivity

ENTITY product_definition_formation;
    id          : identifier;
    description : text;
    of_product  : product;
  UNIQUE
    ur1 : id, of_product;
END_ENTITY; -- product_definition_formation

ENTITY product_definition_formation_relationship;
    id                                    : identifier;
    name                                  : label;
    description                           : text;
    relating_product_definition_formation : product_definition_formation;
    related_product_definition_formation  : product_definition_formation;
END_ENTITY; -- product_definition_formation_relationship

ENTITY product_definition_formation_with_specified_source
  SUBTYPE OF (product_definition_formation);
    make_or_buy : source;
END_ENTITY; -- product_definition_formation_with_specified_source

ENTITY product_definition_relationship;
    id                          : identifier;
    name                        : label;
    description                 : 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: (NOT ('STRUCTURAL_ANALYSIS_DESIGN.SHAPE_DEFINITION' IN TYPEOF(
             SELF\property_definition.definition)));
END_ENTITY; -- product_definition_shape

ENTITY product_definition_usage
  SUPERTYPE OF (ONEOF (make_from_usage_option,assembly_component_usage))
  SUBTYPE OF (product_definition_relationship);
  UNIQUE
    ur1 : id, relating_product_definition, related_product_definition;
  WHERE
    wr1: acyclic_product_definition_relationship(SELF,[SELF\
             product_definition_relationship.related_product_definition],
             'STRUCTURAL_ANALYSIS_DESIGN.PRODUCT_DEFINITION_USAGE');
END_ENTITY; -- product_definition_usage

ENTITY product_definition_with_associated_documents
  SUBTYPE OF (product_definition);
    documentation_ids : SET [1:?] OF document;
END_ENTITY; -- product_definition_with_associated_documents
```



```
ENTITY product_material_composition_relationship
  SUBTYPE OF (product_definition_relationship);
    class                : label;
    constituent_amount   : SET [1:?] OF measure_with_unit;
    composition_basis    : label;
    determination_method : text;
END_ENTITY; -- product_material_composition_relationship

ENTITY product_related_product_category
  SUBTYPE OF (product_category);
    products : SET [1:?] OF product;
END_ENTITY; -- product_related_product_category

ENTITY promissory_usage_occurrence
  SUBTYPE OF (assembly_component_usage);
END_ENTITY; -- promissory_usage_occurrence

ENTITY property_definition;
    name        : label;
    description : text;
    definition  : characterized_definition;
END_ENTITY; -- property_definition

ENTITY property_definition_representation;
    definition          : property_definition;
    used_representation : representation;
END_ENTITY; -- property_definition_representation

ENTITY qualified_representation_item
  SUBTYPE OF (representation_item);
    qualifiers : SET [1:?] OF value_qualifier;
  WHERE
    wr1: (SIZEOF(QUERY ( temp <* qualifiers | (
             'STRUCTURAL_ANALYSIS_DESIGN.PRECISION_QUALIFIER' IN TYPEOF(
             temp)) )) < 2);
END_ENTITY; -- qualified_representation_item

ENTITY qualitative_uncertainty
  SUBTYPE OF (uncertainty_qualifier);
    uncertainty_value : text;
END_ENTITY; -- qualitative_uncertainty

ENTITY quantified_assembly_component_usage
  SUBTYPE OF (assembly_component_usage);
    quantity : measure_with_unit;
END_ENTITY; -- quantified_assembly_component_usage

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 real_literal
  SUBTYPE OF (literal_number);
    SELF\literal_number.the_value : REAL;
END_ENTITY; -- real_literal

ENTITY real_numeric_variable
  SUBTYPE OF (numeric_variable);
END_ENTITY; -- real_numeric_variable

ENTITY rectangular_composite_surface
  SUBTYPE OF (bounded_surface);
    segments : LIST [1:?] OF LIST [1:?] OF surface_patch;
  DERIVE
    n_u : INTEGER := SIZEOF(segments);
    n_v : INTEGER := SIZEOF(segments[1]);
  WHERE
    wr1: ([] = QUERY ( s <* segments | (n_v <> SIZEOF(s)) ));
    wr2: constraints_rectangular_composite_surface(SELF);
END_ENTITY; -- rectangular_composite_surface

ENTITY rectangular_trimmed_surface
  SUBTYPE OF (bounded_surface);
    basis_surface : surface;
    u1            : parameter_value;
    u2            : parameter_value;
    v1            : parameter_value;
    v2            : parameter_value;
    usense        : BOOLEAN;
```

```
    vsense        : BOOLEAN;
  WHERE
    wr1: (u1 <> u2);
    wr2: (v1 <> v2);
    wr3: ((('STRUCTURAL_ANALYSIS_DESIGN.ELEMENTARY_SURFACE' IN TYPEOF(
             basis_surface)) AND (NOT ('STRUCTURAL_ANALYSIS_DESIGN.PLANE'
             IN TYPEOF(basis_surface)))) OR (
             'STRUCTURAL_ANALYSIS_DESIGN.SURFACE_OF_REVOLUTION' IN
             TYPEOF(basis_surface)) OR (usense = (u2 > u1)));
    wr4: (('STRUCTURAL_ANALYSIS_DESIGN.SPHERICAL_SURFACE' IN TYPEOF(
             basis_surface)) OR (
             'STRUCTURAL_ANALYSIS_DESIGN.TOROIDAL_SURFACE' IN TYPEOF(
             basis_surface)) OR (vsense = (v2 > v1)));
END_ENTITY; -- rectangular_trimmed_surface

ENTITY reparametrised_composite_curve_segment
  SUBTYPE OF (composite_curve_segment);
    param_length : parameter_value;
  WHERE
    wr1: (param_length > 0);
END_ENTITY; -- reparametrised_composite_curve_segment

ENTITY representation;
    name             : label;
    items            : SET [1:?] OF representation_item;
    context_of_items : representation_context;
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 : text;
    rep_1       : representation;
    rep_2       : representation;
END_ENTITY; -- representation_relationship

ENTITY representation_relationship_with_transformation
  SUBTYPE OF (representation_relationship);
```

```
    transformation_operator : transformation;
  WHERE
    wr1: (SELF\representation_relationship.rep_1.context_of_items :<>:
             SELF\representation_relationship.rep_2.context_of_items);
END_ENTITY; -- representation_relationship_with_transformation

ENTITY result;
    result_id          : identifier;
    creating_software : text;
    description        : text;
END_ENTITY; -- result

ENTITY result_analysis_step
  SUPERTYPE OF (ONEOF (result_linear_static_analysis_sub_step,
      result_linear_modes_and_frequencies_analysis_sub_step))
  SUBTYPE OF (analysis_step);
    analysis_result : result;
  UNIQUE
    ur1 : analysis_control, analysis_result;
END_ENTITY; -- result_analysis_step

ENTITY result_linear_modes_and_frequencies_analysis_sub_step
  SUBTYPE OF (result_analysis_step);
    states : SET [1:?] OF calculated_state;
END_ENTITY; -- result_linear_modes_and_frequencies_analysis_sub_step

ENTITY result_linear_static_analysis_sub_step
  SUBTYPE OF (result_analysis_step);
    state : calculated_state;
END_ENTITY; -- result_linear_static_analysis_sub_step

ENTITY retention
  SUBTYPE OF (action);
  WHERE
    wr1: (SIZEOF(QUERY ( aa <* USEDIN(SELF,'STRUCTURAL_ANALYSIS_DESIGN.'
             + 'ACTION_ASSIGNMENT.ASSIGNED_ACTION') | (
             'STRUCTURAL_ANALYSIS_DESIGN.RETENTION_ASSIGNMENT' IN TYPEOF(
             aa)) )) > 0);
    wr2: ((SIZEOF(QUERY ( ada <* USEDIN(SELF,
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'APPLIED_DATE_ASSIGNMENT.ITEMS') | (ada.role.name =
             'start date') )) + SIZEOF(QUERY ( adata <* USEDIN(SELF,
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS') | (adata.role.name
             = 'start date') ))) = 1);
    wr3: ((SIZEOF(QUERY ( ada <* USEDIN(SELF,
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'APPLIED_DATE_ASSIGNMENT.ITEMS') | (ada.role.name =
             'earliest end date') )) + SIZEOF(QUERY ( adata <* USEDIN(
             SELF,'STRUCTURAL_ANALYSIS_DESIGN.' +
             'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS') | (adata.role.name
             = 'earliest end date') ))) = 1);
    wr4: ((SIZEOF(QUERY ( ada <* USEDIN(SELF,
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'APPLIED_DATE_ASSIGNMENT.ITEMS') | (ada.role.name =
             'latest end date') )) + SIZEOF(QUERY ( adata <* USEDIN(SELF,
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'APPLIED_DATE_AND_TIME_ASSIGNMENT.ITEMS') | (adata.role.name
             = 'latest end date') ))) = 1);
END_ENTITY; -- retention
```



```
ENTITY retention_assignment
  SUBTYPE OF (action_assignment);
    items : SET [1:?] OF retention_item;
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.RETENTION' IN TYPEOF(SELF.
             assigned_action));
END_ENTITY; -- retention_assignment

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: ('STRUCTURAL_ANALYSIS_DESIGN.PCURVE' IN TYPEOF(SELF\
             surface_curve.associated_geometry[1]));
    wr4: ('STRUCTURAL_ANALYSIS_DESIGN.PCURVE' IN TYPEOF(SELF\
             surface_curve.associated_geometry[2]));
END_ENTITY; -- seam_curve

ENTITY security_classification;
    name           : label;
    purpose        : text;
    security_level : security_classification_level;
END_ENTITY; -- security_classification

ENTITY security_classification_assignment
  ABSTRACT SUPERTYPE;
    assigned_security_classification : security_classification;
END_ENTITY; -- security_classification_assignment

ENTITY security_classification_level;
    name : label;
END_ENTITY; -- security_classification_level

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           : text;
    of_shape              : product_definition_shape;
    product_definitional  : LOGICAL;
END_ENTITY; -- shape_aspect

ENTITY shape_aspect_relationship;
    name                  : label;
    description           : text;
    relating_shape_aspect : shape_aspect;
    related_shape_aspect  : shape_aspect;
END_ENTITY; -- shape_aspect_relationship

ENTITY shape_definition_representation
  SUBTYPE OF (property_definition_representation);
  WHERE
    wr1: (('STRUCTURAL_ANALYSIS_DESIGN.SHAPE_DEFINITION' IN TYPEOF(SELF.
             definition.definition)) OR (
```



```
              'STRUCTURAL_ANALYSIS_DESIGN.PRODUCT_DEFINITION_SHAPE' IN
              TYPEOF(SELF.definition)));
     wr2: ('STRUCTURAL_ANALYSIS_DESIGN.SHAPE_REPRESENTATION' IN TYPEOF(
              SELF.used_representation));
 END_ENTITY; -- shape_definition_representation

 ENTITY shape_dimension_representation
   SUBTYPE OF (shape_representation);
   WHERE
     wr1: (SIZEOF(QUERY ( temp <* SELF.items | (NOT (
              'STRUCTURAL_ANALYSIS_DESIGN.MEASURE_REPRESENTATION_ITEM' IN
              TYPEOF(temp))) )) = 0);
     wr2: (SIZEOF(SELF.items) <= 2);
     wr3: (SIZEOF(QUERY ( pos_mri <* QUERY ( real_mri <* SELF.items | (
              'REAL' IN TYPEOF(real_mri\measure_with_unit.value_component))
)
               | (NOT (pos_mri\measure_with_unit.value_component > 0)) ))
              = 0);
 END_ENTITY; -- shape_dimension_representation

 ENTITY shape_representation
   SUBTYPE OF (representation);
 END_ENTITY; -- shape_representation

 ENTITY shape_representation_relationship
   SUBTYPE OF (representation_relationship);
   WHERE
     wr1: ('STRUCTURAL_ANALYSIS_DESIGN.SHAPE_REPRESENTATION' IN (TYPEOF(
              SELF\representation_relationship.rep_1) + TYPEOF(SELF\
              representation_relationship.rep_2)));
 END_ENTITY; -- shape_representation_relationship

 ENTITY shell_based_surface_model
   SUBTYPE OF (geometric_representation_item);
     sbsm_boundary : SET [1:?] OF shell;
   WHERE
     wr1: constraints_geometry_shell_based_surface_model(SELF);
 END_ENTITY; -- shell_based_surface_model

 ENTITY shell_based_wireframe_model
   SUBTYPE OF (geometric_representation_item);
     sbwm_boundary : SET [1:?] OF shell;
   WHERE
     wr1: constraints_geometry_shell_based_wireframe_model(SELF);
 END_ENTITY; -- shell_based_wireframe_model

 ENTITY shell_based_wireframe_shape_representation
   SUBTYPE OF (shape_representation);
   WHERE
     wr1 : (SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
               'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_WIREFRAME_MODEL',
               'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM',
               'STRUCTURAL_ANALYSIS_DESIGN.AXIS2_PLACEMENT_3D'] * TYPEOF(
               it)) = 1)) )) = 0);
     wr2 : (SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
               'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_WIREFRAME_MODEL',
               'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM'] * TYPEOF(it)) = 1)
))
               >= 1);
     wr3 : (SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
               'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN
```



```
             TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <*
            sbwm\shell_based_wireframe_model.sbwm_boundary | (
            'STRUCTURAL_ANALYSIS_DESIGN.WIRE_SHELL' IN TYPEOF(sb)) ) |
            (NOT (SIZEOF(QUERY ( eloop <* QUERY ( wsb <* ws\wire_shell.
            wire_shell_extent | ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP'
            IN TYPEOF(wsb)) ) | (NOT (SIZEOF(QUERY ( el <* eloop\path.
            edge_list | (NOT ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_CURVE'
            IN TYPEOF(el.edge_element))) )) = 0)) )) = 0)) )) = 0)) ))
            = 0);
  wr4 : (SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN
             TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <*
            sbwm\shell_based_wireframe_model.sbwm_boundary | (
            'STRUCTURAL_ANALYSIS_DESIGN.WIRE_SHELL' IN TYPEOF(sb)) ) |
            (NOT (SIZEOF(QUERY ( eloop <* QUERY ( wsb <* ws\wire_shell.
            wire_shell_extent | ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP'
            IN TYPEOF(wsb)) ) | (NOT (SIZEOF(QUERY ( pline_el <*
            QUERY ( el <* eloop\path.edge_list | (
            'STRUCTURAL_ANALYSIS_DESIGN.POLYLINE' IN TYPEOF(el.
            edge_element\edge_curve.edge_geometry)) ) | (NOT (SIZEOF(
            pline_el.edge_element\edge_curve.edge_geometry\polyline.
            points) > 2)) )) = 0)) )) = 0)) )) = 0)) )) = 0);
  wr5 : (SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN
             TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <*
            sbwm\shell_based_wireframe_model.sbwm_boundary | (
            'STRUCTURAL_ANALYSIS_DESIGN.WIRE_SHELL' IN TYPEOF(sb)) ) |
            (NOT (SIZEOF(QUERY ( eloop <* QUERY ( wsb <* ws\wire_shell.
            wire_shell_extent | ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP'
            IN TYPEOF(wsb)) ) | (NOT (SIZEOF(QUERY ( el <* eloop\path.
            edge_list | (NOT valid_wireframe_edge_curve(el.edge_element
            \edge_curve.edge_geometry)) )) = 0)) )) = 0)) )) = 0)) )) =
             0);
  wr6 : (SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN
             TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <*
            sbwm\shell_based_wireframe_model.sbwm_boundary | (
            'STRUCTURAL_ANALYSIS_DESIGN.WIRE_SHELL' IN TYPEOF(sb)) ) |
            (NOT (SIZEOF(QUERY ( eloop <* QUERY ( wsb <* ws\wire_shell.
            wire_shell_extent | ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP'
            IN TYPEOF(wsb)) ) | (NOT (SIZEOF(QUERY ( el <* eloop\path.
            edge_list | (NOT ((
            'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_POINT' IN TYPEOF(el.
            edge_element.edge_start)) AND (
            'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_POINT' IN TYPEOF(el.
            edge_element.edge_end)))) )) = 0)) )) = 0)) )) = 0)) )) = 0);
  wr7 : (SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN
             TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <*
            sbwm\shell_based_wireframe_model.sbwm_boundary | (
            'STRUCTURAL_ANALYSIS_DESIGN.WIRE_SHELL' IN TYPEOF(sb)) ) |
            (NOT (SIZEOF(QUERY ( eloop <* QUERY ( wsb <* ws\wire_shell.
            wire_shell_extent | ('STRUCTURAL_ANALYSIS_DESIGN.EDGE_LOOP'
            IN TYPEOF(wsb)) ) | (NOT (SIZEOF(QUERY ( el <* eloop\path.
            edge_list | (NOT (valid_wireframe_vertex_point(el.
            edge_element.edge_start\vertex_point.vertex_geometry) AND
            valid_wireframe_vertex_point(el.edge_element.edge_end\
            vertex_point.vertex_geometry))) )) = 0)) )) = 0)) )) = 0)) ))
            = 0);
  wr8 : (SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
            'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN
```

```
                 TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <*
                sbwm\shell_based_wireframe_model.sbwm_boundary | (
                'STRUCTURAL_ANALYSIS_DESIGN.WIRE_SHELL' IN TYPEOF(sb)) ) |
                (NOT (SIZEOF(QUERY ( vloop <* QUERY ( wsb <* ws\wire_shell.
                wire_shell_extent | (
                'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_LOOP' IN TYPEOF(wsb)) )
                 | (NOT ('STRUCTURAL_ANALYSIS_DESIGN.VERTEX_POINT' IN
                TYPEOF(vloop\vertex_loop.loop_vertex))) )) = 0)) )) = 0)) ))
                = 0);
      wr9 : (SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
                'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN
                 TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <*
                sbwm\shell_based_wireframe_model.sbwm_boundary | (
                'STRUCTURAL_ANALYSIS_DESIGN.WIRE_SHELL' IN TYPEOF(sb)) ) |
                (NOT (SIZEOF(QUERY ( vloop <* QUERY ( wsb <* ws\wire_shell.
                wire_shell_extent | (
                'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_LOOP' IN TYPEOF(wsb)) )
                 | (NOT valid_wireframe_vertex_point(vloop\vertex_loop.
                loop_vertex\vertex_point.vertex_geometry)) )) = 0)) )) = 0))
))
                = 0);
      wr10: (SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
                'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN
                 TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( vs <* QUERY ( sb <*
                sbwm\shell_based_wireframe_model.sbwm_boundary | (
                'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_SHELL' IN TYPEOF(sb)) )
                 | (NOT ('STRUCTURAL_ANALYSIS_DESIGN.VERTEX_POINT' IN
                TYPEOF(vs\vertex_shell.vertex_shell_extent.loop_vertex))) ))
                = 0)) )) = 0);
      wr11: (SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
                'STRUCTURAL_ANALYSIS_DESIGN.SHELL_BASED_WIREFRAME_MODEL' IN
                 TYPEOF(it)) ) | (NOT (SIZEOF(QUERY ( vs <* QUERY ( sb <*
                sbwm\shell_based_wireframe_model.sbwm_boundary | (
                'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_SHELL' IN TYPEOF(sb)) )
                 | (NOT valid_wireframe_vertex_point(vs\vertex_shell.
                vertex_shell_extent.loop_vertex\vertex_point.
                vertex_geometry)) )) = 0)) )) = 0);
      wr12: (SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
                'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM' IN TYPEOF(it)) )
                 | (NOT (('STRUCTURAL_ANALYSIS_DESIGN.' +
                'SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION') IN TYPEOF(mi\
                mapped_item.mapping_source.mapped_representation))) )) = 0);
      wr13: (SELF.context_of_items\geometric_representation_context.
                coordinate_space_dimension = 3);
  END_ENTITY; -- shell_based_wireframe_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(SELF.name);
  END_ENTITY; -- si_unit

  ENTITY simple_generic_expression
    ABSTRACT SUPERTYPE OF (ONEOF (generic_literal,generic_variable))
    SUBTYPE OF (generic_expression);
  END_ENTITY; -- simple_generic_expression
```

```
ENTITY simple_numeric_expression
  ABSTRACT SUPERTYPE OF (ONEOF (literal_number,numeric_variable))
  SUBTYPE OF (numeric_expression, simple_generic_expression);
END_ENTITY; -- simple_numeric_expression

ENTITY simple_plane_2d_element_property
  SUBTYPE OF (plane_2d_element_property);
END_ENTITY; -- simple_plane_2d_element_property

ENTITY single_point_constraint_element
  SUBTYPE OF (constraint_element);
    required_node       : node_output_reference;
    coordinate_system   : fea_axis2_placement_3d;
    freedoms_and_values : SET [1:?] OF freedom_and_coefficient;
    description         : text;
END_ENTITY; -- single_point_constraint_element

ENTITY single_point_constraint_element_values
  SUBTYPE OF (state_definition);
    element            : single_point_constraint_element;
    degrees_of_freedom : freedoms_list;
    b                  : LIST [1:?] OF measure_or_unspecified_value;
  WHERE
    wr1: (SIZEOF(degrees_of_freedom.freedoms) = SIZEOF(b));
END_ENTITY; -- single_point_constraint_element_values

ENTITY smeared_material_definition
  SUBTYPE OF (product_definition);
END_ENTITY; -- smeared_material_definition

ENTITY solid_angle_measure_with_unit
  SUBTYPE OF (measure_with_unit);
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.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_constraint
  SUBTYPE OF (constraint_element);
    required_solid            : analysis_item_within_representation;
    coordinate_system         : fea_axis2_placement_3d;
    freedoms_and_coefficients : SET [1:?] OF freedom_and_coefficient;
    description               : text;
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.SOLID_MODEL' IN TYPEOF(
             required_solid.item));
END_ENTITY; -- solid_constraint
```

```
ENTITY solid_freedom_action_definition
  SUBTYPE OF (solid_freedom_and_value_definition);
    action : action_type;
END_ENTITY; -- solid_freedom_action_definition

ENTITY solid_freedom_and_value_definition
  SUPERTYPE OF (ONEOF (solid_freedom_values,
      solid_freedom_action_definition))
  SUBTYPE OF (state_definition);
    required_solid     : analysis_item_within_representation;
    coordinate_system  : fea_axis2_placement_3d;
    degrees_of_freedom : freedoms_list;
    values             : LIST [1:?] OF measure_or_unspecified_value;
  WHERE
    wr1: (SIZEOF(degrees_of_freedom.freedoms) = SIZEOF(values));
    wr2: ('STRUCTURAL_ANALYSIS_DESIGN.SOLID_MODEL' IN TYPEOF(
             required_solid.item));
END_ENTITY; -- solid_freedom_and_value_definition

ENTITY solid_freedom_values
  SUBTYPE OF (solid_freedom_and_value_definition);
END_ENTITY; -- solid_freedom_values

ENTITY solid_model
  SUPERTYPE OF (manifold_solid_brep)
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- solid_model

ENTITY specified_higher_usage_occurrence
  SUBTYPE OF (assembly_component_usage);
    upper_usage : assembly_component_usage;
    next_usage  : next_assembly_usage_occurrence;
  UNIQUE
    ur1 : upper_usage, next_usage;
  WHERE
    wr1: (SELF :<>: upper_usage);
    wr2: (SELF\product_definition_relationship.
             relating_product_definition :=: upper_usage.
             relating_product_definition);
    wr3: (SELF\product_definition_relationship.
             related_product_definition :=: next_usage.
             related_product_definition);
    wr4: (upper_usage.related_product_definition :=: next_usage.
             relating_product_definition);
    wr5: (NOT ('STRUCTURAL_ANALYSIS_DESIGN.PROMISSORY_USAGE_OCCURRENCE'
             IN TYPEOF(upper_usage)));
END_ENTITY; -- specified_higher_usage_occurrence

ENTITY specified_state
  SUBTYPE OF (state);
END_ENTITY; -- specified_state

ENTITY spherical_point
  SUBTYPE OF (cartesian_point);
    r     : length_measure;
    theta : plane_angle_measure;
    phi   : plane_angle_measure;
  DERIVE
    SELF\cartesian_point.coordinates : LIST [1:3] OF length_measure := [
                                          r * SIN(theta) * COS(phi),r *
                                          SIN(theta) * SIN(phi),r * COS(
```



```
                                                  theta)];
  WHERE
    wr1: (r >= 0);
END_ENTITY; -- spherical_point

ENTITY spherical_surface
  SUBTYPE OF (elementary_surface);
    radius : positive_length_measure;
END_ENTITY; -- spherical_surface

ENTITY standard_uncertainty
  SUPERTYPE OF (expanded_uncertainty)
  SUBTYPE OF (uncertainty_qualifier);
    uncertainty_value : REAL;
END_ENTITY; -- standard_uncertainty

ENTITY state
  SUPERTYPE OF (ONEOF (specified_state,calculated_state,
      linearly_superimposed_state,output_request_state));
    state_id    : identifier;
    description : text;
END_ENTITY; -- state

ENTITY state_component
  SUBTYPE OF (state);
    state  : linearly_superimposed_state;
    factor : context_dependent_measure;
END_ENTITY; -- state_component

ENTITY state_definition
  SUPERTYPE OF (ONEOF (field_variable_definition,
      nodal_freedom_and_value_definition,element_nodal_freedom_actions,
      point_freedom_and_value_definition,
      curve_freedom_and_value_definition,
      surface_freedom_and_value_definition,
      solid_freedom_and_value_definition,
      linear_constraint_equation_element_value,
      single_point_constraint_element_values,analysis_message));
    defined_state : state;
END_ENTITY; -- state_definition

ENTITY state_relationship;
    name           : label;
    description    : text;
    relating_state : state;
    related_state  : state;
END_ENTITY; -- state_relationship

ENTITY state_with_harmonic
  SUBTYPE OF (state);
    symmetry : cylindrical_symmetry_control;
END_ENTITY; -- state_with_harmonic

ENTITY stationary_mass
  SUBTYPE OF (point_element_matrix);
    mass               : ARRAY [1:3] OF context_dependent_measure;
    moments_of_inertia : symmetric_tensor2_3d;
    coordinate_system  : fea_axis2_placement_3d;
    offset_vector      : ARRAY [1:3] OF context_dependent_measure;
END_ENTITY; -- stationary_mass
```

```
ENTITY structural_response_property
  SUBTYPE OF (property_definition);
END_ENTITY; -- structural_response_property

ENTITY structural_response_property_definition_representation
  SUBTYPE OF (property_definition_representation);
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.STRUCTURAL_RESPONSE_PROPERTY' IN
             TYPEOF(SELF\property_definition_representation.definition));
    wr2: ((('STRUCTURAL_ANALYSIS_DESIGN.FEA_MODEL' IN TYPEOF(SELF\
             property_definition_representation.used_representation)) AND
              ('STRUCTURAL_ANALYSIS_DESIGN.FEA_MODEL_DEFINITION' IN
             TYPEOF(SELF\property_definition_representation.definition.
             definition))) OR ((
             'STRUCTURAL_ANALYSIS_DESIGN.ELEMENT_REPRESENTATION' IN
             TYPEOF(SELF\property_definition_representation.
             used_representation)) AND (
             'STRUCTURAL_ANALYSIS_DESIGN.ELEMENT_DEFINITION' IN TYPEOF(
             SELF\property_definition_representation.definition.
             definition))) OR ((
             'STRUCTURAL_ANALYSIS_DESIGN.NODE_REPRESENTATION' IN TYPEOF(
             SELF\property_definition_representation.used_representation))
             AND ('STRUCTURAL_ANALYSIS_DESIGN.NODE_DEFINITION' IN TYPEOF(
             SELF\property_definition_representation.definition.
             definition))));
END_ENTITY; -- structural_response_property_definition_representation

ENTITY substructure_element_representation
  SUBTYPE OF (element_representation);
    model_ref              : fea_model;
    substructure_model_ref : fea_model;
  UNIQUE
    ur1 : model_ref, name;
END_ENTITY; -- substructure_element_representation

ENTITY substructure_node_reference;
    substructure_element_ref : substructure_element_representation;
    node_ref                 : node_representation;
END_ENTITY; -- substructure_node_reference

ENTITY substructure_node_relationship
  SUBTYPE OF (representation_relationship);
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.NODE_REPRESENTATION' IN TYPEOF(
             SELF\representation_relationship.rep_1));
    wr2: ('STRUCTURAL_ANALYSIS_DESIGN.NODE_REPRESENTATION' IN TYPEOF(
             SELF\representation_relationship.rep_2));
END_ENTITY; -- substructure_node_relationship

ENTITY supplied_part_relationship
  SUBTYPE OF (product_definition_relationship);
END_ENTITY; -- supplied_part_relationship

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_2d_element_basis;
      descriptor               : surface_2d_element_descriptor;
      variable                 : surface_element_variable;
      variable_order           : element_order;
      variable_shape_function  : shape_function;
      evaluation_points        : LIST [1:?] OF surface_element_location;
  END_ENTITY; -- surface_2d_element_basis

  ENTITY
surface_2d_element_boundary_constant_specified_surface_variable_value
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_variable;
      element_face      : surface_2d_face;
      coordinate_system : OPTIONAL surface_2d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; --
surface_2d_element_boundary_constant_specified_surface_variable_value

  ENTITY surface_2d_element_boundary_constant_specified_variable_value
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_aggregated_variable;
      element_face      : surface_2d_face;
      coordinate_system : OPTIONAL surface_2d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; --
surface_2d_element_boundary_constant_specified_variable_value

  ENTITY
surface_2d_element_boundary_edge_constant_specified_surface_variable_value
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_edge_variable;
      element_edge      : surface_2d_edge;
      coordinate_system : OPTIONAL surface_2d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; --
surface_2d_element_boundary_edge_constant_specified_surface_variable_value

  ENTITY surface_2d_element_boundary_edge_constant_specified_variable_value
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_edge_variable;
      element_edge      : surface_2d_edge;
      coordinate_system : OPTIONAL surface_2d_element_coordinate_system;
```

```
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
                coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
                state_definition.defined_state));
  END_ENTITY; --
surface_2d_element_boundary_edge_constant_specified_variable_value

  ENTITY
surface_2d_element_boundary_edge_location_point_surface_variable_values
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      basis                : BOOLEAN;
      values_and_locations : SET [1:?] OF
surface_2d_element_value_and_volume_location;
      variable             : boundary_edge_variable;
      element_edge         : surface_2d_edge;
    WHERE
      wr1: consistent_set_values(values_and_locations,variable);
      wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
                SELF\state_definition.defined_state));
  END_ENTITY; --
surface_2d_element_boundary_edge_location_point_surface_variable_values

  ENTITY surface_2d_element_boundary_edge_location_point_variable_values
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      basis                : BOOLEAN;
      values_and_locations : SET [1:?] OF
                                 surface_2d_element_value_and_location;
      variable             : boundary_edge_variable;
      element_edge         : surface_2d_edge;
    WHERE
      wr1: consistent_set_values(values_and_locations,variable);
      wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
                SELF\state_definition.defined_state));
  END_ENTITY; --
surface_2d_element_boundary_edge_location_point_variable_values

  ENTITY surface_2d_element_boundary_edge_nodal_specified_variable_values
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      values                 : LIST [1:?] OF field_value;
      additional_node_values : BOOLEAN;
      variable               : boundary_edge_variable;
      element_edge           : surface_2d_edge;
    WHERE
      wr1: consistent_list_values(values,variable);
      wr2: appropriate_list_value_existence(values,TYPEOF(SELF\
                state_definition.defined_state));
  END_ENTITY; --
surface_2d_element_boundary_edge_nodal_specified_variable_values

  ENTITY surface_2d_element_boundary_edge_whole_edge_variable_value
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_aggregated_variable;
      element_edge      : surface_2d_edge;
      coordinate_system : OPTIONAL surface_2d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
                coordinate_system);
      wr2: consistent_value(simple_value,variable);
```

```
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- surface_2d_element_boundary_edge_whole_edge_variable_value

  ENTITY surface_2d_element_boundary_location_point_surface_variable_values
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      basis                : BOOLEAN;
      values_and_locations : SET [1:?] OF
                                   surface_2d_element_value_and_location;
      variable             : boundary_variable;
      element_face         : surface_2d_face;
    WHERE
      wr1: consistent_set_values(values_and_locations,variable);
      wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
               SELF\state_definition.defined_state));
  END_ENTITY; --
surface_2d_element_boundary_location_point_surface_variable_values

  ENTITY surface_2d_element_boundary_nodal_specified_variable_values
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      values                 : LIST [1:?] OF field_value;
      additional_node_values : BOOLEAN;
      variable               : boundary_variable;
      element_face           : surface_2d_face;
    WHERE
      wr1: consistent_list_values(values,variable);
      wr2: appropriate_list_value_existence(values,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- surface_2d_element_boundary_nodal_specified_variable_values

  ENTITY surface_2d_element_boundary_whole_face_variable_value
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_aggregated_variable;
      element_face      : surface_2d_face;
      coordinate_system : OPTIONAL surface_2d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- surface_2d_element_boundary_whole_face_variable_value

  ENTITY surface_2d_element_constant_specified_variable_value
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : surface_element_variable;
      coordinate_system : OPTIONAL surface_2d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- surface_2d_element_constant_specified_variable_value

  ENTITY surface_2d_element_constant_specified_volume_variable_value
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : volume_variable;
```

```
      coordinate_system : OPTIONAL surface_2d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- surface_2d_element_constant_specified_volume_variable_value

  ENTITY surface_2d_element_field_variable_definition
    SUPERTYPE OF (ONEOF (
        surface_2d_element_location_point_volume_variable_values,
        surface_2d_element_location_point_variable_values,
        surface_2d_whole_element_variable_value,
        surface_2d_element_constant_specified_variable_value,
        surface_2d_element_constant_specified_volume_variable_value,
        surface_2d_element_nodal_specified_variable_values,
        surface_2d_element_boundary_location_point_surface_variable_values,
        surface_2d_element_boundary_whole_face_variable_value,

surface_2d_element_boundary_constant_specified_variable_value,surface_2d_elem
ent_boundary_constant_specified_surface_variable_value,

surface_2d_element_boundary_nodal_specified_variable_values,surface_2d_elemen
t_boundary_edge_location_point_surface_variable_values,
        surface_2d_element_boundary_edge_location_point_variable_values,
        surface_2d_element_boundary_edge_whole_edge_variable_value,

surface_2d_element_boundary_edge_constant_specified_variable_value,surface_2d
_element_boundary_edge_constant_specified_surface_variable_value,
        surface_2d_element_boundary_edge_nodal_specified_variable_values))
    SUBTYPE OF (field_variable_element_definition);
      element : surface_2d_element_output_reference;
  END_ENTITY; -- surface_2d_element_field_variable_definition

  ENTITY surface_2d_element_group
    SUBTYPE OF (element_group);
    WHERE
      wr1: (SIZEOF(QUERY ( item <* elements | ((NOT ((
               'STRUCTURAL_ANALYSIS_DESIGN.' +
               'AXISYMMETRIC_SURFACE_2D_ELEMENT_REPRESENTATION') IN TYPEOF(
               item))) AND (NOT (('STRUCTURAL_ANALYSIS_DESIGN.' +
               'PLANE_SURFACE_2D_ELEMENT_REPRESENTATION') IN TYPEOF(item))))
))
               = 0);
  END_ENTITY; -- surface_2d_element_group

  ENTITY surface_2d_element_integrated_matrix;
      descriptor              : surface_2d_element_descriptor;
      property_type           : surface_matrix_property_type;
      integration_description : text;
  END_ENTITY; -- surface_2d_element_integrated_matrix

  ENTITY surface_2d_element_integrated_matrix_with_definition
    SUBTYPE OF (surface_2d_element_integrated_matrix);
      integration_definition : surface_2d_element_integration;
  END_ENTITY; -- surface_2d_element_integrated_matrix_with_definition

  ENTITY surface_2d_element_integration;
      element_length : surface_2d_element_length_integration;
      section        : surface_section_integration;
```



```
  END_ENTITY; -- surface_2d_element_integration

  ENTITY surface_2d_element_length_integration_explicit;
      integration_positions_and_weights : SET [1:?] OF
                                        surface_position_weight;
  END_ENTITY; -- surface_2d_element_length_integration_explicit

  ENTITY surface_2d_element_length_integration_rule;
      integration_method : integration_rule;
      integration_order  : INTEGER;
  END_ENTITY; -- surface_2d_element_length_integration_rule

  ENTITY surface_2d_element_location_point_variable_values
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      basis               : BOOLEAN;
      values_and_locations : SET [1:?] OF
                              surface_2d_element_value_and_location;
      variable            : surface_element_variable;
    WHERE
      wr1: consistent_set_values(values_and_locations,variable);
      wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
               SELF\state_definition.defined_state));
  END_ENTITY; -- surface_2d_element_location_point_variable_values

  ENTITY surface_2d_element_location_point_volume_variable_values
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      basis               : BOOLEAN;
      values_and_locations : SET [1:?] OF
surface_2d_element_value_and_volume_location;
      variable            : volume_variable;
    WHERE
      wr1: consistent_set_values(values_and_locations,variable);
      wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
               SELF\state_definition.defined_state));
  END_ENTITY; -- surface_2d_element_location_point_volume_variable_values

  ENTITY surface_2d_element_nodal_specified_variable_values
    SUBTYPE OF (surface_2d_element_field_variable_definition);
      values                : LIST [1:?] OF field_value;
      additional_node_values : BOOLEAN;
      variable              : surface_element_variable;
    WHERE
      wr1: consistent_list_values(values,variable);
      wr2: appropriate_list_value_existence(values,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- surface_2d_element_nodal_specified_variable_values

  ENTITY surface_2d_element_value_and_location;
      simple_value      : field_value;
      location          : surface_element_location;
      coordinate_system : OPTIONAL surface_2d_state_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
  END_ENTITY; -- surface_2d_element_value_and_location

  ENTITY surface_2d_element_value_and_volume_location;
      simple_value      : field_value;
      location          : surface_volume_element_location;
      coordinate_system : OPTIONAL surface_2d_state_coordinate_system;
    WHERE
```

```
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
END_ENTITY; -- surface_2d_element_value_and_volume_location

ENTITY surface_2d_node_field_aggregated_variable_values
  SUBTYPE OF (surface_2d_node_field_variable_definition);
    simple_value      : field_value;
    variable          : volume_aggregated_variable;
    coordinate_system : OPTIONAL surface_2d_state_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- surface_2d_node_field_aggregated_variable_values

ENTITY surface_2d_node_field_section_variable_values
  SUBTYPE OF (surface_2d_node_field_variable_definition);
    simple_value      : field_value;
    variable          : surface_element_variable;
    location          : surface_section_element_location;
    coordinate_system : OPTIONAL surface_2d_state_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- surface_2d_node_field_section_variable_values

ENTITY surface_2d_node_field_variable_definition
  SUPERTYPE OF (ONEOF (surface_2d_node_field_section_variable_values,
      surface_2d_node_field_aggregated_variable_values))
  SUBTYPE OF (field_variable_node_definition);
END_ENTITY; -- surface_2d_node_field_variable_definition

ENTITY surface_2d_substructure_element_reference;
    substructure_element_ref : substructure_element_representation;
    element_ref              : surface_2d_element_representation;
END_ENTITY; -- surface_2d_substructure_element_reference

ENTITY surface_2d_whole_element_variable_value
  SUBTYPE OF (surface_2d_element_field_variable_definition);
    simple_value      : field_value;
    variable          : volume_aggregated_variable;
    coordinate_system : OPTIONAL surface_2d_element_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- surface_2d_whole_element_variable_value

ENTITY surface_3d_element_basis;
    descriptor              : surface_3d_element_descriptor;
    variable                : surface_element_variable;
```



```
      variable_order          : element_order;
      variable_shape_function : shape_function;
      evaluation_points       : LIST [1:?] OF surface_element_location;
  END_ENTITY; -- surface_3d_element_basis

  ENTITY
surface_3d_element_boundary_constant_specified_surface_variable_value
    SUBTYPE OF (surface_3d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_variable;
      element_face      : surface_3d_face;
      coordinate_system : OPTIONAL surface_3d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
                coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
                state_definition.defined_state));
  END_ENTITY; --
surface_3d_element_boundary_constant_specified_surface_variable_value

  ENTITY surface_3d_element_boundary_constant_specified_variable_value
    SUBTYPE OF (surface_3d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_aggregated_variable;
      element_face      : surface_3d_face;
      coordinate_system : OPTIONAL surface_3d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
                coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
                state_definition.defined_state));
  END_ENTITY; --
surface_3d_element_boundary_constant_specified_variable_value

  ENTITY
surface_3d_element_boundary_edge_constant_specified_surface_variable_value
    SUBTYPE OF (surface_3d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_edge_variable;
      element_edge      : surface_3d_edge;
      coordinate_system : OPTIONAL surface_3d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
                coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
                state_definition.defined_state));
  END_ENTITY; --
surface_3d_element_boundary_edge_constant_specified_surface_variable_value

  ENTITY surface_3d_element_boundary_edge_constant_specified_variable_value
    SUBTYPE OF (surface_3d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_edge_variable;
      element_edge      : surface_3d_edge;
      coordinate_system : OPTIONAL surface_3d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
                coordinate_system);
```

```
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
  END_ENTITY; --
surface_3d_element_boundary_edge_constant_specified_variable_value

  ENTITY
surface_3d_element_boundary_edge_location_point_surface_variable_values
    SUBTYPE OF (surface_3d_element_field_variable_definition);
      basis                : BOOLEAN;
      values_and_locations : SET [1:?] OF
surface_3d_element_value_and_volume_location;
      variable             : boundary_edge_variable;
      element_edge         : surface_3d_edge;
    WHERE
      wr1: consistent_set_values(values_and_locations,variable);
      wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
             SELF\state_definition.defined_state));
  END_ENTITY; --
surface_3d_element_boundary_edge_location_point_surface_variable_values

  ENTITY surface_3d_element_boundary_edge_location_point_variable_values
    SUBTYPE OF (surface_3d_element_field_variable_definition);
      basis                : BOOLEAN;
      values_and_locations : SET [1:?] OF
                                 surface_3d_element_value_and_location;
      variable             : boundary_edge_variable;
      element_edge         : surface_3d_edge;
    WHERE
      wr1: consistent_set_values(values_and_locations,variable);
      wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
             SELF\state_definition.defined_state));
  END_ENTITY; --
surface_3d_element_boundary_edge_location_point_variable_values

  ENTITY surface_3d_element_boundary_edge_nodal_specified_variable_values
    SUBTYPE OF (surface_3d_element_field_variable_definition);
      values                 : LIST [1:?] OF field_value;
      additional_node_values : BOOLEAN;
      variable               : boundary_edge_variable;
      element_edge           : surface_3d_edge;
    WHERE
      wr1: consistent_list_values(values,variable);
      wr2: appropriate_list_value_existence(values,TYPEOF(SELF\
             state_definition.defined_state));
  END_ENTITY; --
surface_3d_element_boundary_edge_nodal_specified_variable_values

  ENTITY surface_3d_element_boundary_edge_whole_edge_variable_value
    SUBTYPE OF (surface_3d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_aggregated_variable;
      element_edge      : surface_3d_edge;
      coordinate_system : OPTIONAL surface_3d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
  END_ENTITY; -- surface_3d_element_boundary_edge_whole_edge_variable_value
```



```
  ENTITY surface_3d_element_boundary_location_point_surface_variable_values
    SUBTYPE OF (surface_3d_element_field_variable_definition);
      basis                : BOOLEAN;
      values_and_locations : SET [1:?] OF
                               surface_3d_element_value_and_location;
      variable             : boundary_variable;
      element_face         : surface_3d_face;
    WHERE
      wr1: consistent_set_values(values_and_locations,variable);
      wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
              SELF\state_definition.defined_state));
  END_ENTITY; --
surface_3d_element_boundary_location_point_surface_variable_values

  ENTITY surface_3d_element_boundary_nodal_specified_variable_values
    SUBTYPE OF (surface_3d_element_field_variable_definition);
      values                 : LIST [1:?] OF field_value;
      additional_node_values : BOOLEAN;
      variable               : boundary_variable;
      element_face           : surface_3d_face;
    WHERE
      wr1: consistent_list_values(values,variable);
      wr2: appropriate_list_value_existence(values,TYPEOF(SELF\
              state_definition.defined_state));
  END_ENTITY; -- surface_3d_element_boundary_nodal_specified_variable_values

  ENTITY surface_3d_element_boundary_whole_face_variable_value
    SUBTYPE OF (surface_3d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_aggregated_variable;
      element_face      : surface_3d_face;
      coordinate_system : OPTIONAL surface_3d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
              coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
              state_definition.defined_state));
  END_ENTITY; -- surface_3d_element_boundary_whole_face_variable_value

  ENTITY surface_3d_element_constant_specified_variable_value
    SUBTYPE OF (surface_3d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : surface_element_variable;
      coordinate_system : OPTIONAL surface_3d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
              coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
              state_definition.defined_state));
  END_ENTITY; -- surface_3d_element_constant_specified_variable_value

  ENTITY surface_3d_element_constant_specified_volume_variable_value
    SUBTYPE OF (surface_3d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : volume_variable;
      coordinate_system : OPTIONAL surface_3d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
              coordinate_system);
```

```
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- surface_3d_element_constant_specified_volume_variable_value

ENTITY surface_3d_element_descriptor
  SUBTYPE OF (element_descriptor);
    purpose : SET [1:?] OF SET [1:?] OF surface_element_purpose;
    shape   : element_2d_shape;
END_ENTITY; -- surface_3d_element_descriptor

ENTITY surface_3d_element_field_integration_explicit;
    integration_positions_and_weights : SET [1:?] OF
                                        surface_position_weight;
END_ENTITY; -- surface_3d_element_field_integration_explicit

ENTITY surface_3d_element_field_integration_rule;
    integration_method : integration_rule;
    integration_order  : ARRAY [1:2] OF INTEGER;
END_ENTITY; -- surface_3d_element_field_integration_rule

ENTITY surface_3d_element_field_variable_definition
  SUPERTYPE OF (ONEOF (
      surface_3d_element_location_point_volume_variable_values,
      surface_3d_element_location_point_variable_values,
      surface_3d_whole_element_variable_value,
      surface_3d_element_constant_specified_variable_value,
      surface_3d_element_constant_specified_volume_variable_value,
      surface_3d_element_nodal_specified_variable_values,
      surface_3d_element_boundary_location_point_surface_variable_values,
      surface_3d_element_boundary_whole_face_variable_value,

surface_3d_element_boundary_constant_specified_variable_value,surface_3d_elem
ent_boundary_constant_specified_surface_variable_value,

surface_3d_element_boundary_nodal_specified_variable_values,surface_3d_elemen
t_boundary_edge_location_point_surface_variable_values,
      surface_3d_element_boundary_edge_location_point_variable_values,
      surface_3d_element_boundary_edge_whole_edge_variable_value,

surface_3d_element_boundary_edge_constant_specified_variable_value,surface_3d
_element_boundary_edge_constant_specified_surface_variable_value,
      surface_3d_element_boundary_edge_nodal_specified_variable_values))
  SUBTYPE OF (field_variable_element_definition);
    element : surface_3d_element_output_reference;
END_ENTITY; -- surface_3d_element_field_variable_definition

ENTITY surface_3d_element_group
  SUBTYPE OF (element_group);
  WHERE
    wr1: (SIZEOF(QUERY ( item <* elements | (NOT ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'SURFACE_3D_ELEMENT_REPRESENTATION') IN TYPEOF(item))) )) =
             0);
END_ENTITY; -- surface_3d_element_group

ENTITY surface_3d_element_integrated_matrix;
    descriptor              : surface_3d_element_descriptor;
    property_type           : surface_matrix_property_type;
    integration_description : text;
END_ENTITY; -- surface_3d_element_integrated_matrix
```



```
ENTITY surface_3d_element_integrated_matrix_with_definition
  SUBTYPE OF (surface_3d_element_integrated_matrix);
    integration_definition : surface_3d_element_integration;
END_ENTITY; -- surface_3d_element_integrated_matrix_with_definition

ENTITY surface_3d_element_integration;
    field   : surface_3d_element_field_integration;
    section : surface_section_integration;
END_ENTITY; -- surface_3d_element_integration

ENTITY surface_3d_element_location_point_variable_values
  SUBTYPE OF (surface_3d_element_field_variable_definition);
    basis                : BOOLEAN;
    values_and_locations : SET [1:?] OF
                              surface_3d_element_value_and_location;
    variable             : surface_element_variable;
  WHERE
    wr1: consistent_set_values(values_and_locations,variable);
    wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
             SELF\state_definition.defined_state));
END_ENTITY; -- surface_3d_element_location_point_variable_values

ENTITY surface_3d_element_location_point_volume_variable_values
  SUBTYPE OF (surface_3d_element_field_variable_definition);
    basis                : BOOLEAN;
    values_and_locations : SET [1:?] OF
surface_3d_element_value_and_volume_location;
    variable             : volume_variable;
  WHERE
    wr1: consistent_set_values(values_and_locations,variable);
    wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
             SELF\state_definition.defined_state));
END_ENTITY; -- surface_3d_element_location_point_volume_variable_values

ENTITY surface_3d_element_nodal_specified_variable_values
  SUBTYPE OF (surface_3d_element_field_variable_definition);
    values                 : LIST [1:?] OF field_value;
    additional_node_values : BOOLEAN;
    variable               : surface_element_variable;
  WHERE
    wr1: consistent_list_values(values,variable);
    wr2: appropriate_list_value_existence(values,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- surface_3d_element_nodal_specified_variable_values

ENTITY surface_3d_element_representation
  SUBTYPE OF (element_representation);
    model_ref          : fea_model_3d;
    element_descriptor : surface_3d_element_descriptor;
    property           : surface_element_property;
    material           : element_material;
  UNIQUE
    ur1 : model_ref, name;
  WHERE
    wr1: ((SIZEOF(QUERY ( item <* SELF\representation.items | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'PARAMETRIC_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM') IN
             TYPEOF(item)) )) + SIZEOF(QUERY ( item <* SELF\
             representation.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
             'CONSTANT_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM') IN TYPEOF(
             item)) )) + SIZEOF(QUERY ( item <* SELF\representation.items
```

```
             | ((’STRUCTURAL_ANALYSIS_DESIGN.’ +
             ’ALIGNED_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM’) IN TYPEOF(
             item)) ))) = 1);
    wr2: (SIZEOF(QUERY ( item1 <* material.properties | (SIZEOF(
             QUERY ( item2 <* item1\property_definition_representation.
             used_representation.items | (SIZEOF([
             ’STRUCTURAL_ANALYSIS_DESIGN.’ + ’FEA_LINEAR_ELASTICITY’,
             ’STRUCTURAL_ANALYSIS_DESIGN.’ + ’FEA_MASS_DENSITY’,
             ’STRUCTURAL_ANALYSIS_DESIGN.’ + ’FEA_AREA_DENSITY’,
             ’STRUCTURAL_ANALYSIS_DESIGN.’ +
             ’FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION’,
             ’STRUCTURAL_ANALYSIS_DESIGN.’ +
             ’FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION’,
             ’STRUCTURAL_ANALYSIS_DESIGN.’ + ’FEA_MOISTURE_ABSORPTION’,
             ’STRUCTURAL_ANALYSIS_DESIGN.’ +
             ’FEA_SHELL_MEMBRANE_STIFFNESS’,’STRUCTURAL_ANALYSIS_DESIGN.’
             + ’FEA_SHELL_BENDING_STIFFNESS’,
             ’STRUCTURAL_ANALYSIS_DESIGN.’ +
             ’FEA_SHELL_MEMBRANE_BENDING_COUPLING_STIFFNESS’,
             ’STRUCTURAL_ANALYSIS_DESIGN.’ + ’FEA_SHELL_SHEAR_STIFFNESS’]
             * TYPEOF(item2)) = 1) )) = 1) )) >= 1);
    wr3: (’STRUCTURAL_ANALYSIS_DESIGN.PARAMETRIC_REPRESENTATION_CONTEXT’
             IN TYPEOF(SELF\representation.context_of_items));
    fu1: required_2d_nodes(SELF\element_representation.node_list,
             element_descriptor.shape,element_descriptor\
             element_descriptor.topology_order);
END_ENTITY; -- surface_3d_element_representation

ENTITY surface_3d_element_value_and_location;
    simple_value      : field_value;
    location          : surface_element_location;
    coordinate_system : OPTIONAL surface_3d_state_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
END_ENTITY; -- surface_3d_element_value_and_location

ENTITY surface_3d_element_value_and_volume_location;
    simple_value      : field_value;
    location          : surface_volume_element_location;
    coordinate_system : OPTIONAL surface_3d_state_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
END_ENTITY; -- surface_3d_element_value_and_volume_location

ENTITY surface_3d_node_field_aggregated_variable_values
  SUBTYPE OF (surface_3d_node_field_variable_definition);
    simple_value      : field_value;
    variable          : volume_aggregated_variable;
    coordinate_system : OPTIONAL surface_3d_state_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- surface_3d_node_field_aggregated_variable_values
```

```
ENTITY surface_3d_node_field_section_variable_values
  SUBTYPE OF (surface_3d_node_field_variable_definition);
    simple_value      : field_value;
    variable          : surface_element_variable;
    location          : surface_section_element_location;
    coordinate_system : OPTIONAL surface_3d_state_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- surface_3d_node_field_section_variable_values

ENTITY surface_3d_node_field_variable_definition
  SUPERTYPE OF (ONEOF (surface_3d_node_field_section_variable_values,
      surface_3d_node_field_aggregated_variable_values))
  SUBTYPE OF (field_variable_node_definition);
END_ENTITY; -- surface_3d_node_field_variable_definition

ENTITY surface_3d_substructure_element_reference;
    substructure_element_ref : substructure_element_representation;
    element_ref              : surface_3d_element_representation;
END_ENTITY; -- surface_3d_substructure_element_reference

ENTITY surface_3d_whole_element_variable_value
  SUBTYPE OF (surface_3d_element_field_variable_definition);
    simple_value      : field_value;
    variable          : volume_aggregated_variable;
    coordinate_system : OPTIONAL surface_3d_element_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
             coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- surface_3d_whole_element_variable_value

ENTITY surface_constraint
  SUBTYPE OF (constraint_element);
    required_surface          : analysis_item_within_representation;
    coordinate_system         : fea_axis2_placement_3d;
    freedoms_and_coefficients : SET [1:?] OF freedom_and_coefficient;
    description               : text;
  WHERE
    wr1: (('STRUCTURAL_ANALYSIS_DESIGN.SURFACE' IN TYPEOF(
             required_surface.item)) OR (
             'STRUCTURAL_ANALYSIS_DESIGN.FACE_SURFACE' IN TYPEOF(
             required_surface.item)));
END_ENTITY; -- surface_constraint

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: (('STRUCTURAL_ANALYSIS_DESIGN.PCURVE' IN TYPEOF(
             associated_geometry[1])) OR (master_representation <>
             pcurve_s1));
    wr3: (('STRUCTURAL_ANALYSIS_DESIGN.PCURVE' IN TYPEOF(
             associated_geometry[2])) OR (master_representation <>
             pcurve_s2));
    wr4: (NOT ('STRUCTURAL_ANALYSIS_DESIGN.PCURVE' IN TYPEOF(curve_3d)));
END_ENTITY; -- surface_curve

ENTITY surface_element_location;
    coordinates : fea_parametric_point;
END_ENTITY; -- surface_element_location

ENTITY surface_element_property;
    property_id : identifier;
    description : text;
    section     : surface_section_field;
END_ENTITY; -- surface_element_property

ENTITY surface_freedom_action_definition
  SUBTYPE OF (surface_freedom_and_value_definition);
    action : action_type;
END_ENTITY; -- surface_freedom_action_definition

ENTITY surface_freedom_and_value_definition
  SUPERTYPE OF (ONEOF (surface_freedom_values,
      surface_freedom_action_definition))
  SUBTYPE OF (state_definition);
    required_surface   : analysis_item_within_representation;
    coordinate_system  : fea_axis2_placement_3d;
    degrees_of_freedom : freedoms_list;
    values             : LIST [1:?] OF measure_or_unspecified_value;
  WHERE
    wr1: (SIZEOF(degrees_of_freedom.freedoms) = SIZEOF(values));
    wr2: (('STRUCTURAL_ANALYSIS_DESIGN.SURFACE' IN TYPEOF(
             required_surface.item)) OR (
             'STRUCTURAL_ANALYSIS_DESIGN.FACE_SURFACE' IN TYPEOF(
             required_surface.item)));
END_ENTITY; -- surface_freedom_and_value_definition

ENTITY surface_freedom_values
  SUBTYPE OF (surface_freedom_and_value_definition);
END_ENTITY; -- surface_freedom_values

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, 1.0));
END_ENTITY; -- surface_of_revolution
```

```
ENTITY surface_patch
  SUBTYPE OF (founded_item);
    parent_surface : bounded_surface;
    u_transition   : transition_code;
    v_transition   : transition_code;
    u_sense        : BOOLEAN;
    v_sense        : BOOLEAN;
  INVERSE
    using_surfaces : BAG [1:?] OF rectangular_composite_surface FOR
                     segments;
  WHERE
    wr1: (NOT ('STRUCTURAL_ANALYSIS_DESIGN.CURVE_BOUNDED_SURFACE' IN
         TYPEOF(parent_surface)));
END_ENTITY; -- surface_patch

ENTITY surface_position_weight;
    integration_position : surface_element_location;
    integration_weight   : context_dependent_measure;
END_ENTITY; -- surface_position_weight

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 surface_section
  SUPERTYPE OF (ONEOF (uniform_surface_section,
      uniform_surface_section_layered));
    offset                      : measure_or_unspecified_value;
    non_structural_mass         : measure_or_unspecified_value;
    non_structural_mass_offset : measure_or_unspecified_value;
END_ENTITY; -- surface_section

ENTITY surface_section_element_location
  SUPERTYPE OF (ONEOF (surface_section_element_location_absolute,
      surface_section_element_location_dimensionless));
    above_material_discontinuity : LOGICAL;
END_ENTITY; -- surface_section_element_location

ENTITY surface_section_element_location_absolute
  SUBTYPE OF (surface_section_element_location);
    offset : context_dependent_measure;
END_ENTITY; -- surface_section_element_location_absolute

ENTITY surface_section_element_location_dimensionless
  SUBTYPE OF (surface_section_element_location);
    coordinate : LIST [1:1] OF parameter_value;
  WHERE
    wr1: valid_parametric_coordinate(coordinate);
END_ENTITY; -- surface_section_element_location_dimensionless

ENTITY surface_section_field
  SUPERTYPE OF (ONEOF (surface_section_field_constant,
      surface_section_field_varying));
END_ENTITY; -- surface_section_field

ENTITY surface_section_field_constant
  SUBTYPE OF (surface_section_field);
```

```
      definition : surface_section;
  END_ENTITY; -- surface_section_field_constant

  ENTITY surface_section_field_varying
    SUBTYPE OF (surface_section_field);
      definitions            : LIST [1:?] OF surface_section;
      additional_node_values : BOOLEAN;
  END_ENTITY; -- surface_section_field_varying

  ENTITY surface_section_integration_explicit;
      integration_positions_and_weights : SET [1:?] OF
surface_section_position_weight;
  END_ENTITY; -- surface_section_integration_explicit

  ENTITY surface_section_integration_rule;
      integration_method : integration_rule;
      integration_order  : INTEGER;
  END_ENTITY; -- surface_section_integration_rule

  ENTITY surface_section_position_weight;
      integration_position : surface_section_element_location;
      integration_weight   : context_dependent_measure;
  END_ENTITY; -- surface_section_position_weight

  ENTITY surface_volume_element_location;
      field_location   : surface_element_location;
      section_location : surface_section_element_location;
  END_ENTITY; -- surface_volume_element_location

  ENTITY swept_surface
    SUPERTYPE OF (ONEOF (surface_of_linear_extrusion,surface_of_revolution))
    SUBTYPE OF (surface);
      swept_curve : curve;
  END_ENTITY; -- swept_surface

  ENTITY symmetry_control
    SUPERTYPE OF (ONEOF (no_symmetry_control,cylindrical_symmetry_control));
  END_ENTITY; -- symmetry_control

  ENTITY system_and_freedom;
      matrix_coordinate_system :
directionally_explicit_element_coordinate_system;
      freedom                  : degree_of_freedom;
  END_ENTITY; -- system_and_freedom

  ENTITY tensor_representation_item
    SUBTYPE OF (representation_item);
      tensor_value : tensor_type;
  END_ENTITY; -- tensor_representation_item

  ENTITY thermodynamic_temperature_measure_with_unit
    SUBTYPE OF (measure_with_unit);
    WHERE
      wr1: ('STRUCTURAL_ANALYSIS_DESIGN.THERMODYNAMIC_TEMPERATURE_UNIT' IN
                TYPEOF(SELF\measure_with_unit.unit_component));
  END_ENTITY; -- thermodynamic_temperature_measure_with_unit

  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 thickness_laminate_definition
  SUBTYPE OF (product_definition);
  WHERE
    wr1: (SIZEOF(QUERY ( pdr <* USEDIN(SELF,
              'STRUCTURAL_ANALYSIS_DESIGN.' +
              'PRODUCT_DEFINITION_RELATIONSHIP.' +
              'RELATING_PRODUCT_DEFINITION') | ((
              'STRUCTURAL_ANALYSIS_DESIGN.' +
              'NEXT_ASSEMBLY_USAGE_OCCURRENCE') IN TYPEOF(pdr)) )) = 1);
END_ENTITY; -- thickness_laminate_definition

ENTITY time_measure_with_unit
  SUBTYPE OF (measure_with_unit);
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.TIME_UNIT' IN TYPEOF(SELF\
              measure_with_unit.unit_component));
END_ENTITY; -- time_measure_with_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,vertex_shell,
      wire_shell,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 transformation_with_derived_angle
  SUBTYPE OF (item_defined_transformation);
  DERIVE
    orientation_angle : plane_angle_measure := derive_angle(SELF\
                           item_defined_transformation.transform_item_1,
                           SELF\item_defined_transformation.
                           transform_item_2);
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.AXIS2_PLACEMENT_3D' IN TYPEOF(SELF
```

```
              \item_defined_transformation.transform_item_1));
      wr2: ('STRUCTURAL_ANALYSIS_DESIGN.AXIS2_PLACEMENT_3D' IN TYPEOF(SELF
              \item_defined_transformation.transform_item_2));
      wr3: ((SELF\item_defined_transformation.transform_item_1\
              axis2_placement_3d.p[3].direction_ratios[1] = SELF\
              item_defined_transformation.transform_item_2\
              axis2_placement_3d.p[3].direction_ratios[1]) AND (SELF\
              item_defined_transformation.transform_item_1\
              axis2_placement_3d.p[3].direction_ratios[2] = SELF\
              item_defined_transformation.transform_item_2\
              axis2_placement_3d.p[3].direction_ratios[2]) AND (SELF\
              item_defined_transformation.transform_item_1\
              axis2_placement_3d.p[3].direction_ratios[3] = SELF\
              item_defined_transformation.transform_item_2\
              axis2_placement_3d.p[3].direction_ratios[3]));
  END_ENTITY; -- transformation_with_derived_angle

  ENTITY trimmed_curve
    SUBTYPE OF (bounded_curve);
      basis_curve           : curve;
      trim_1                : SET [1:2] OF trimming_select;
      trim_2                : SET [1:2] OF trimming_select;
      sense_agreement       : BOOLEAN;
      master_representation : trimming_preference;
    WHERE
      wr1: ((HIINDEX(trim_1) = 1) OR (TYPEOF(trim_1[1]) <>
TYPEOF(trim_1[2])));
      wr2: ((HIINDEX(trim_2) = 1) OR (TYPEOF(trim_2[1]) <>
TYPEOF(trim_2[2])));
  END_ENTITY; -- trimmed_curve

  ENTITY type_qualifier;
      name : label;
  END_ENTITY; -- type_qualifier

  ENTITY unary_generic_expression
    ABSTRACT SUPERTYPE
    SUBTYPE OF (generic_expression);
      operand : generic_expression;
  END_ENTITY; -- unary_generic_expression

  ENTITY unary_numeric_expression
    ABSTRACT SUPERTYPE
    SUBTYPE OF (numeric_expression, unary_generic_expression);
      SELF\unary_generic_expression.operand : numeric_expression;
  END_ENTITY; -- unary_numeric_expression

  ENTITY uncertainty_measure_with_unit
    SUBTYPE OF (measure_with_unit);
      name        : label;
      description : text;
    WHERE
      wr1: valid_measure_value(SELF\measure_with_unit.value_component);
  END_ENTITY; -- uncertainty_measure_with_unit

  ENTITY uncertainty_qualifier
    SUPERTYPE OF (ONEOF (standard_uncertainty,qualitative_uncertainty));
      measure_name : label;
      description  : text;
  END_ENTITY; -- uncertainty_qualifier
```

```
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 uniform_surface_section
  SUBTYPE OF (surface_section);
    thickness         : context_dependent_measure;
    bending_thickness : measure_or_unspecified_value;
    shear_thickness   : measure_or_unspecified_value;
END_ENTITY; -- uniform_surface_section

ENTITY uniform_surface_section_layered
  SUBTYPE OF (surface_section);
END_ENTITY; -- uniform_surface_section_layered

ENTITY variable
  ABSTRACT SUPERTYPE OF (numeric_variable)
  SUBTYPE OF (generic_variable);
END_ENTITY; -- variable

ENTITY variable_semantics
  ABSTRACT SUPERTYPE;
END_ENTITY; -- variable_semantics

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 : 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 vertex_shell
  SUBTYPE OF (topological_representation_item);
    vertex_shell_extent : vertex_loop;
END_ENTITY; -- vertex_shell
```

```
  ENTITY volume_2d_element_basis;
      descriptor              : volume_2d_element_descriptor;
      variable                : volume_variable;
      variable_order          : element_order;
      variable_shape_function : shape_function;
      evaluation_points       : LIST [1:?] OF volume_element_location;
  END_ENTITY; -- volume_2d_element_basis

  ENTITY volume_2d_element_boundary_constant_specified_variable_value
    SUBTYPE OF (volume_2d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_variable;
      element_face      : volume_2d_face;
      coordinate_system : OPTIONAL volume_2d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- volume_2d_element_boundary_constant_specified_variable_value

  ENTITY
volume_2d_element_boundary_edge_constant_specified_volume_variable_value
    SUBTYPE OF (volume_2d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_edge_variable;
      element_edge      : volume_2d_edge;
      coordinate_system : OPTIONAL volume_2d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; --
volume_2d_element_boundary_edge_constant_specified_volume_variable_value

  ENTITY
volume_2d_element_boundary_edge_location_point_volume_variable_values
    SUBTYPE OF (volume_2d_element_field_variable_definition);
      basis                : BOOLEAN;
      values_and_locations : SET [1:?] OF
                                 volume_2d_element_value_and_location;
      variable             : boundary_edge_variable;
      element_edge         : volume_2d_edge;
    WHERE
      wr1: consistent_set_values(values_and_locations,variable);
      wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
               SELF\state_definition.defined_state));
  END_ENTITY; --
volume_2d_element_boundary_edge_location_point_volume_variable_values

  ENTITY volume_2d_element_boundary_edge_nodal_specified_variable_values
    SUBTYPE OF (volume_2d_element_field_variable_definition);
      values                 : LIST [1:?] OF field_value;
      additional_node_values : BOOLEAN;
      variable               : boundary_edge_variable;
      element_edge           : volume_2d_edge;
    WHERE
      wr1: consistent_list_values(values,variable);
```



```
      wr2: appropriate_list_value_existence(values,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; --
volume_2d_element_boundary_edge_nodal_specified_variable_values

  ENTITY volume_2d_element_boundary_edge_whole_edge_variable_value
    SUBTYPE OF (volume_2d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_aggregated_variable;
      element_edge      : volume_2d_edge;
      coordinate_system : OPTIONAL volume_2d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- volume_2d_element_boundary_edge_whole_edge_variable_value

  ENTITY volume_2d_element_boundary_location_point_variable_values
    SUBTYPE OF (volume_2d_element_field_variable_definition);
      basis                : BOOLEAN;
      values_and_locations : SET [1:?] OF
                                 volume_2d_element_value_and_location;
      variable             : boundary_variable;
      element_face         : volume_2d_face;
    WHERE
      wr1: consistent_set_values(values_and_locations,variable);
      wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
               SELF\state_definition.defined_state));
  END_ENTITY; -- volume_2d_element_boundary_location_point_variable_values

  ENTITY volume_2d_element_boundary_nodal_specified_variable_values
    SUBTYPE OF (volume_2d_element_field_variable_definition);
      values                 : LIST [1:?] OF field_value;
      variable               : boundary_variable;
      additional_node_values : BOOLEAN;
      element_face           : volume_2d_face;
    WHERE
      wr1: consistent_list_values(values,variable);
      wr2: appropriate_list_value_existence(values,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- volume_2d_element_boundary_nodal_specified_variable_values

  ENTITY volume_2d_element_boundary_whole_face_variable_value
    SUBTYPE OF (volume_2d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_aggregated_variable;
      element_face      : volume_2d_face;
      coordinate_system : OPTIONAL volume_2d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- volume_2d_element_boundary_whole_face_variable_value

  ENTITY volume_2d_element_constant_specified_variable_value
    SUBTYPE OF (volume_2d_element_field_variable_definition);
      simple_value      : field_value;
```

```
      variable          : volume_variable;
      coordinate_system : OPTIONAL volume_2d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- volume_2d_element_constant_specified_variable_value

  ENTITY volume_2d_element_field_integration_explicit;
      integration_positions_and_weights : SET [1:?] OF
                                        volume_position_weight;
  END_ENTITY; -- volume_2d_element_field_integration_explicit

  ENTITY volume_2d_element_field_integration_rule;
      integration_method : integration_rule;
      integration_order  : ARRAY [1:2] OF INTEGER;
  END_ENTITY; -- volume_2d_element_field_integration_rule

  ENTITY volume_2d_element_field_variable_definition
    SUPERTYPE OF (ONEOF (volume_2d_element_location_point_variable_values,
        volume_2d_whole_element_variable_value,
        volume_2d_element_constant_specified_variable_value,
        volume_2d_element_nodal_specified_variable_values,
        volume_2d_element_boundary_location_point_variable_values,
        volume_2d_element_boundary_whole_face_variable_value,
        volume_2d_element_boundary_constant_specified_variable_value,

volume_2d_element_boundary_nodal_specified_variable_values,volume_2d_element_
boundary_edge_location_point_volume_variable_values,

volume_2d_element_boundary_edge_whole_edge_variable_value,volume_2d_element_b
oundary_edge_constant_specified_volume_variable_value,
        volume_2d_element_boundary_edge_nodal_specified_variable_values))
    SUBTYPE OF (field_variable_element_definition);
      element : volume_2d_element_output_reference;
  END_ENTITY; -- volume_2d_element_field_variable_definition

  ENTITY volume_2d_element_group
    SUBTYPE OF (element_group);
    WHERE
      wr1: (SIZEOF(QUERY ( item <* elements | ((NOT ((
               'STRUCTURAL_ANALYSIS_DESIGN.' +
               'AXISYMMETRIC_VOLUME_2D_ELEMENT_REPRESENTATION') IN TYPEOF(
               item))) AND (NOT (('STRUCTURAL_ANALYSIS_DESIGN.' +
               'PLANE_VOLUME_2D_ELEMENT_REPRESENTATION') IN TYPEOF(item))))
))
               = 0);
  END_ENTITY; -- volume_2d_element_group

  ENTITY volume_2d_element_integrated_matrix;
      descriptor             : volume_2d_element_descriptor;
      property_type          : matrix_property_type;
      integration_description : text;
  END_ENTITY; -- volume_2d_element_integrated_matrix

  ENTITY volume_2d_element_integrated_matrix_with_definition
    SUBTYPE OF (volume_2d_element_integrated_matrix);
      integration_definition : volume_2d_element_field_integration;
  END_ENTITY; -- volume_2d_element_integrated_matrix_with_definition
```



```
ENTITY volume_2d_element_location_point_variable_values
  SUBTYPE OF (volume_2d_element_field_variable_definition);
    basis                : BOOLEAN;
    values_and_locations : SET [1:?] OF
                               volume_2d_element_value_and_location;
    variable             : volume_variable;
  WHERE
    wr1: consistent_set_values(values_and_locations,variable);
    wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
            SELF\state_definition.defined_state));
END_ENTITY; -- volume_2d_element_location_point_variable_values

ENTITY volume_2d_element_nodal_specified_variable_values
  SUBTYPE OF (volume_2d_element_field_variable_definition);
    values                 : LIST [1:?] OF field_value;
    additional_node_values : BOOLEAN;
    variable               : volume_variable;
  WHERE
    wr1: consistent_list_values(values,variable);
    wr2: appropriate_list_value_existence(values,TYPEOF(SELF\
            state_definition.defined_state));
END_ENTITY; -- volume_2d_element_nodal_specified_variable_values

ENTITY volume_2d_element_value_and_location;
    simple_value      : field_value;
    location          : volume_element_location;
    coordinate_system : OPTIONAL volume_2d_element_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
            coordinate_system);
END_ENTITY; -- volume_2d_element_value_and_location

ENTITY volume_2d_node_field_variable_definition
  SUBTYPE OF (field_variable_node_definition);
    simple_value      : field_value;
    variable          : volume_variable;
    coordinate_system : OPTIONAL volume_2d_element_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
            coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
            state_definition.defined_state));
END_ENTITY; -- volume_2d_node_field_variable_definition

ENTITY volume_2d_substructure_element_reference;
    substructure_element_ref : substructure_element_representation;
    element_ref              : volume_2d_element_representation;
END_ENTITY; -- volume_2d_substructure_element_reference

ENTITY volume_2d_whole_element_variable_value
  SUBTYPE OF (volume_2d_element_field_variable_definition);
    simple_value      : field_value;
    variable          : volume_aggregated_variable;
    coordinate_system : OPTIONAL volume_2d_element_coordinate_system;
```

```
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- volume_2d_whole_element_variable_value

  ENTITY volume_3d_element_basis;
      descriptor                : volume_3d_element_descriptor;
      variable                  : volume_variable;
      variable_order            : element_order;
      variable_shape_function : shape_function;
      evaluation_points         : LIST [1:?] OF volume_element_location;
  END_ENTITY; -- volume_3d_element_basis

  ENTITY volume_3d_element_boundary_constant_specified_variable_value
    SUBTYPE OF (volume_3d_element_field_variable_definition);
      simple_value        : field_value;
      variable            : boundary_variable;
      element_face        : volume_3d_face;
      coordinate_system : OPTIONAL volume_3d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- volume_3d_element_boundary_constant_specified_variable_value

  ENTITY
volume_3d_element_boundary_edge_constant_specified_volume_variable_value
    SUBTYPE OF (volume_3d_element_field_variable_definition);
      simple_value        : field_value;
      variable            : boundary_edge_variable;
      element_edge        : volume_3d_edge;
      coordinate_system : OPTIONAL volume_3d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; --
volume_3d_element_boundary_edge_constant_specified_volume_variable_value

  ENTITY
volume_3d_element_boundary_edge_location_point_volume_variable_values
    SUBTYPE OF (volume_3d_element_field_variable_definition);
      basis                   : BOOLEAN;
      values_and_locations : SET [1:?] OF
                                  volume_3d_element_value_and_location;
      variable                : boundary_edge_variable;
      element_edge            : volume_3d_edge;
    WHERE
      wr1: consistent_set_values(values_and_locations,variable);
      wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
               SELF\state_definition.defined_state));
  END_ENTITY; --
volume_3d_element_boundary_edge_location_point_volume_variable_values
```



```
  ENTITY volume_3d_element_boundary_edge_nodal_specified_variable_values
    SUBTYPE OF (volume_3d_element_field_variable_definition);
      values                 : LIST [1:?] OF field_value;
      additional_node_values : BOOLEAN;
      variable               : boundary_edge_variable;
      element_edge           : volume_3d_edge;
    WHERE
      wr1: consistent_list_values(values,variable);
      wr2: appropriate_list_value_existence(values,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; --
volume_3d_element_boundary_edge_nodal_specified_variable_values

  ENTITY volume_3d_element_boundary_edge_whole_edge_variable_value
    SUBTYPE OF (volume_3d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_aggregated_variable;
      element_edge      : volume_3d_edge;
      coordinate_system : OPTIONAL volume_3d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- volume_3d_element_boundary_edge_whole_edge_variable_value

  ENTITY volume_3d_element_boundary_location_point_variable_values
    SUBTYPE OF (volume_3d_element_field_variable_definition);
      basis                : BOOLEAN;
      values_and_locations : SET [1:?] OF
                                 volume_3d_element_value_and_location;
      variable             : boundary_variable;
      element_face         : volume_3d_face;
    WHERE
      wr1: consistent_set_values(values_and_locations,variable);
      wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
               SELF\state_definition.defined_state));
  END_ENTITY; -- volume_3d_element_boundary_location_point_variable_values

  ENTITY volume_3d_element_boundary_nodal_specified_variable_values
    SUBTYPE OF (volume_3d_element_field_variable_definition);
      values                 : LIST [1:?] OF field_value;
      additional_node_values : BOOLEAN;
      variable               : boundary_variable;
      element_face           : volume_3d_face;
    WHERE
      wr1: consistent_list_values(values,variable);
      wr2: appropriate_list_value_existence(values,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- volume_3d_element_boundary_nodal_specified_variable_values

  ENTITY volume_3d_element_boundary_whole_face_variable_value
    SUBTYPE OF (volume_3d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : boundary_aggregated_variable;
      element_face      : volume_3d_face;
      coordinate_system : OPTIONAL volume_3d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
```

```
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- volume_3d_element_boundary_whole_face_variable_value

  ENTITY volume_3d_element_constant_specified_variable_value
    SUBTYPE OF (volume_3d_element_field_variable_definition);
      simple_value      : field_value;
      variable          : volume_variable;
      coordinate_system : OPTIONAL volume_3d_element_coordinate_system;
    WHERE
      wr1: necessary_value_coordinate_system(simple_value,
               coordinate_system);
      wr2: consistent_value(simple_value,variable);
      wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
               state_definition.defined_state));
  END_ENTITY; -- volume_3d_element_constant_specified_variable_value

  ENTITY volume_3d_element_descriptor
    SUBTYPE OF (element_descriptor);
      purpose : SET [1:?] OF volume_element_purpose;
      shape   : volume_3d_element_shape;
  END_ENTITY; -- volume_3d_element_descriptor

  ENTITY volume_3d_element_field_integration_explicit;
      integration_positions_and_weights : SET [1:?] OF
                                        volume_position_weight;
  END_ENTITY; -- volume_3d_element_field_integration_explicit

  ENTITY volume_3d_element_field_integration_rule;
      integration_method : integration_rule;
      integration_order  : ARRAY [1:3] OF INTEGER;
  END_ENTITY; -- volume_3d_element_field_integration_rule

  ENTITY volume_3d_element_field_variable_definition
    SUPERTYPE OF (ONEOF (volume_3d_element_location_point_variable_values,
        volume_3d_whole_element_variable_value,
        volume_3d_element_constant_specified_variable_value,
        volume_3d_element_nodal_specified_variable_values,
        volume_3d_element_boundary_location_point_variable_values,
        volume_3d_element_boundary_whole_face_variable_value,
        volume_3d_element_boundary_constant_specified_variable_value,

volume_3d_element_boundary_nodal_specified_variable_values,volume_3d_element_
boundary_edge_location_point_volume_variable_values,

volume_3d_element_boundary_edge_whole_edge_variable_value,volume_3d_element_b
oundary_edge_constant_specified_volume_variable_value,
        volume_3d_element_boundary_edge_nodal_specified_variable_values))
    SUBTYPE OF (field_variable_element_definition);
      element : volume_3d_element_output_reference;
  END_ENTITY; -- volume_3d_element_field_variable_definition

  ENTITY volume_3d_element_group
    SUBTYPE OF (element_group);
    WHERE
      wr1: (SIZEOF(QUERY ( item <* elements | (NOT ((
               'STRUCTURAL_ANALYSIS_DESIGN.' +
               'VOLUME_3D_ELEMENT_REPRESENTATION') IN TYPEOF(item))) )) = 0);
  END_ENTITY; -- volume_3d_element_group
```



```
ENTITY volume_3d_element_integrated_matrix;
    descriptor             : volume_3d_element_descriptor;
    property_type          : matrix_property_type;
    integration_description : text;
END_ENTITY; -- volume_3d_element_integrated_matrix

ENTITY volume_3d_element_integrated_matrix_with_definition
  SUBTYPE OF (volume_3d_element_integrated_matrix);
    integration_definition : volume_3d_element_field_integration;
END_ENTITY; -- volume_3d_element_integrated_matrix_with_definition

ENTITY volume_3d_element_location_point_variable_values
  SUBTYPE OF (volume_3d_element_field_variable_definition);
    basis               : BOOLEAN;
    values_and_locations : SET [1:?] OF
                              volume_3d_element_value_and_location;
    variable            : volume_variable;
  WHERE
    wr1: consistent_set_values(values_and_locations,variable);
    wr2: appropriate_set_value_existence(values_and_locations,TYPEOF(
             SELF\state_definition.defined_state));
END_ENTITY; -- volume_3d_element_location_point_variable_values

ENTITY volume_3d_element_nodal_specified_variable_values
  SUBTYPE OF (volume_3d_element_field_variable_definition);
    values                 : LIST [1:?] OF field_value;
    additional_node_values : BOOLEAN;
    variable               : volume_variable;
  WHERE
    wr1: consistent_list_values(values,variable);
    wr2: appropriate_list_value_existence(values,TYPEOF(SELF\
             state_definition.defined_state));
END_ENTITY; -- volume_3d_element_nodal_specified_variable_values

ENTITY volume_3d_element_representation
  SUBTYPE OF (element_representation);
    model_ref          : fea_model_3d;
    element_descriptor : volume_3d_element_descriptor;
    material           : element_material;
  UNIQUE
    ur1 : model_ref, name;
  WHERE
    wr1: ((SIZEOF(QUERY ( item <* SELF\representation.items | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'PARAMETRIC_VOLUME_3D_ELEMENT_COORDINATE_SYSTEM') IN TYPEOF(
             item)) )) + SIZEOF(QUERY ( item <* SELF\representation.items
              | (('STRUCTURAL_ANALYSIS_DESIGN.' +
             'ARBITRARY_VOLUME_3D_ELEMENT_COORDINATE_SYSTEM') IN TYPEOF(
             item)) ))) = 1);
    wr2: (SIZEOF(QUERY ( item1 <* material.properties | (SIZEOF(
             QUERY ( item2 <* item1\property_definition_representation.
             used_representation.items | (SIZEOF([
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_LINEAR_ELASTICITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MASS_DENSITY',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
             'STRUCTURAL_ANALYSIS_DESIGN.' + 'FEA_MOISTURE_ABSORPTION'] *
              TYPEOF(item2)) = 1) )) = 1) )) >= 1);
    wr3: ('STRUCTURAL_ANALYSIS_DESIGN.PARAMETRIC_REPRESENTATION_CONTEXT'
```

```
            IN TYPEOF(SELF\representation.context_of_items));
    fu1: required_3d_nodes(SELF\element_representation.node_list,
            element_descriptor.shape,element_descriptor\
            element_descriptor.topology_order);
END_ENTITY; -- volume_3d_element_representation

ENTITY volume_3d_element_value_and_location;
    simple_value      : field_value;
    location          : volume_element_location;
    coordinate_system : OPTIONAL volume_3d_element_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
            coordinate_system);
END_ENTITY; -- volume_3d_element_value_and_location

ENTITY volume_3d_node_field_variable_definition
  SUBTYPE OF (field_variable_node_definition);
    simple_value      : field_value;
    variable          : volume_variable;
    coordinate_system : OPTIONAL volume_3d_element_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
            coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
            state_definition.defined_state));
END_ENTITY; -- volume_3d_node_field_variable_definition

ENTITY volume_3d_substructure_element_reference;
    substructure_element_ref : substructure_element_representation;
    element_ref              : volume_3d_element_representation;
END_ENTITY; -- volume_3d_substructure_element_reference

ENTITY volume_3d_whole_element_variable_value
  SUBTYPE OF (volume_3d_element_field_variable_definition);
    simple_value      : field_value;
    variable          : volume_aggregated_variable;
    coordinate_system : OPTIONAL volume_3d_element_coordinate_system;
  WHERE
    wr1: necessary_value_coordinate_system(simple_value,
            coordinate_system);
    wr2: consistent_value(simple_value,variable);
    wr3: appropriate_value_existence(simple_value,TYPEOF(SELF\
            state_definition.defined_state));
END_ENTITY; -- volume_3d_whole_element_variable_value

ENTITY volume_element_location;
    coordinates : fea_parametric_point;
END_ENTITY; -- volume_element_location

ENTITY volume_measure_with_unit
  SUBTYPE OF (measure_with_unit);
  WHERE
    wr1: ('STRUCTURAL_ANALYSIS_DESIGN.VOLUME_UNIT' IN TYPEOF(SELF\
            measure_with_unit.unit_component));
END_ENTITY; -- volume_measure_with_unit

ENTITY volume_position_weight;
    integration_position : volume_element_location;
    integration_weight   : context_dependent_measure;
END_ENTITY; -- volume_position_weight
```

```
ENTITY volume_unit
  SUBTYPE OF (named_unit);
  WHERE
    wr1: ((SELF\named_unit.dimensions.length_exponent = 3) 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; -- volume_unit

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

ENTITY whole_model_analysis_message
  SUPERTYPE OF (whole_model_modes_and_frequencies_analysis_message)
  SUBTYPE OF (analysis_message);
END_ENTITY; -- whole_model_analysis_message

ENTITY whole_model_modes_and_frequencies_analysis_message
  SUBTYPE OF (whole_model_analysis_message);
    mode      : count_measure;
    frequency : context_dependent_measure;
END_ENTITY; -- whole_model_modes_and_frequencies_analysis_message

ENTITY wire_shell
  SUBTYPE OF (topological_representation_item);
    wire_shell_extent : SET [1:?] OF loop;
  WHERE
    wr1: (NOT mixed_loop_type_set(wire_shell_extent));
END_ENTITY; -- wire_shell

RULE action_directive_requires_date_time FOR (action_directive,
             applied_date_and_time_assignment, applied_date_assignment);

WHERE
  wr1: (SIZEOF(QUERY ( ad <* action_directive | (NOT ((SIZEOF(
           QUERY ( adta <* applied_date_and_time_assignment | ((ad IN
           adta.items) AND (adta.role.name IN ['change date','start
date'])) ))
           + SIZEOF(QUERY ( ada <* applied_date_assignment | ((ad IN ada.
           items) AND (ada.role.name IN ['change date','start date'])) )))
           = 1)) )) = 0);

END_RULE; -- action_directive_requires_date_time

RULE action_request_requires_date_time FOR (versioned_action_request,
           applied_date_and_time_assignment, applied_date_assignment);

WHERE
  wr1: (SIZEOF(QUERY ( ar <* versioned_action_request | (NOT ((SIZEOF(
           QUERY ( adta <* applied_date_and_time_assignment | (ar IN adta
           .items) )) + SIZEOF(QUERY ( ada <* applied_date_assignment | (
           ar IN ada.items) ))) = 1)) )) = 0);

END_RULE; -- action_request_requires_date_time
```

```
RULE action_request_requires_person_organization FOR (
           versioned_action_request,
           applied_person_and_organization_assignment,
           applied_organization_assignment);

WHERE
  wr1: (SIZEOF(QUERY ( ar <* versioned_action_request | (NOT ((SIZEOF(
           QUERY ( apoa <* applied_person_and_organization_assignment | (
           (ar IN apoa.items) AND (apoa.role.name = 'request recipient'))
))
           + SIZEOF(QUERY ( aoa <* applied_organization_assignment | ((ar
           IN aoa.items) AND (aoa.role.name = 'request recipient')) )))
           >= 1)) )) = 0);

END_RULE; -- action_request_requires_person_organization

RULE approval_requires_approval_date_time FOR (approval,
           approval_date_time);

WHERE
  wr1: (SIZEOF(QUERY ( e <* approval | (NOT (SIZEOF(QUERY ( s <*
           approval_date_time | (e :=: s.dated_approval) )) = 1)) )) = 0);

END_RULE; -- approval_requires_approval_date_time

RULE approval_requires_approval_person_organization FOR (approval,
           approval_person_organization);

WHERE
  wr1: (SIZEOF(QUERY ( e <* approval | (NOT (SIZEOF(QUERY ( s <*
           approval_person_organization | (e :=: s.authorized_approval) ))
           >= 1)) )) = 0);

END_RULE; -- approval_requires_approval_person_organization

RULE approvals_are_assigned FOR (approval, approval_assignment);

WHERE
  wr1: (SIZEOF(QUERY ( app <* approval | (NOT (SIZEOF(QUERY ( aa <*
           approval_assignment | (app :=: aa.assigned_approval) )) >= 1))
))
           = 0);

END_RULE; -- approvals_are_assigned

RULE as_required_quantity FOR (measure_with_unit);

WHERE
  wr1: (SIZEOF(QUERY ( m <* measure_with_unit | ((
           'STRUCTURAL_ANALYSIS_DESIGN.DESCRIPTIVE_MEASURE' IN TYPEOF(m.
           value_component)) AND (NOT (m.value_component = 'as required')))
))
           = 0);

END_RULE; -- as_required_quantity

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 component_class_for_assembly_select FOR (
          composite_assembly_sequence_definition,
          next_assembly_usage_occurrence,
          product_related_product_category);

  LOCAL
    i     : INTEGER;
    j     : INTEGER;
    k     : INTEGER;
    nnauo : INTEGER;
    dkuhr : LOGICAL;
    rp    : product;
    nprpc : INTEGER;
  END_LOCAL;
  dkuhr := TRUE;
  REPEAT i := LOINDEX(composite_assembly_sequence_definition) TO
      HIINDEX(composite_assembly_sequence_definition) BY 1;
    nnauo := 0;
    REPEAT j := LOINDEX(next_assembly_usage_occurrence) TO HIINDEX(
        next_assembly_usage_occurrence) BY 1;
      IF composite_assembly_sequence_definition[i] =
          next_assembly_usage_occurrence[j].relating_product_definition
           THEN
        rp := next_assembly_usage_occurrence[j].
            related_product_definition.formation.of_product;
        nprpc := 0;
        REPEAT k := LOINDEX(product_related_product_category) TO
            HIINDEX(product_related_product_category) BY 1;
          IF (rp IN product_related_product_category[k].products) AND (
              product_related_product_category[k].name IN ['ply',
              'ply laminate','filament laminate','processed core',
              'composite assembly']) THEN
            nprpc := nprpc + 1;
          END_IF;
        END_REPEAT;
        IF nprpc = 1 THEN
          nnauo := nnauo + 1;
        ELSE
          dkuhr := FALSE;
          ESCAPE;
        END_IF;
      END_IF;
    END_REPEAT;
    IF dkuhr = FALSE THEN
      ESCAPE;
    END_IF;
    IF nnauo = 0 THEN
      dkuhr := FALSE;
      ESCAPE;
    END_IF;
  END_REPEAT;
```

```
WHERE
  wr1: dkuhr;

END_RULE; -- component_class_for_assembly_select

RULE configuration_item_requires_person_organization FOR (
           configuration_item, applied_person_and_organization_assignment,
           applied_organization_assignment);

WHERE
  wr1: (SIZEOF(QUERY ( ci <* configuration_item | (NOT ((SIZEOF(
           QUERY ( apoa <* applied_person_and_organization_assignment | (
           ci IN apoa.items) )) + SIZEOF(QUERY ( aoa <*
           applied_organization_assignment | (ci IN aoa.items) ))) = 1)) ))
           = 0);

END_RULE; -- configuration_item_requires_person_organization

RULE coordinated_assembly_and_shape FOR (next_assembly_usage_occurrence);

WHERE
  wr1: (SIZEOF(QUERY ( nauo <* next_assembly_usage_occurrence | (NOT
           assembly_shape_is_defined(nauo)) )) = 0);

END_RULE; -- coordinated_assembly_and_shape

RULE dependent_instantiable_approval_status FOR (approval_status);

WHERE
  wr1: (SIZEOF(QUERY ( ast <* approval_status | (NOT (SIZEOF(USEDIN(ast,
           ''')) >= 1)) )) = 0);

END_RULE; -- dependent_instantiable_approval_status

RULE dependent_instantiable_certification_type FOR (certification_type);

WHERE
  wr1: (SIZEOF(QUERY ( ct <* certification_type | (NOT (SIZEOF(USEDIN(ct,
           '')) >= 1)) )) = 0);

END_RULE; -- dependent_instantiable_certification_type

RULE dependent_instantiable_contract_type FOR (contract_type);

WHERE
  wr1: (SIZEOF(QUERY ( ct <* contract_type | (NOT (SIZEOF(USEDIN(ct,''))
           >= 1)) )) = 0);

END_RULE; -- dependent_instantiable_contract_type

RULE dependent_instantiable_date FOR (date);

WHERE
  wr1: (SIZEOF(QUERY ( dt <* date | (NOT (SIZEOF(USEDIN(dt,'')) >= 1)) ))
           = 0);

END_RULE; -- dependent_instantiable_date

RULE dependent_instantiable_date_and_time FOR (date_and_time);
```

```
WHERE
  wr1: (SIZEOF(QUERY ( dat <* date_and_time | (NOT (SIZEOF(USEDIN(dat,''))
           >= 1)) )) = 0);

END_RULE; -- dependent_instantiable_date_and_time

RULE dependent_instantiable_date_role FOR (date_role);

WHERE
  wr1: (SIZEOF(QUERY ( dr <* date_role | (NOT (SIZEOF(USEDIN(dr,'')) >=
           1)) )) = 0);

END_RULE; -- dependent_instantiable_date_role

RULE dependent_instantiable_date_time_role FOR (date_time_role);

WHERE
  wr1: (SIZEOF(QUERY ( dtr <* date_time_role | (NOT (SIZEOF(USEDIN(dtr,
           '')) >= 1)) )) = 0);

END_RULE; -- dependent_instantiable_date_time_role

RULE dependent_instantiable_document_type FOR (document_type);

WHERE
  wr1: (SIZEOF(QUERY ( dt <* document_type | (NOT (SIZEOF(USEDIN(dt,''))
           >= 1)) )) = 0);

END_RULE; -- dependent_instantiable_document_type

RULE dependent_instantiable_named_unit FOR (named_unit);

WHERE
  wr1: (SIZEOF(QUERY ( nu <* named_unit | (NOT (SIZEOF(USEDIN(nu,'')) >=
            1)) )) = 0);

END_RULE; -- dependent_instantiable_named_unit

RULE dependent_instantiable_organization_role FOR (organization_role);

WHERE
  wr1: (SIZEOF(QUERY ( orl <* organization_role | (NOT (SIZEOF(USEDIN(
           orl,'')) >= 1)) )) = 0);

END_RULE; -- dependent_instantiable_organization_role

RULE dependent_instantiable_parametric_representation_context FOR (
           parametric_representation_context);

WHERE
  wr1: (SIZEOF(QUERY ( prc <* parametric_representation_context | (NOT (
           SIZEOF(USEDIN(prc,'')) >= 1)) )) = 0);

END_RULE; -- dependent_instantiable_parametric_representation_context

RULE dependent_instantiable_person_and_organization_role FOR (
           person_and_organization_role);

WHERE
  wr1: (SIZEOF(QUERY ( paor <* person_and_organization_role | (NOT (
           SIZEOF(USEDIN(paor,'')) >= 1)) )) = 0);
```

```
END_RULE; -- dependent_instantiable_person_and_organization_role

RULE dependent_instantiable_representation_item FOR (representation_item);

WHERE
  wr1: (SIZEOF(QUERY ( ri <* representation_item | (NOT (SIZEOF(USEDIN(
           ri,'')) >= 1)) )) = 0);

END_RULE; -- dependent_instantiable_representation_item

RULE dependent_instantiable_security_classification_level FOR (
           security_classification_level);

WHERE
  wr1: (SIZEOF(QUERY ( scl <* security_classification_level | (NOT (
           SIZEOF(USEDIN(scl,'')) >= 1)) )) = 0);

END_RULE; -- dependent_instantiable_security_classification_level

RULE dependent_instantiable_shape_representation FOR (
           shape_representation);

WHERE
  wr1: (SIZEOF(QUERY ( sr <* shape_representation | (NOT (SIZEOF(USEDIN(
           sr,'')) >= 1)) )) = 0);

END_RULE; -- dependent_instantiable_shape_representation

RULE global_unit_assignment FOR (global_unit_assigned_context);

WHERE
  wr1: (SIZEOF(QUERY ( guac <* global_unit_assigned_context | (NOT (
           SIZEOF(guac.units) >= 2)) )) = 0);
  wr2: (SIZEOF(QUERY ( guac <* global_unit_assigned_context | (NOT ((
           SIZEOF(QUERY ( u <* guac.units | (
           'STRUCTURAL_ANALYSIS_DESIGN.LENGTH_UNIT' IN TYPEOF(u)) )) = 1)
           AND (SIZEOF(QUERY ( u <* guac.units | (
           'STRUCTURAL_ANALYSIS_DESIGN.PLANE_ANGLE_UNIT' IN TYPEOF(u)) ))
           = 1))) )) = 0);

END_RULE; -- global_unit_assignment

RULE pdu_requires_security_classification FOR (product_definition_usage,
           applied_security_classification_assignment);

WHERE
  wr1: (SIZEOF(QUERY ( e <* product_definition_usage | (NOT (SIZEOF(
           QUERY ( s <* applied_security_classification_assignment | (e
           IN s.items) )) = 1)) )) = 0);

END_RULE; -- pdu_requires_security_classification

RULE ply_reference FOR (ply_laminate_sequence_definition,
           next_assembly_usage_occurrence,
           product_related_product_category);
```

```
  LOCAL
    i     : INTEGER;
    j     : INTEGER;
    k     : INTEGER;
    nnauo : INTEGER;
    dkuhr : LOGICAL;
    rp    : product;
    nprpc : INTEGER;
  END_LOCAL;
  dkuhr := TRUE;
  REPEAT i := LOINDEX(ply_laminate_sequence_definition) TO HIINDEX(
      ply_laminate_sequence_definition) BY 1;
    nnauo := 0;
    REPEAT j := LOINDEX(next_assembly_usage_occurrence) TO HIINDEX(
        next_assembly_usage_occurrence) BY 1;
      IF ply_laminate_sequence_definition[i] =
          next_assembly_usage_occurrence[j].relating_product_definition
           THEN
        rp := next_assembly_usage_occurrence[j].
            related_product_definition.formation.of_product;
        nprpc := 0;
        REPEAT k := LOINDEX(product_related_product_category) TO
            HIINDEX(product_related_product_category) BY 1;
          IF (product_related_product_category[k].name = 'ply') AND (rp
              IN product_related_product_category[k].products) THEN
            nprpc := nprpc + 1;
          END_IF;
        END_REPEAT;
        IF nprpc = 1 THEN
          nnauo := nnauo + 1;
        ELSE
          dkuhr := FALSE;
          ESCAPE;
        END_IF;
      END_IF;
    END_REPEAT;
    IF dkuhr = FALSE THEN
      ESCAPE;
    END_IF;
    IF nnauo = 0 THEN
      dkuhr := FALSE;
      ESCAPE;
    END_IF;
  END_REPEAT;

WHERE
  wr1: dkuhr;

END_RULE; -- ply_reference

RULE ply_stock_material_select FOR (product_related_product_category,
           make_from_usage_option);

  LOCAL
    i     : INTEGER;
    j     : INTEGER;
    k     : INTEGER;
    dkuhr : LOGICAL;
    kp    : INTEGER;
    rp    : product;
    nprpc : INTEGER;
```

```
    nmfuo : INTEGER;
  END_LOCAL;
  dkuhr := TRUE;
  REPEAT kp := LOINDEX(product_related_product_category) TO HIINDEX(
      product_related_product_category) BY 1;
    IF product_related_product_category[kp].name = 'ply' THEN
      REPEAT i := LOINDEX(product_related_product_category[kp].products)
           TO HIINDEX(product_related_product_category[kp].products) BY 1;
        nmfuo := 0;
        REPEAT j := LOINDEX(make_from_usage_option) TO HIINDEX(
            make_from_usage_option) BY 1;
          IF product_related_product_category[kp].products[i] =
              make_from_usage_option[j].relating_product_definition.
              formation.of_product THEN
            rp := make_from_usage_option[j].related_product_definition.
                formation.of_product;
            REPEAT k := LOINDEX(product_related_product_category) TO
                HIINDEX(product_related_product_category) BY 1;
              IF (rp IN product_related_product_category[k].products)
                  AND (product_related_product_category[k].name IN [
                  'isotropic material','filament assembly',
                  'discontinuous fiber assembly']) THEN
                nprpc := nprpc + 1;
              END_IF;
            END_REPEAT;
            IF nprpc = 1 THEN
              nmfuo := nmfuo + 1;
            ELSE
              dkuhr := FALSE;
              ESCAPE;
            END_IF;
          END_IF;
        END_REPEAT;
        IF dkuhr = FALSE THEN
          ESCAPE;
        END_IF;
        IF nmfuo <> 1 THEN
          dkuhr := FALSE;
          ESCAPE;
        END_IF;
      END_REPEAT;
    END_IF;
  END_REPEAT;

WHERE
  wr1: dkuhr;

END_RULE; -- ply_stock_material_select

RULE product_concept_requires_configuration_item FOR (product_concept,
           configuration_item);

WHERE
  wr1: (SIZEOF(QUERY ( pc <* product_concept | (NOT (SIZEOF(
           QUERY ( ci <* configuration_item | (pc :=: ci.item_concept) ))
           >= 1)) )) = 0);

END_RULE; -- product_concept_requires_configuration_item

RULE product_definition_requires_date_time FOR (product_definition,
           applied_date_and_time_assignment, applied_date_assignment);
```



```
  WHERE
    wr1: (SIZEOF(QUERY ( pd <* product_definition | (NOT ((SIZEOF(
           QUERY ( adta <* applied_date_and_time_assignment | (pd IN adta
           .items) )) + SIZEOF(QUERY ( ada <* applied_date_assignment | (
           pd IN ada.items) ))) = 1)) )) = 0);

  END_RULE; -- product_definition_requires_date_time

  RULE product_definition_requires_person_organization FOR (
           product_definition, applied_person_and_organization_assignment,
           applied_organization_assignment);

  WHERE
    wr1: (SIZEOF(QUERY ( pd <* product_definition | (NOT ((SIZEOF(
           QUERY ( apoa <* applied_person_and_organization_assignment | (
           pd IN apoa.items) )) + SIZEOF(QUERY ( aoa <*
           applied_organization_assignment | (pd IN aoa.items) ))) = 1)) ))
           = 0);

  END_RULE; -- product_definition_requires_person_organization

  RULE product_requires_person_organization FOR (product,
           applied_person_and_organization_assignment,
           applied_organization_assignment);

  WHERE
    wr1: (SIZEOF(QUERY ( prod <* product | (NOT ((SIZEOF(QUERY ( apoa <*
           applied_person_and_organization_assignment | (prod IN apoa.
           items) )) + SIZEOF(QUERY ( aoa <*
           applied_organization_assignment | (prod IN aoa.items) ))) = 1))
))
           = 0);

  END_RULE; -- product_requires_person_organization

  RULE product_requires_product_category FOR (product,
           product_related_product_category);

  WHERE
    wr1: (SIZEOF(QUERY ( prod <* product | (NOT (SIZEOF(QUERY ( prpc <*
           product_related_product_category | ((prod IN prpc.products)
           AND (prpc.name IN ['assembly','inseparable assembly','detail',
           'customer furnished equipment'])) )) = 1)) )) = 0);

  END_RULE; -- product_requires_product_category

  RULE product_requires_version FOR (product, product_definition_formation);

  WHERE
    wr1: (SIZEOF(QUERY ( prod <* product | (NOT (SIZEOF(QUERY ( pdf <*
           product_definition_formation | (prod :=: pdf.of_product) )) >=
            1)) )) = 0);

  END_RULE; -- product_requires_version

  RULE product_version_requires_person_organization FOR (
           product_definition_formation,
           applied_person_and_organization_assignment,
           applied_organization_assignment);
```

```
WHERE
  wr1: (SIZEOF(QUERY ( pdf <* product_definition_formation | (NOT ((
           SIZEOF(QUERY ( apoa <*
           applied_person_and_organization_assignment | ((pdf IN apoa.
           items) AND (apoa.role.name = 'creator')) )) + SIZEOF(
           QUERY ( aoa <* applied_organization_assignment | ((pdf IN aoa.
           items) AND (aoa.role.name = 'creator')) ))) = 1)) )) = 0);
  wr2: (SIZEOF(QUERY ( pdf <* product_definition_formation | (NOT ((
           SIZEOF(QUERY ( apoa <*
           applied_person_and_organization_assignment | ((pdf IN apoa.
           items) AND (apoa.role.name IN ['analysis supplier',
           'design supplier','part supplier'])) )) + SIZEOF(
           QUERY ( aoa <* applied_organization_assignment | ((pdf IN aoa.
           items) AND (aoa.role.name IN ['analysis supplier',
           'design supplier','part supplier'])) ))) >= 1)) )) = 0);

END_RULE; -- product_version_requires_person_organization

RULE product_version_requires_security_classification FOR (
           product_definition_formation,
           applied_security_classification_assignment);

WHERE
  wr1: (SIZEOF(QUERY ( e <* product_definition_formation | (NOT (SIZEOF(
           QUERY ( s <* applied_security_classification_assignment | (e
           IN s.items) )) = 1)) )) = 0);

END_RULE; -- product_version_requires_security_classification

RULE restrict_action_request_status FOR (action_request_status);

WHERE
  wr1: (SIZEOF(QUERY ( ars <* action_request_status | (NOT (ars.status
           IN ['proposed','in work','issued','hold'])) )) = 0);

END_RULE; -- restrict_action_request_status

RULE restrict_approval_status FOR (approval_status);

WHERE
  wr1: (SIZEOF(QUERY ( ast <* approval_status | (NOT (ast.name IN [
           'approved','not yet approved','disapproved','withdrawn'])) ))
           = 0);

END_RULE; -- restrict_approval_status

RULE restrict_date_time_role FOR (date_time_role, date_role);

WHERE
  wr1: ((SIZEOF(QUERY ( dtr <* date_time_role | (NOT (dtr.name IN [
           'change date','creation date','request date','release date',
           'start date','earliest end date','latest end date',
           'contract date','sign off date','certification date',
           'classification date','declassification date'])) )) + SIZEOF(
           QUERY ( dr <* date_role | (NOT (dr.name IN ['change date',
           'creation date','request date','release date','start date',
           'earliest end date','latest end date','contract date',
           'sign off date','certification date','classification date',
           'declassification date'])) ))) = 0);

END_RULE; -- restrict_date_time_role
```



```
RULE restrict_person_organization_role FOR (person_and_organization_role,
          organization_role);

WHERE
  wr1: ((SIZEOF(QUERY ( porl <* person_and_organization_role | (NOT (
          porl.name IN ['request recipient','initiator','part supplier',
          'design supplier','analysis supplier','creator','design owner',
          'analysis owner','contractor','configuration manager',
          'classification officer'])) )) + SIZEOF(QUERY ( orl <*
          organization_role | (NOT (orl.name IN ['request recipient',
          'initiator','part supplier','design supplier',
          'analysis supplier','creator','design owner','analysis owner',
          'contractor','configuration manager','classification officer']))
)))
          = 0);

END_RULE; -- restrict_person_organization_role

RULE restrict_product_category_value FOR (
          product_related_product_category);

WHERE
  wr1: (SIZEOF(QUERY ( prpc <* product_related_product_category | (NOT (
          prpc.name IN ['assembly','detail',
          'customer furnished equipment','inseparable assembly',
          'standard part','linear static analysis',
          'linear modes and frequencies analysis','cast','coined',
          'drawn','extruded','forged','formed','machined','molded',
          'rolled','sheared','anisotropic material','composite assembly',
          'discontinuous fiber assembly','filament assembly',
          'filament laminate','isotropic material','ply','ply laminate',
          'ply piece','processed core','stock core'])) )) = 0);

END_RULE; -- restrict_product_category_value

RULE restrict_product_definition_context FOR (product_definition_context);

WHERE
  wr1: (SIZEOF(QUERY ( pdc <* product_definition_context | (NOT (pdc.
          life_cycle_stage IN ['analysis','design'])) )) = 0);

END_RULE; -- restrict_product_definition_context

RULE restrict_security_classification_level FOR (
          security_classification_level);

WHERE
  wr1: (SIZEOF(QUERY ( scl <* security_classification_level | (NOT (scl.
          name IN ['unclassified','classified','proprietary',
          'confidential','secret','top secret'])) )) = 0);

END_RULE; -- restrict_security_classification_level

RULE stock_material_reference FOR (percentage_ply_definition,
          make_from_usage_option, product_related_product_category);

  LOCAL
    i     : INTEGER;
    j     : INTEGER;
    k     : INTEGER;
```

```
    dkuhr : LOGICAL;
    rp    : product;
    nprpc : INTEGER;
    nmfuo : INTEGER;
  END_LOCAL;
  dkuhr := TRUE;
  REPEAT i := LOINDEX(percentage_ply_definition) TO HIINDEX(
      percentage_ply_definition) BY 1;
    nmfuo := 0;
    REPEAT j := LOINDEX(make_from_usage_option) TO HIINDEX(
        make_from_usage_option) BY 1;
      IF percentage_ply_definition[i] = make_from_usage_option[j].
          relating_product_definition THEN
        rp := make_from_usage_option[j].related_product_definition.
            formation.of_product;
        nprpc := 0;
        REPEAT k := LOINDEX(product_related_product_category) TO
            HIINDEX(product_related_product_category) BY 1;
          IF (rp IN product_related_product_category[k].products) AND (
              product_related_product_category[k].name IN [
              'anisotropic material','isotropic material','stock core',
              'filament assembly','discontinuous fiber assembly']) THEN
            nprpc := nprpc + 1;
          END_IF;
        END_REPEAT;
        IF nprpc = 1 THEN
          nmfuo := nmfuo + 1;
        ELSE
          dkuhr := FALSE;
          ESCAPE;
        END_IF;
      END_IF;
    END_REPEAT;
    IF dkuhr = FALSE THEN
      ESCAPE;
    END_IF;
    IF nmfuo = 0 THEN
      dkuhr := FALSE;
      ESCAPE;
    END_IF;
  END_REPEAT;

WHERE
  wr1: dkuhr;

END_RULE; -- stock_material_reference

RULE subtype_mandatory_action FOR (action);

WHERE
  wr1: (SIZEOF(QUERY ( act <* action | (NOT (
          'STRUCTURAL_ANALYSIS_DESIGN.DIRECTED_ACTION' IN TYPEOF(act))) ))
          = 0);

END_RULE; -- subtype_mandatory_action

RULE subtype_mandatory_shape_representation FOR (shape_representation);

WHERE
  wr1: (SIZEOF(QUERY ( sr <* shape_representation | (NOT ((SIZEOF([
          'STRUCTURAL_ANALYSIS_DESIGN.' +
```

```
             'GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_REPRESENTATION',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION',
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'MANIFOLD_SURFACE_SHAPE_REPRESENTATION',
             'STRUCTURAL_ANALYSIS_DESIGN.POINT_PATH',
             'STRUCTURAL_ANALYSIS_DESIGN.POINT_AND_VECTOR',

'STRUCTURAL_ANALYSIS_DESIGN.POINT_REPRESENTATION','STRUCTURAL_ANALYSIS_DESIGN
.FACETED_BREP_SHAPE_REPRESENTATION','STRUCTURAL_ANALYSIS_DESIGN.ADVANCED_BREP
_SHAPE_REPRESENTATION',
             'STRUCTURAL_ANALYSIS_DESIGN.SHAPE_DIMENSION_REPRESENTATION',
             'STRUCTURAL_ANALYSIS_DESIGN.BEVELED_SHEET_REPRESENTATION',
             'STRUCTURAL_ANALYSIS_DESIGN.COMPOSITE_SHEET_REPRESENTATION'] *
              TYPEOF(sr)) > 0) OR (SIZEOF(QUERY ( it <* sr\representation.
             items | (NOT (SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.PLACEMENT',
             'STRUCTURAL_ANALYSIS_DESIGN.MAPPED_ITEM'] * TYPEOF(it)) = 1)) ))
             = 0) OR (SIZEOF(QUERY ( sdr <* QUERY ( pdr <* USEDIN(sr,
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTATION') | ((
             'STRUCTURAL_ANALYSIS_DESIGN.' +
             'SHAPE_DEFINITION_REPRESENTATION') IN TYPEOF(pdr)) ) | (NOT (
             SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.SHAPE_ASPECT',
             'STRUCTURAL_ANALYSIS_DESIGN.SHAPE_ASPECT_RELATIONSHIP'] *
             TYPEOF(sdr.definition.definition)) = 1)) )) = 0))) )) = 0);

  END_RULE; -- subtype_mandatory_shape_representation

  RULE thickness_laminate_table_component_select FOR (
             thickness_laminate_definition, next_assembly_usage_occurrence,
             product_related_product_category);

    LOCAL
      i     : INTEGER;
      j     : INTEGER;
      k     : INTEGER;
      nnauo : INTEGER;
      dkuhr : LOGICAL;
      rp    : product;
      nprpc : INTEGER;
    END_LOCAL;
    dkuhr := TRUE;
    REPEAT i := LOINDEX(thickness_laminate_definition) TO HIINDEX(
        thickness_laminate_definition) BY 1;
      nnauo := 0;
      REPEAT j := LOINDEX(next_assembly_usage_occurrence) TO HIINDEX(
          next_assembly_usage_occurrence) BY 1;
        IF thickness_laminate_definition[i] =
            next_assembly_usage_occurrence[j].relating_product_definition
             THEN
          rp := next_assembly_usage_occurrence[j].
              related_product_definition.formation.of_product;
          nprpc := 0;
          REPEAT k := LOINDEX(product_related_product_category) TO
              HIINDEX(product_related_product_category) BY 1;
            IF (rp IN product_related_product_category[k].products) AND (
                product_related_product_category[k].name IN ['ply',
```

```
              'filament laminate','processed core']) THEN
            nprpc := nprpc + 1;
          END_IF;
        END_REPEAT;
        IF nprpc = 1 THEN
          nnauo := nnauo + 1;
        ELSE
          dkuhr := FALSE;
          ESCAPE;
        END_IF;
      END_IF;
    END_REPEAT;
    IF dkuhr = FALSE THEN
      ESCAPE;
    END_IF;
    IF nnauo <> 1 THEN
      dkuhr := FALSE;
      ESCAPE;
    END_IF;
  END_REPEAT;

WHERE
  wr1: dkuhr;

END_RULE; -- thickness_laminate_table_component_select

RULE unique_version_change_order_rule FOR (action_directive);

WHERE
  wr1: (SIZEOF(QUERY ( ad <* action_directive | (NOT
           unique_version_change_order(ad)) )) = 0);

END_RULE; -- unique_version_change_order_rule

RULE versioned_action_request_requires_status FOR (
           versioned_action_request, action_request_status);

WHERE
  wr1: (SIZEOF(QUERY ( ar <* versioned_action_request | (NOT (SIZEOF(
           QUERY ( ars <* action_request_status | (ar :=: ars.
           assigned_request) )) = 1)) )) = 0);

END_RULE; -- versioned_action_request_requires_status

FUNCTION acyclic(
             arg1: generic_expression;
             arg2: SET OF generic_expression
    ): BOOLEAN;

  LOCAL
    result : BOOLEAN;
  END_LOCAL;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.SIMPLE_GENERIC_EXPRESSION' IN TYPEOF(
      arg1) THEN
    RETURN(TRUE);
  END_IF;
  IF arg1 IN arg2 THEN
    RETURN(FALSE);
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.UNARY_GENERIC_EXPRESSION' IN TYPEOF(
      arg1) THEN
```



```
    RETURN(acyclic(arg1\unary_generic_expression.operand,arg2 + [arg1]));
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.BINARY_GENERIC_EXPRESSION' IN TYPEOF(
      arg1) THEN
    RETURN(acyclic(arg1\binary_generic_expression.operands[1],arg2 + [
        arg1]) AND acyclic(arg1\binary_generic_expression.operands[2],
        arg2 + [arg1]));
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.MULTIPLE_ARITY_GENERIC_EXPRESSION' IN
      TYPEOF(arg1) THEN
    result := TRUE;
    REPEAT i := 1 TO SIZEOF(arg1\multiple_arity_generic_expression.
        operands) BY 1;
      result := result AND acyclic(arg1\
          multiple_arity_generic_expression.operands[i],arg2 + [arg1]);
    END_REPEAT;
    RETURN(result);
  END_IF;

END_FUNCTION; -- acyclic

FUNCTION acyclic_curve_replica(
             rep: curve_replica;
             parent: curve
    ): BOOLEAN;
  IF NOT ('STRUCTURAL_ANALYSIS_DESIGN.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
    i : INTEGER;
    x : SET OF representation_item;
    y : SET OF representation_item;
  END_LOCAL;
  x := QUERY ( z <* children_set | (
       'STRUCTURAL_ANALYSIS_DESIGN.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],'')) | (
          'STRUCTURAL_ANALYSIS_DESIGN.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 ('STRUCTURAL_ANALYSIS_DESIGN.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
    ): LOGICAL;

  LOCAL
    i              : INTEGER;
    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,'STRUCTURAL_ANALYSIS_DESIGN.'
      + '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_product_definition_relationship(
             relation: product_definition_relationship;
             relatives: SET [1:?] OF product_definition;
             specific_relation: STRING
    ): LOGICAL;

  LOCAL
    x : SET OF product_definition_relationship;
  END_LOCAL;
  IF relation.relating_product_definition IN relatives THEN
    RETURN(FALSE);
  END_IF;
  x := QUERY ( pd <* bag_to_set(USEDIN(relation.
      relating_product_definition,'STRUCTURAL_ANALYSIS_DESIGN.' +
      'PRODUCT_DEFINITION_RELATIONSHIP.' + 'RELATED_PRODUCT_DEFINITION'))
       | (specific_relation IN TYPEOF(pd)) );
  REPEAT i := 1 TO HIINDEX(x) BY 1;
    IF NOT acyclic_product_definition_relationship(x[i],relatives +
        relation.relating_product_definition,specific_relation) THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  RETURN(TRUE);

END_FUNCTION; -- acyclic_product_definition_relationship

FUNCTION acyclic_surface_replica(
             rep: surface_replica;
             parent: surface
    ): BOOLEAN;
  IF NOT ('STRUCTURAL_ANALYSIS_DESIGN.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 applied_date_correlation(
             e: applied_date_assignment
    ): BOOLEAN;

  LOCAL
    dt_role : STRING;
  END_LOCAL;
  dt_role := e\date_assignment.role.name;
  CASE dt_role OF
    'change date'         :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (
          'STRUCTURAL_ANALYSIS_DESIGN.ACTION_DIRECTIVE' IN TYPEOF(x)) ))
           THEN
        RETURN(FALSE);
      END_IF;
    'creation date'       :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'PRODUCT_DEFINITION') IN TYPEOF(x)) )) THEN
```

```
        RETURN(FALSE);
      END_IF;
    'earliest end date'     :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | ('STRUCTURAL_ANALYSIS_DESIGN.RETENTION'
          IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'latest end date'       :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | ('STRUCTURAL_ANALYSIS_DESIGN.RETENTION'
          IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'request date'          :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (
          'STRUCTURAL_ANALYSIS_DESIGN.VERSIONED_ACTION_REQUEST' IN
          TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'release date'          :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (
          'STRUCTURAL_ANALYSIS_DESIGN.ACTION_DIRECTIVE' IN TYPEOF(x)) ))
           THEN
        RETURN(FALSE);
      END_IF;
    'start date'            :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (SIZEOF([
          'STRUCTURAL_ANALYSIS_DESIGN.ACTION_DIRECTIVE',
          'STRUCTURAL_ANALYSIS_DESIGN.RETENTION'] * TYPEOF(x)) = 1) ))
           THEN
        RETURN(FALSE);
      END_IF;
    'sign off date'         :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'APPROVAL_PERSON_ORGANIZATION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'contract date'         :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | ('STRUCTURAL_ANALYSIS_DESIGN.CONTRACT'
          IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'certification date'    :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (
          'STRUCTURAL_ANALYSIS_DESIGN.CERTIFICATION' IN TYPEOF(x)) ))
           THEN
        RETURN(FALSE);
      END_IF;
    'classification date'   :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'SECURITY_CLASSIFICATION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'declassification date'   :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'SECURITY_CLASSIFICATION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    OTHERWISE               :         RETURN(TRUE);
    END_CASE;
  RETURN(TRUE);
```

```
END_FUNCTION; -- applied_date_correlation

FUNCTION applied_date_time_correlation(
             e: applied_date_and_time_assignment
    ): BOOLEAN;

  LOCAL
    dt_role : STRING;
  END_LOCAL;
  dt_role := e\date_and_time_assignment.role.name;
  CASE dt_role OF
    'change date'          :          IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (
          'STRUCTURAL_ANALYSIS_DESIGN.ACTION_DIRECTIVE' IN TYPEOF(x)) ))
           THEN
        RETURN(FALSE);
      END_IF;
    'creation date'        :          IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'PRODUCT_DEFINITION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'earliest end date'    :          IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | ('STRUCTURAL_ANALYSIS_DESIGN.RETENTION'
          IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'latest end date'      :          IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | ('STRUCTURAL_ANALYSIS_DESIGN.RETENTION'
          IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'request date'         :          IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (
          'STRUCTURAL_ANALYSIS_DESIGN.VERSIONED_ACTION_REQUEST' IN
          TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'release date'         :          IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (
          'STRUCTURAL_ANALYSIS_DESIGN.ACTION_DIRECTIVE' IN TYPEOF(x)) ))
           THEN
        RETURN(FALSE);
      END_IF;
    'start date'           :          IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (SIZEOF([
          'STRUCTURAL_ANALYSIS_DESIGN.ACTION_DIRECTIVE',
          'STRUCTURAL_ANALYSIS_DESIGN.RETENTION'] * TYPEOF(x)) = 1) ))
           THEN
        RETURN(FALSE);
      END_IF;
    'sign off date'        :          IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'APPROVAL_PERSON_ORGANIZATION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'contract date'        :          IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | ('STRUCTURAL_ANALYSIS_DESIGN.CONTRACT'
          IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
```

```
    'certification date'   :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (
          'STRUCTURAL_ANALYSIS_DESIGN.CERTIFICATION' IN TYPEOF(x)) ))
           THEN
        RETURN(FALSE);
      END_IF;
    'classification date'  :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'SECURITY_CLASSIFICATION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'declassification date'  :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'SECURITY_CLASSIFICATION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    OTHERWISE              :         RETURN(TRUE);
    END_CASE;
  RETURN(TRUE);

END_FUNCTION; -- applied_date_time_correlation

FUNCTION applied_organization_correlation(
           e: applied_organization_assignment
    ): BOOLEAN;

  LOCAL
    po_role : STRING;
  END_LOCAL;
  po_role := e\organization_assignment.role.name;
  CASE po_role OF
    'request recipient'     :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'VERSIONED_ACTION_REQUEST') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'initiator'             :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.' +
           'VERSIONED_ACTION_REQUEST',
          'STRUCTURAL_ANALYSIS_DESIGN.ACTION_DIRECTIVE'] * TYPEOF(x)) = 1)
))
           THEN
        RETURN(FALSE);
      END_IF;
    'creator'               :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.' +
           'PRODUCT_DEFINITION_FORMATION','STRUCTURAL_ANALYSIS_DESIGN.' +
           'PRODUCT_DEFINITION'] * TYPEOF(x)) = 1) )) THEN
        RETURN(FALSE);
      END_IF;
    'part supplier'         :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'PRODUCT_DEFINITION_FORMATION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'analysis supplier'     :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'PRODUCT_DEFINITION_FORMATION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'design supplier'       :         IF SIZEOF(e.items) <> SIZEOF(
```



```
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'PRODUCT_DEFINITION_FORMATION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'analysis owner'         :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | ('STRUCTURAL_ANALYSIS_DESIGN.PRODUCT' IN
           TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'design owner'           :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | ('STRUCTURAL_ANALYSIS_DESIGN.PRODUCT' IN
           TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'configuration manager' :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'CONFIGURATION_ITEM') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'contractor'             :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | ('STRUCTURAL_ANALYSIS_DESIGN.CONTRACT'
          IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'classification officer'  :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'SECURITY_CLASSIFICATION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    OTHERWISE                :         RETURN(TRUE);
    END_CASE;
  RETURN(TRUE);

END_FUNCTION; -- applied_organization_correlation

FUNCTION applied_person_and_organization_correlation(
             e: applied_person_and_organization_assignment
    ): BOOLEAN;

  LOCAL
    po_role : STRING;
  END_LOCAL;
  po_role := e\person_and_organization_assignment.role.name;
  CASE po_role OF
    'request recipient'      :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'VERSIONED_ACTION_REQUEST') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'initiator'              :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.' +
           'VERSIONED_ACTION_REQUEST',
          'STRUCTURAL_ANALYSIS_DESIGN.ACTION_DIRECTIVE'] * TYPEOF(x)) = 1)
))
           THEN
        RETURN(FALSE);
      END_IF;
    'creator'                :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.' +
           'PRODUCT_DEFINITION_FORMATION','STRUCTURAL_ANALYSIS_DESIGN.' +
           'PRODUCT_DEFINITION'] * TYPEOF(x)) = 1) )) THEN
```

```
        RETURN(FALSE);
      END_IF;
    'part supplier'          :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'PRODUCT_DEFINITION_FORMATION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'analysis supplier'      :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'PRODUCT_DEFINITION_FORMATION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'design supplier'        :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'PRODUCT_DEFINITION_FORMATION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'analysis owner'         :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | ('STRUCTURAL_ANALYSIS_DESIGN.PRODUCT' IN
           TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'design owner'           :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | ('STRUCTURAL_ANALYSIS_DESIGN.PRODUCT' IN
           TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'configuration manager' :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'CONFIGURATION_ITEM') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'contractor'             :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | ('STRUCTURAL_ANALYSIS_DESIGN.CONTRACT'
          IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'classification officer'  :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('STRUCTURAL_ANALYSIS_DESIGN.' +
          'SECURITY_CLASSIFICATION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    OTHERWISE                :         RETURN(TRUE);
    END_CASE;
  RETURN(TRUE);

END_FUNCTION; -- applied_person_and_organization_correlation

FUNCTION appropriate_list_value_existence(
             values: LIST [1:?] OF GENERIC;
             type_self: SET [1:?] OF STRING
    ): BOOLEAN;

  LOCAL
    i : INTEGER;
  END_LOCAL;
```

```
  REPEAT i := 1 TO HIINDEX(values) BY 1;
    IF NOT appropriate_value_existence(values[i],type_self) THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  RETURN(TRUE);

END_FUNCTION; -- appropriate_list_value_existence

FUNCTION appropriate_set_value_existence(
             values_and_locations: SET [1:?] OF GENERIC;
             type_self: SET [1:?] OF STRING
    ): BOOLEAN;

  LOCAL
    i : INTEGER;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(values_and_locations) BY 1;
    IF NOT appropriate_value_existence(values_and_locations[i].
        simple_value,type_self) THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  RETURN(TRUE);

END_FUNCTION; -- appropriate_set_value_existence

FUNCTION appropriate_value_existence(
             a_value_e: GENERIC;
             type_self: SET [1:?] OF STRING
    ): BOOLEAN;

  LOCAL
    feacr        : STRING;
    value_typeof : SET [1:?] OF STRING;
  END_LOCAL;
  feacr := 'STRUCTURAL_ANALYSIS_DESIGN.';
  value_typeof := TYPEOF(a_value_e);
  IF ((feacr + 'OUTPUT_REQUEST_STATE') IN type_self) AND (NOT ((feacr +
      'UNSPECIFIED_VALUE') IN value_typeof)) THEN
    RETURN(FALSE);
  END_IF;
  RETURN(TRUE);

END_FUNCTION; -- appropriate_value_existence

FUNCTION assembly_shape_is_defined(
             assy: next_assembly_usage_occurrence
    ): BOOLEAN;

  LOCAL
    srr_set   : SET OF shape_representation_relationship := [];
    i         : INTEGER;
    j         : INTEGER;
    sdr_set   : SET OF shape_definition_representation := [];
    sdr_a_set : SET OF shape_definition_representation := [];
    pr1_set   : SET OF property_definition := [];
    pdrel_set : SET OF product_definition_relationship := [];
    pr2_set   : SET OF property_definition := [];
  END_LOCAL;
  pr1_set := bag_to_set(USEDIN(assy.related_product_definition,
```

```
    'STRUCTURAL_ANALYSIS_DESIGN.PROPERTY_DEFINITION.' + 'DEFINITION'));
REPEAT i := 1 TO HIINDEX(pr1_set) BY 1;
  sdr_set := sdr_set + QUERY ( pdr <* bag_to_set(USEDIN(pr1_set[i],
      'STRUCTURAL_ANALYSIS_DESIGN.' +
      'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) | ((
      'STRUCTURAL_ANALYSIS_DESIGN.' + 'SHAPE_DEFINITION_REPRESENTATION')
      IN TYPEOF(pdr)) );
END_REPEAT;
pdrel_set := bag_to_set(USEDIN(assy.related_product_definition,
    'STRUCTURAL_ANALYSIS_DESIGN.PRODUCT_DEFINITION_RELATIONSHIP.' +
    'RELATED_PRODUCT_DEFINITION'));
REPEAT j := 1 TO HIINDEX(pdrel_set) BY 1;
  pr2_set := pr2_set + bag_to_set(USEDIN(pdrel_set[j],
      'STRUCTURAL_ANALYSIS_DESIGN.PROPERTY_DEFINITION.' + 'DEFINITION'));
END_REPEAT;
REPEAT i := 1 TO HIINDEX(pr2_set) BY 1;
  sdr_set := sdr_set + QUERY ( pdr <* bag_to_set(USEDIN(pr2_set[i],
      'STRUCTURAL_ANALYSIS_DESIGN.' +
      'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) | ((
      'STRUCTURAL_ANALYSIS_DESIGN.' + 'SHAPE_DEFINITION_REPRESENTATION')
      IN TYPEOF(pdr)) );
END_REPEAT;
pr1_set := bag_to_set(USEDIN(assy.relating_product_definition,
    'STRUCTURAL_ANALYSIS_DESIGN.PROPERTY_DEFINITION.' + 'DEFINITION'));
REPEAT i := 1 TO HIINDEX(pr1_set) BY 1;
  sdr_a_set := sdr_a_set + QUERY ( pdr <* bag_to_set(USEDIN(pr1_set[i],
      'STRUCTURAL_ANALYSIS_DESIGN.' +
      'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) | ((
      'STRUCTURAL_ANALYSIS_DESIGN.' + 'SHAPE_DEFINITION_REPRESENTATION')
      IN TYPEOF(pdr)) );
END_REPEAT;
IF (SIZEOF(sdr_set) > 0) AND (SIZEOF(sdr_a_set) > 0) THEN
  REPEAT i := 1 TO HIINDEX(sdr_set) BY 1;
    srr_set := QUERY ( rr <* bag_to_set(USEDIN(sdr_set[i]\
        property_definition_representation.used_representation,
        'STRUCTURAL_ANALYSIS_DESIGN.REPRESENTATION_RELATIONSHIP.REP_2'))
         | (('STRUCTURAL_ANALYSIS_DESIGN.' +
        'SHAPE_REPRESENTATION_RELATIONSHIP') IN TYPEOF(rr)) );
    IF SIZEOF(srr_set) > 0 THEN
      REPEAT j := 1 TO HIINDEX(srr_set) BY 1;
        IF SIZEOF(QUERY ( pdr <* bag_to_set(USEDIN(srr_set[j]\
            representation_relationship.rep_1,
            'STRUCTURAL_ANALYSIS_DESIGN.' +
            'PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTATION'))
             | (('STRUCTURAL_ANALYSIS_DESIGN.' +
            'SHAPE_DEFINITION_REPRESENTATION') IN TYPEOF(pdr)) ) *
            sdr_a_set) >= 1 THEN
          IF SIZEOF(QUERY ( cdsr <* bag_to_set(USEDIN(srr_set[j],
              'STRUCTURAL_ANALYSIS_DESIGN.' +
              'CONTEXT_DEPENDENT_SHAPE_REPRESENTATION.' +
              'REPRESENTATION_RELATION')) | (NOT (cdsr.
              represented_product_relation.definition :=: assy)) )) > 0
               THEN
            RETURN(FALSE);
          END_IF;
        END_IF;
      END_REPEAT;
    END_IF;
  END_REPEAT;
END_IF;
RETURN(TRUE);
```



```
END_FUNCTION; -- assembly_shape_is_defined

FUNCTION associated_surface(
             arg: pcurve_or_surface
    ): surface;

  LOCAL
    surf : surface;
  END_LOCAL;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.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
    i       : INTEGER;
    the_set : SET OF GENERIC:intype := [];
  END_LOCAL;
  IF SIZEOF(the_bag) > 0 THEN
    REPEAT i := 1 TO HIINDEX(the_bag) BY 1;
      the_set := the_set + the_bag[i];
    END_REPEAT;
  END_IF;
  RETURN(the_set);

END_FUNCTION; -- bag_to_set

FUNCTION base_axis(
             dim: INTEGER;
             axis1, axis2, axis3: direction
    ): LIST [2:3] OF direction;

  LOCAL
    u      : LIST [2:3] OF direction;
    d1     : direction;
    d2     : direction;
    factor : REAL;
  END_LOCAL;
  IF dim = 3 THEN
    d1 := NVL(normalise(axis3),dummy_gri || direction([0,0,1]));
    d2 := first_proj_axis(d1,axis1);
    u := [d2,second_proj_axis(d1,d2,axis2),d1];
  ELSE
    IF EXISTS(axis1) THEN
      d1 := normalise(axis1);
      u := [d1,orthogonal_complement(d1)];
      IF EXISTS(axis2) THEN
        factor := dot_product(axis2,u[2]);
        IF factor < 0 THEN
          u[2].direction_ratios[1] := -u[2].direction_ratios[1];
          u[2].direction_ratios[2] := -u[2].direction_ratios[2];
        END_IF;
      END_IF;
```

```
    ELSE
      IF EXISTS(axis2) THEN
        d1 := normalise(axis2);
        u := [orthogonal_complement(d1),d1];
        u[1].direction_ratios[1] := -u[1].direction_ratios[1];
        u[1].direction_ratios[2] := -u[1].direction_ratios[2];
      ELSE
        u := [dummy_gri || direction([1,0]),dummy_gri || direction([0,1])];
      END_IF;
    END_IF;
  END_IF;
  RETURN(u);

END_FUNCTION; -- base_axis

FUNCTION boolean_choose(
             b: BOOLEAN;
             choice1, choice2: GENERIC:item
    ): GENERIC:item;
  IF b THEN
    RETURN(choice1);
  ELSE
    RETURN(choice2);
  END_IF;

END_FUNCTION; -- boolean_choose

FUNCTION build_2axes(
             ref_direction: direction
    ): LIST [2:2] OF direction;

  LOCAL
    d : direction := NVL(normalise(ref_direction),dummy_gri ||
         direction([1,0]));
  END_LOCAL;
  RETURN([d,orthogonal_complement(d)]);

END_FUNCTION; -- build_2axes

FUNCTION build_axes(
             axis, ref_direction: direction
    ): LIST [3:3] OF direction;

  LOCAL
    d1 : direction;
    d2 : direction;
  END_LOCAL;
  d1 := NVL(normalise(axis),dummy_gri || direction([0,0,1]));
  d2 := first_proj_axis(d1,ref_direction);
  RETURN([d2,normalise(cross_product(d1,d2)).orientation,d1]);

END_FUNCTION; -- build_axes

FUNCTION build_direction_node(
             node_1, node_2: node_representation
    ): LIST [2:3] OF REAL;

  LOCAL
    ndim                   : INTEGER;
    i                      : INTEGER;
    u                      : direction;
```



```
    node_2_point           : cartesian_point;
    node_1_point           : cartesian_point;
    rep_items              : SET [1:?] OF representation_item;
    nodal_direction_ratios : LIST [2:3] OF REAL;
  END_LOCAL;
  rep_items := node_1.items;
  REPEAT i := 1 TO SIZEOF(rep_items) BY 1;
    IF 'STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT' IN TYPEOF(rep_items[
        i]) THEN
      node_1_point := rep_items[i];
      ESCAPE;
    END_IF;
  END_REPEAT;
  rep_items := node_2.items;
  REPEAT i := 1 TO SIZEOF(rep_items) BY 1;
    IF 'STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT' IN TYPEOF(rep_items[
        i]) THEN
      node_2_point := rep_items[i];
      ESCAPE;
    END_IF;
  END_REPEAT;
  ndim := HIINDEX(node_2_point.coordinates);
  REPEAT i := 1 TO ndim BY 1;
    u.direction_ratios[i] := node_2_point.coordinates[i] - node_1_point.
        coordinates[i];
  END_REPEAT;
  u := normalise(u);
  REPEAT i := 1 TO ndim BY 1;
    nodal_direction_ratios[i] := u.direction_ratios[i];
  END_REPEAT;
  RETURN(nodal_direction_ratios);

END_FUNCTION; -- build_direction_node

FUNCTION closed_shell_reversed(
             a_shell: closed_shell
    ): oriented_closed_shell;

  LOCAL
    the_reverse : oriented_closed_shell;
  END_LOCAL;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.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 consistent_element_or_group_reference(
            aspect, element: GENERIC
    ): BOOLEAN;

  LOCAL
    i    : INTEGER;
    srrs : STRING;
  END_LOCAL;
  srrs := 'STRUCTURAL_ANALYSIS_DESIGN.';
  IF (srrs + 'ELEMENT_REPRESENTATION') IN TYPEOF(element) THEN
    RETURN(consistent_element_reference(aspect,element));
  END_IF;
  IF (srrs + 'ELEMENT_GROUP') IN TYPEOF(element) THEN
    REPEAT i := 1 TO HIINDEX(element.elements) BY 1;
      IF NOT consistent_element_reference(aspect,element.elements[i])
           THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
    RETURN(TRUE);
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- consistent_element_or_group_reference

FUNCTION consistent_element_reference(
            aspect: GENERIC;
            element: element_representation
    ): BOOLEAN;

  LOCAL
    element_type : SET [1:?] OF STRING;
    feacr        : STRING;
    srrs         : STRING;
    aspect_type  : SET [1:?] OF STRING;
  END_LOCAL;
  srrs := 'STRUCTURAL_ANALYSIS_DESIGN.';
  feacr := 'STRUCTURAL_ANALYSIS_DESIGN.';
  aspect_type := TYPEOF(aspect);
  element_type := TYPEOF(element);
  IF (srrs + 'ELEMENT_REPRESENTATION') IN element_type THEN
    IF (srrs + 'ELEMENT_VOLUME') IN aspect_type THEN
      RETURN(TRUE);
    END_IF;
  END_IF;
  IF (srrs + 'VOLUME_3D_ELEMENT_REPRESENTATION') IN element_type THEN
    IF ((feacr + 'VOLUME_3D_FACE') IN aspect_type) OR ((feacr +
        'VOLUME_3D_EDGE') IN aspect_type) THEN
      RETURN(TRUE);
    END_IF;
  END_IF;
  IF ((srrs + 'AXISYMMETRIC_VOLUME_2D_ELEMENT_REPRESENTATION') IN
      element_type) OR ((srrs + 'PLANE_VOLUME_2D_ELEMENT_REPRESENTATION')
```

```
      IN element_type) THEN
    IF ((feacr + 'VOLUME_2D_FACE') IN aspect_type) OR ((feacr +
        'VOLUME_2D_EDGE') IN aspect_type) THEN
      RETURN(TRUE);
    END_IF;
  END_IF;
  IF (srrs + 'SURFACE_3D_ELEMENT_REPRESENTATION') IN element_type THEN
    IF ((feacr + 'SURFACE_3D_FACE') IN aspect_type) OR ((feacr +
        'SURFACE_3D_EDGE') IN aspect_type) THEN
      RETURN(TRUE);
    END_IF;
  END_IF;
  IF ((srrs + 'AXISYMMETRIC_SURFACE_2D_ELEMENT_REPRESENTATION') IN
      element_type) OR ((srrs + 'PLANE_SURFACE_2D_ELEMENT_REPRESENTATION')
      IN element_type) THEN
    IF ((feacr + 'SURFACE_2D_FACE') IN aspect_type) OR ((feacr +
        'SURFACE_2D_EDGE') IN aspect_type) THEN
      RETURN(TRUE);
    END_IF;
  END_IF;
  IF ((srrs + 'CURVE_3D_ELEMENT_REPRESENTATION') IN element_type) OR ((
      srrs + 'AXISYMMETRIC_CURVE_2D_ELEMENT_REPRESENTATION') IN
      element_type) OR ((srrs + 'PLANE_CURVE_2D_ELEMENT_REPRESENTATION')
      IN element_type) THEN
    IF (srrs + 'CURVE_EDGE') IN aspect_type THEN
      RETURN(TRUE);
    END_IF;
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- consistent_element_reference

FUNCTION consistent_geometric_reference(
             aspect: GENERIC;
             item: geometric_representation_item
    ): BOOLEAN;

  LOCAL
    item_type   : SET [1:?] OF STRING;
    feacr       : STRING;
    srrs        : STRING;
    aspect_type : SET [1:?] OF STRING;
  END_LOCAL;
  srrs := 'STRUCTURAL_ANALYSIS_DESIGN.';
  feacr := 'STRUCTURAL_ANALYSIS_DESIGN.';
  aspect_type := TYPEOF(aspect);
  item_type := TYPEOF(item);
  IF 'STRUCTURAL_ANALYSIS_DESIGN.SOLID_MODEL' IN item_type THEN
    IF (srrs + 'ELEMENT_VOLUME') IN aspect_type THEN
      RETURN(TRUE);
    END_IF;
  END_IF;
  IF ('STRUCTURAL_ANALYSIS_DESIGN.SURFACE' IN item_type) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.FACE_SURFACE' IN item_type) THEN
    IF SIZEOF([feacr + 'VOLUME_3D_FACE',feacr + 'VOLUME_2D_FACE',feacr +
        'SURFACE_3D_FACE',feacr + 'SURFACE_2D_FACE'] * aspect_type) = 1
        THEN
      RETURN(TRUE);
    END_IF;
  END_IF;
  IF ('STRUCTURAL_ANALYSIS_DESIGN.CURVE' IN item_type) OR (
```

```
      'STRUCTURAL_ANALYSIS_DESIGN.EDGE_CURVE' IN item_type) THEN
    IF SIZEOF(['feacr + 'VOLUME_3D_EDGE',feacr + 'VOLUME_2D_EDGE',feacr +
         'SURFACE_3D_EDGE',feacr + 'SURFACE_2D_EDGE',srrs + 'CURVE_EDGE']
        * aspect_type) = 1 THEN
      RETURN(TRUE);
    END_IF;
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- consistent_geometric_reference

FUNCTION consistent_list_values(
             values: LIST [1:?] OF field_value;
             variable: GENERIC
    ): BOOLEAN;

  LOCAL
    fv_type : SET [1:?] OF STRING;
    i       : INTEGER;
    vv_type : STRING;
  END_LOCAL;
  vv_type := variable_value_type(variable);
  REPEAT i := 1 TO HIINDEX(values) BY 1;
    fv_type := TYPEOF(values[i]);
    IF NOT (('STRUCTURAL_ANALYSIS_DESIGN.' + 'UNSPECIFIED_VALUE') IN
        fv_type) THEN
      IF NOT (vv_type IN fv_type) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
  END_REPEAT;
  RETURN(TRUE);

END_FUNCTION; -- consistent_list_values

FUNCTION consistent_set_values(
             values_and_locations: SET [1:?] OF GENERIC;
             variable: GENERIC
    ): BOOLEAN;

  LOCAL
    fv_type : SET [1:?] OF STRING;
    i       : INTEGER;
    vv_type : STRING;
  END_LOCAL;
  vv_type := variable_value_type(variable);
  REPEAT i := 1 TO HIINDEX(values_and_locations) BY 1;
    fv_type := TYPEOF(values_and_locations[i].simple_value);
    IF NOT (('STRUCTURAL_ANALYSIS_DESIGN.' + 'UNSPECIFIED_VALUE') IN
        fv_type) THEN
      IF NOT (vv_type IN fv_type) THEN
        RETURN(FALSE);


      END_IF;
    END_IF;
  END_REPEAT;
  RETURN(TRUE);

END_FUNCTION; -- consistent_set_values
```



```
FUNCTION consistent_value(
             c_value: field_value;
             variable: GENERIC
    ): BOOLEAN;

  LOCAL
    fv_type : SET [1:?] OF STRING;
    vv_type : STRING;
  END_LOCAL;
  vv_type := variable_value_type(variable);
  fv_type := TYPEOF(c_value);
  IF NOT (('STRUCTURAL_ANALYSIS_DESIGN.' + 'UNSPECIFIED_VALUE') IN
      fv_type) THEN
    IF NOT (vv_type IN fv_type) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  RETURN(TRUE);

END_FUNCTION; -- consistent_value

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 ('STRUCTURAL_ANALYSIS_DESIGN.PCURVE' IN TYPEOF(c\
        composite_curve.segments[k].parent_curve))) AND (NOT (
        'STRUCTURAL_ANALYSIS_DESIGN.SURFACE_CURVE' IN TYPEOF(c\
        composite_curve.segments[k].parent_curve))) AND (NOT (
        'STRUCTURAL_ANALYSIS_DESIGN.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_geometry_shell_based_surface_model(
             m: shell_based_surface_model
    ): BOOLEAN;

  LOCAL
    result : BOOLEAN := TRUE;
  END_LOCAL;
  REPEAT j := 1 TO SIZEOF(m.sbsm_boundary) BY 1;
    IF (NOT ('STRUCTURAL_ANALYSIS_DESIGN.OPEN_SHELL' IN TYPEOF(m.
        sbsm_boundary[j]))) AND (NOT (
        'STRUCTURAL_ANALYSIS_DESIGN.CLOSED_SHELL' IN TYPEOF(m.
        sbsm_boundary[j]))) THEN
      result := FALSE;
      RETURN(result);
    END_IF;
  END_REPEAT;
  RETURN(result);

END_FUNCTION; -- constraints_geometry_shell_based_surface_model
```

```
FUNCTION constraints_geometry_shell_based_wireframe_model(
             m: shell_based_wireframe_model
    ): BOOLEAN;

  LOCAL
    result : BOOLEAN := TRUE;
  END_LOCAL;
  REPEAT j := 1 TO SIZEOF(m.sbwm_boundary) BY 1;
    IF (NOT ('STRUCTURAL_ANALYSIS_DESIGN.WIRE_SHELL' IN TYPEOF(m.
        sbwm_boundary[j]))) AND (NOT (
        'STRUCTURAL_ANALYSIS_DESIGN.VERTEX_SHELL' IN TYPEOF(m.
        sbwm_boundary[j]))) THEN
      result := FALSE;
      RETURN(result);
    END_IF;
  END_REPEAT;
  RETURN(result);

END_FUNCTION; -- constraints_geometry_shell_based_wireframe_model

FUNCTION constraints_param_b_spline(
             degree, up_knots, up_cp: INTEGER;
             knot_mult: LIST OF INTEGER;
             knots: LIST OF parameter_value
    ): BOOLEAN;

  LOCAL
    k      : INTEGER;
    sum    : INTEGER;
    result : BOOLEAN := TRUE;
  END_LOCAL;
  sum := knot_mult[1];
  REPEAT i := 2 TO up_knots BY 1;
    sum := sum + knot_mult[i];
  END_REPEAT;
  IF (degree < 1) OR (up_knots < 2) OR (up_cp < degree) OR (sum <> (
      degree + up_cp + 2)) THEN
    result := FALSE;
    RETURN(result);
  END_IF;
  k := knot_mult[1];
  IF (k < 1) OR (k > (degree + 1)) THEN
    result := FALSE;
    RETURN(result);
  END_IF;
  REPEAT i := 2 TO up_knots BY 1;
    IF (knot_mult[i] < 1) OR (knots[i] <= knots[i - 1]) THEN
      result := FALSE;
      RETURN(result);
    END_IF;
    k := knot_mult[i];
    IF (i < up_knots) AND (k > degree) THEN
      result := FALSE;
      RETURN(result);
    END_IF;
    IF (i = up_knots) AND (k > (degree + 1)) THEN
      result := FALSE;
      RETURN(result);
    END_IF;
  END_REPEAT;
  RETURN(result);
```



```
END_FUNCTION; -- constraints_param_b_spline

FUNCTION constraints_rectangular_composite_surface(
             s: rectangular_composite_surface
    ): BOOLEAN;
  REPEAT i := 1 TO s.n_u BY 1;
    REPEAT j := 1 TO s.n_v BY 1;
      IF NOT (('STRUCTURAL_ANALYSIS_DESIGN.B_SPLINE_SURFACE' IN TYPEOF(s
          .segments[i][j].parent_surface)) OR (
          'STRUCTURAL_ANALYSIS_DESIGN.RECTANGULAR_TRIMMED_SURFACE' IN
          TYPEOF(s.segments[i][j].parent_surface))) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_REPEAT;
  REPEAT i := 1 TO s.n_u - 1 BY 1;
    REPEAT j := 1 TO s.n_v BY 1;
      IF s.segments[i][j].u_transition = discontinuous THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_REPEAT;
  REPEAT i := 1 TO s.n_u BY 1;
    REPEAT j := 1 TO s.n_v - 1 BY 1;
      IF s.segments[i][j].v_transition = discontinuous THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_REPEAT;
  RETURN(TRUE);

END_FUNCTION; -- constraints_rectangular_composite_surface

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_angle(
             placement_1, placement_2: axis2_placement_3d
    ): plane_angle_measure;

  LOCAL
    v2     : direction;
    v1     : direction;
    mag_v2 : REAL;
    mag_v1 : REAL;
    theta  : plane_angle_measure;
  END_LOCAL;
  v1 := placement_1.p[1];
  v2 := placement_2.p[1];
  mag_v1 := SQRT((v1.direction_ratios[1] * v1.direction_ratios[1]) + (v1
      .direction_ratios[2] * v1.direction_ratios[2]));
  mag_v2 := SQRT((v2.direction_ratios[1] * v2.direction_ratios[1]) + (v2
      .direction_ratios[2] * v2.direction_ratios[2]));
  IF (mag_v1 = 0) OR (mag_v2 = 0) THEN
    theta := 0;
    RETURN(theta);
  END_IF;
  theta := ACOS(((v1.direction_ratios[1] * v2.direction_ratios[1]) + (v1
      .direction_ratios[2] * v2.direction_ratios[2])) / (mag_v1 * mag_v2));
  RETURN(theta);

END_FUNCTION; -- derive_angle

FUNCTION derive_dimensional_exponents(
             x: unit
    ): dimensional_exponents;

  LOCAL
    i      : INTEGER;
    result : dimensional_exponents := dimensional_exponents(0,0,0,0,0,0,
              0);
  END_LOCAL;
```



```
  IF 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT' IN TYPEOF(item) THEN
    dim := SIZEOF(item\cartesian_point.coordinates);
    RETURN(dim);
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.DIRECTION' IN TYPEOF(item) THEN
    dim := SIZEOF(item\direction.direction_ratios);
    RETURN(dim);
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.VECTOR' IN TYPEOF(item) THEN
    dim := SIZEOF(item\vector.orientation\direction.direction_ratios);
    RETURN(dim);
  END_IF;
  x := using_representations(item);
  y := x[1].context_of_items;
  dim := y\geometric_representation_context.coordinate_space_dimension;
  RETURN(dim);

END_FUNCTION; -- dimension_of

FUNCTION dimensions_for_si_unit(
             n: si_unit_name
    ): dimensional_exponents;
  CASE n OF
```

```
    metre          :         RETURN(dimensional_exponents(1,0,0,0,0,0,0));
    gram           :         RETURN(dimensional_exponents(0,1,0,0,0,0,0));
    second         :         RETURN(dimensional_exponents(0,0,1,0,0,0,0));
    ampere         :         RETURN(dimensional_exponents(0,0,0,1,0,0,0));
    kelvin         :         RETURN(dimensional_exponents(0,0,0,0,1,0,0));
    mole           :         RETURN(dimensional_exponents(0,0,0,0,0,1,0));
    candela        :         RETURN(dimensional_exponents(0,0,0,0,0,0,1));
    radian         :         RETURN(dimensional_exponents(0,0,0,0,0,0,0));
    steradian      :         RETURN(dimensional_exponents(0,0,0,0,0,0,0));
    hertz          :         RETURN(dimensional_exponents(0,0,-1,0,0,0,0));
    newton         :         RETURN(dimensional_exponents(1,1,-2,0,0,0,0));
    pascal         :         RETURN(dimensional_exponents(-1,1,-2,0,0,0,0));
    joule          :         RETURN(dimensional_exponents(2,1,-2,0,0,0,0));
    watt           :         RETURN(dimensional_exponents(2,1,-3,0,0,0,0));
    coulomb        :         RETURN(dimensional_exponents(0,0,1,1,0,0,0));
    volt           :         RETURN(dimensional_exponents(2,1,-3,-1,0,0,0));
    farad          :         RETURN(dimensional_exponents(-2,-1,4,1,0,0,0));
    ohm            :         RETURN(dimensional_exponents(2,1,-3,-2,0,0,0));
    siemens        :         RETURN(dimensional_exponents(-2,-1,3,2,0,0,0));
    weber          :         RETURN(dimensional_exponents(2,1,-2,-1,0,0,0));
    tesla          :         RETURN(dimensional_exponents(0,1,-2,-1,0,0,0));
    henry          :         RETURN(dimensional_exponents(2,1,-2,-2,0,0,0));
    degree_celsius  :          RETURN(dimensional_exponents(0,0,0,0,1,0,0));
    lumen          :         RETURN(dimensional_exponents(0,0,0,0,0,0,1));
    lux            :         RETURN(dimensional_exponents(-2,0,0,0,0,0,1));
    becquerel      :         RETURN(dimensional_exponents(0,0,-1,0,0,0,0));
    gray           :         RETURN(dimensional_exponents(2,0,-2,0,0,0,0));
    sievert        :         RETURN(dimensional_exponents(2,0,-2,0,0,0,0));
    END_CASE;

END_FUNCTION; -- dimensions_for_si_unit

FUNCTION dot_product(
             arg1, arg2: direction
    ): REAL;

  LOCAL
    ndim   : INTEGER;
    scalar : REAL;
    vec1   : direction;
    vec2   : direction;
  END_LOCAL;
  IF (NOT EXISTS(arg1)) OR (NOT EXISTS(arg2)) THEN
    scalar := ?;
  ELSE
    IF arg1.dim <> arg2.dim THEN
      scalar := ?;
    ELSE
      BEGIN
        vec1 := normalise(arg1);
        vec2 := normalise(arg2);
        ndim := arg1.dim;
        scalar := 0;
        REPEAT i := 1 TO ndim BY 1;
          scalar := scalar + (vec1.direction_ratios[i] * vec2.
              direction_ratios[i]);
        END_REPEAT;
      END;
    END_IF;
  END_IF;
  RETURN(scalar);
```

```
END_FUNCTION; -- dot_product

FUNCTION edge_reversed(
             an_edge: edge
    ): oriented_edge;

  LOCAL
    the_reverse : oriented_edge;
  END_LOCAL;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 gbsf_check_curve(
             cv: representation_item
    ): BOOLEAN;
  IF SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.BOUNDED_CURVE',
      'STRUCTURAL_ANALYSIS_DESIGN.CONIC',
      'STRUCTURAL_ANALYSIS_DESIGN.CURVE_REPLICA',
      'STRUCTURAL_ANALYSIS_DESIGN.LINE',
      'STRUCTURAL_ANALYSIS_DESIGN.OFFSET_CURVE_3D'] * TYPEOF(cv)) > 1
       THEN
    RETURN(FALSE);
  END_IF;
  IF SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.CIRCLE',
      'STRUCTURAL_ANALYSIS_DESIGN.ELLIPSE',
      'STRUCTURAL_ANALYSIS_DESIGN.TRIMMED_CURVE'] * TYPEOF(cv)) = 1 THEN
    RETURN(TRUE);
  ELSE
    IF (('STRUCTURAL_ANALYSIS_DESIGN.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 (('STRUCTURAL_ANALYSIS_DESIGN.COMPOSITE_CURVE' IN TYPEOF(cv))
          AND (cv\composite_curve.self_intersect = FALSE)) OR (cv\
```



```
          composite_curve.self_intersect = UNKNOWN) THEN
        RETURN(SIZEOF(QUERY ( seg <* cv\composite_curve.segments | (NOT
            gbsf_check_curve(seg.parent_curve)) )) = 0);
      ELSE
        IF 'STRUCTURAL_ANALYSIS_DESIGN.CURVE_REPLICA' IN TYPEOF(cv)
             THEN
          RETURN(gbsf_check_curve(cv\curve_replica.parent_curve));
        ELSE
          IF ('STRUCTURAL_ANALYSIS_DESIGN.OFFSET_CURVE_3D' IN TYPEOF(cv))
              AND ((cv\offset_curve_3d.self_intersect = FALSE) OR (cv\
              offset_curve_3d.self_intersect = UNKNOWN)) AND (NOT (
              'STRUCTURAL_ANALYSIS_DESIGN.POLYLINE' IN TYPEOF(cv\
              offset_curve_3d.basis_curve))) THEN
            RETURN(gbsf_check_curve(cv\offset_curve_3d.basis_curve));
          ELSE
            IF 'STRUCTURAL_ANALYSIS_DESIGN.PCURVE' IN TYPEOF(cv) THEN
              RETURN(gbsf_check_curve(cv\pcurve.reference_to_curve\
                  representation.items[1]) AND gbsf_check_surface(cv\
                  pcurve.basis_surface));
            ELSE
              IF 'STRUCTURAL_ANALYSIS_DESIGN.POLYLINE' IN TYPEOF(cv)
                   THEN
                IF SIZEOF(cv\polyline.points) >= 3 THEN
                  RETURN(TRUE);
                END_IF;
              ELSE
                IF 'STRUCTURAL_ANALYSIS_DESIGN.SURFACE_CURVE' IN TYPEOF(
                    cv) THEN
                  IF gbsf_check_curve(cv\surface_curve.curve_3d) THEN
                    REPEAT i := 1 TO SIZEOF(cv\surface_curve.
                        associated_geometry) BY 1;
                      IF 'STRUCTURAL_ANALYSIS_DESIGN.SURFACE' IN TYPEOF(
                          cv\surface_curve.associated_geometry[i]) THEN
                        IF NOT gbsf_check_surface(cv\surface_curve.
                            associated_geometry[i]) THEN
                          RETURN(FALSE);
                        END_IF;
                      ELSE
                        IF 'STRUCTURAL_ANALYSIS_DESIGN.PCURVE' IN
                            TYPEOF(cv\surface_curve.associated_geometry[i])
                             THEN
                          IF NOT gbsf_check_curve(cv\surface_curve.
                              associated_geometry[i]) THEN
                            RETURN(FALSE);
                          END_IF;
                        END_IF;
                      END_IF;
                    END_REPEAT;
                    RETURN(TRUE);
                  END_IF;
                END_IF;
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- gbsf_check_curve
```

```
FUNCTION gbsf_check_point(
             pnt: point
    ): BOOLEAN;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT' IN TYPEOF(pnt) THEN
    RETURN(TRUE);
  ELSE
    IF 'STRUCTURAL_ANALYSIS_DESIGN.POINT_ON_CURVE' IN TYPEOF(pnt) THEN
      RETURN(gbsf_check_curve(pnt\point_on_curve.basis_curve));
    ELSE
      IF 'STRUCTURAL_ANALYSIS_DESIGN.POINT_ON_SURFACE' IN TYPEOF(pnt)
           THEN
        RETURN(gbsf_check_surface(pnt\point_on_surface.basis_surface));
      ELSE
        IF 'STRUCTURAL_ANALYSIS_DESIGN.DEGENERATE_PCURVE' IN TYPEOF(pnt)
             THEN
          RETURN(gbsf_check_curve(pnt\degenerate_pcurve.
              reference_to_curve\representation.items[1]) AND
              gbsf_check_surface(pnt\degenerate_pcurve.basis_surface));
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- gbsf_check_point

FUNCTION gbsf_check_surface(
             sf: surface
    ): BOOLEAN;
  IF (('STRUCTURAL_ANALYSIS_DESIGN.B_SPLINE_SURFACE' IN TYPEOF(sf)) AND
      (sf\b_spline_surface.self_intersect = FALSE)) OR (sf\
      b_spline_surface.self_intersect = UNKNOWN) THEN
    RETURN(TRUE);
  ELSE
    IF SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.SPHERICAL_SURFACE',
        'STRUCTURAL_ANALYSIS_DESIGN.TOROIDAL_SURFACE',
        'STRUCTURAL_ANALYSIS_DESIGN.CURVE_BOUNDED_SURFACE',
        'STRUCTURAL_ANALYSIS_DESIGN.RECTANGULAR_TRIMMED_SURFACE'] *
        TYPEOF(sf)) = 1 THEN
      RETURN(TRUE);
    ELSE
      IF (('STRUCTURAL_ANALYSIS_DESIGN.OFFSET_SURFACE' IN TYPEOF(sf))
          AND (sf\offset_surface.self_intersect = FALSE)) OR (sf\
          offset_surface.self_intersect = UNKNOWN) THEN
        RETURN(gbsf_check_surface(sf\offset_surface.basis_surface));
      ELSE
        IF 'STRUCTURAL_ANALYSIS_DESIGN.RECTANGULAR_COMPOSITE_SURFACE' IN
             TYPEOF(sf) THEN
          REPEAT i := 1 TO SIZEOF(sf\rectangular_composite_surface.
              segments) BY 1;
            REPEAT j := 1 TO SIZEOF(sf\rectangular_composite_surface.
                segments[i]) BY 1;
              IF NOT gbsf_check_surface(sf\rectangular_composite_surface
                  .segments[i][j].parent_surface) THEN
                RETURN(FALSE);
              END_IF;
            END_REPEAT;
          END_REPEAT;
          RETURN(TRUE);
        ELSE
          IF 'STRUCTURAL_ANALYSIS_DESIGN.SURFACE_REPLICA' IN TYPEOF(sf)
```



```
            THEN
          RETURN(gbsf_check_surface(sf\surface_replica.parent_surface));
        ELSE
          IF 'STRUCTURAL_ANALYSIS_DESIGN.SURFACE_OF_REVOLUTION' IN
              TYPEOF(sf) THEN
            RETURN(gbsf_check_curve(sf\swept_surface.swept_curve));
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
END_IF;
RETURN(FALSE);

END_FUNCTION; -- gbsf_check_surface

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 'STRUCTURAL_ANALYSIS_DESIGN.PCURVE' IN TYPEOF(c) THEN
    surfs := [c\pcurve.basis_surface];
  ELSE
    IF 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 is_acyclic(
             arg: generic_expression
    ): BOOLEAN;
  RETURN(acyclic(arg,[]));

END_FUNCTION; -- is_acyclic
```

```
FUNCTION is_int_expr(
             arg: numeric_expression
    ): BOOLEAN;

  LOCAL
    i : INTEGER;
  END_LOCAL;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.INT_LITERAL' IN TYPEOF(arg) THEN
    RETURN(TRUE);
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.REAL_LITERAL' IN TYPEOF(arg) THEN
    RETURN(FALSE);
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.INT_NUMERIC_VARIABLE' IN TYPEOF(arg)
       THEN
    RETURN(TRUE);
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.REAL_NUMERIC_VARIABLE' IN TYPEOF(arg)
       THEN
    RETURN(FALSE);
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.ABS_FUNCTION' IN TYPEOF(arg) THEN
    RETURN(is_int_expr(arg\unary_numeric_expression.operand));
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.MINUS_FUNCTION' IN TYPEOF(arg) THEN
    RETURN(is_int_expr(arg\unary_numeric_expression.operand));
  END_IF;
  IF ('STRUCTURAL_ANALYSIS_DESIGN.SIN_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.COS_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.TAN_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.ASIN_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.ACOS_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.ATAN_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.EXP_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.LOG_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.LOG2_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.LOG10_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.SQUARE_ROOT_FUNCTION' IN TYPEOF(arg))
       THEN
    RETURN(FALSE);
  END_IF;
  IF ('STRUCTURAL_ANALYSIS_DESIGN.PLUS_EXPRESSION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.MULT_EXPRESSION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.MAXIMUM_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.MINIMUM_FUNCTION' IN TYPEOF(arg)) THEN
    REPEAT i := 1 TO SIZEOF(arg\multiple_arity_numeric_expression.
        operands) BY 1;
      IF NOT is_int_expr(arg\multiple_arity_numeric_expression.operands[
          i]) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
    RETURN(TRUE);
  END_IF;
  IF ('STRUCTURAL_ANALYSIS_DESIGN.MINUS_EXPRESSION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.POWER_EXPRESSION' IN TYPEOF(arg)) THEN
    RETURN(is_int_expr(arg\binary_numeric_expression.operands[1]) AND
        is_int_expr(arg\binary_numeric_expression.operands[2]));
  END_IF;
  IF ('STRUCTURAL_ANALYSIS_DESIGN.DIV_EXPRESSION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.MOD_EXPRESSION' IN TYPEOF(arg)) THEN
```



```
      RETURN(TRUE);
    END_IF;
    IF 'STRUCTURAL_ANALYSIS_DESIGN.SLASH_EXPRESSION' IN TYPEOF(arg) THEN
      RETURN(FALSE);
    END_IF;
    IF 'STRUCTURAL_ANALYSIS_DESIGN.LENGTH_FUNCTION' IN TYPEOF(arg) THEN
      RETURN(TRUE);
    END_IF;
    IF 'STRUCTURAL_ANALYSIS_DESIGN.VALUE_FUNCTION' IN TYPEOF(arg) THEN
      IF 'STRUCTURAL_ANALYSIS_DESIGN.INT_VALUE_FUNCTION' IN TYPEOF(arg)
           THEN
        RETURN(TRUE);
      ELSE
        RETURN(FALSE);
      END_IF;
    END_IF;
    IF 'STRUCTURAL_ANALYSIS_DESIGN.INTEGER_DEFINED_FUNCTION' IN TYPEOF(arg)
         THEN
      RETURN(TRUE);
    END_IF;
    IF 'STRUCTURAL_ANALYSIS_DESIGN.REAL_DEFINED_FUNCTION' IN TYPEOF(arg)
         THEN
      RETURN(FALSE);
    END_IF;
    IF 'STRUCTURAL_ANALYSIS_DESIGN.BOOLEAN_DEFINED_FUNCTION' IN TYPEOF(arg)
         THEN
      RETURN(FALSE);
    END_IF;
    IF 'STRUCTURAL_ANALYSIS_DESIGN.STRING_DEFINED_FUNCTION' IN TYPEOF(arg)
         THEN
      RETURN(FALSE);
    END_IF;
    RETURN(FALSE);

  END_FUNCTION; -- is_int_expr

  FUNCTION is_sql_mappable(
               arg: expression
      ): BOOLEAN;

    LOCAL
      i : INTEGER;
    END_LOCAL;
    IF 'STRUCTURAL_ANALYSIS_DESIGN.SIMPLE_NUMERIC_EXPRESSION' IN TYPEOF(
        arg) THEN
      RETURN(TRUE);
    END_IF;
    IF 'STRUCTURAL_ANALYSIS_DESIGN.SQL_MAPPABLE_DEFINED_FUNCTION' IN
        TYPEOF(arg) THEN
      RETURN(TRUE);
    END_IF;
    IF 'STRUCTURAL_ANALYSIS_DESIGN.MINUS_FUNCTION' IN TYPEOF(arg) THEN
      RETURN(is_sql_mappable(arg\unary_numeric_expression.operand));
    END_IF;
    IF ('STRUCTURAL_ANALYSIS_DESIGN.ABS_FUNCTION' IN TYPEOF(arg)) OR (
        'STRUCTURAL_ANALYSIS_DESIGN.SIN_FUNCTION' IN TYPEOF(arg)) OR (
        'STRUCTURAL_ANALYSIS_DESIGN.COS_FUNCTION' IN TYPEOF(arg)) OR (
        'STRUCTURAL_ANALYSIS_DESIGN.TAN_FUNCTION' IN TYPEOF(arg)) OR (
        'STRUCTURAL_ANALYSIS_DESIGN.ASIN_FUNCTION' IN TYPEOF(arg)) OR (
        'STRUCTURAL_ANALYSIS_DESIGN.ACOS_FUNCTION' IN TYPEOF(arg)) OR (
        'STRUCTURAL_ANALYSIS_DESIGN.ATAN_FUNCTION' IN TYPEOF(arg)) OR (
```

```
      'STRUCTURAL_ANALYSIS_DESIGN.EXP_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.LOG_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.LOG2_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.LOG10_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.SQUARE_ROOT_FUNCTION' IN TYPEOF(arg))
      OR ('STRUCTURAL_ANALYSIS_DESIGN.VALUE_FUNCTION' IN TYPEOF(arg)) OR
      ('STRUCTURAL_ANALYSIS_DESIGN.LENGTH_FUNCTION' IN TYPEOF(arg)) THEN
    RETURN(FALSE);
  END_IF;
  IF ('STRUCTURAL_ANALYSIS_DESIGN.PLUS_EXPRESSION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.MULT_EXPRESSION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.MAXIMUM_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.MINIMUM_FUNCTION' IN TYPEOF(arg)) THEN
    REPEAT i := 1 TO SIZEOF(arg\multiple_arity_numeric_expression.
        operands) BY 1;
      IF NOT is_sql_mappable(arg\multiple_arity_numeric_expression.
          operands[i]) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
    RETURN(TRUE);
  END_IF;
  IF ('STRUCTURAL_ANALYSIS_DESIGN.MINUS_EXPRESSION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.SLASH_EXPRESSION' IN TYPEOF(arg)) THEN
    RETURN(is_sql_mappable(arg\binary_numeric_expression.operands[1])
        AND is_sql_mappable(arg\binary_numeric_expression.operands[2]));
  END_IF;
  IF ('STRUCTURAL_ANALYSIS_DESIGN.DIV_EXPRESSION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.MOD_EXPRESSION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.POWER_EXPRESSION' IN TYPEOF(arg)) THEN
    RETURN(FALSE);
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.SIMPLE_BOOLEAN_EXPRESSION' IN TYPEOF(
      arg) THEN
    RETURN(TRUE);
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.NOT_EXPRESSION' IN TYPEOF(arg) THEN
    RETURN(is_sql_mappable(arg\unary_generic_expression.operand));
  END_IF;
  IF ('STRUCTURAL_ANALYSIS_DESIGN.ODD_FUNCTION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.XOR_EXPRESSION' IN TYPEOF(arg)) THEN
    RETURN(FALSE);
  END_IF;
  IF ('STRUCTURAL_ANALYSIS_DESIGN.AND_EXPRESSION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.OR_EXPRESSION' IN TYPEOF(arg)) THEN
    REPEAT i := 1 TO SIZEOF(arg\multiple_arity_boolean_expression.
        operands) BY 1;
      IF NOT is_sql_mappable(arg\multiple_arity_boolean_expression.
          operands[i]) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
    RETURN(TRUE);
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.EQUALS_EXPRESSION' IN TYPEOF(arg) THEN
    RETURN(is_sql_mappable(arg\binary_generic_expression.operands[1])
        AND is_sql_mappable(arg\binary_generic_expression.operands[2]));
  END_IF;
  IF ('STRUCTURAL_ANALYSIS_DESIGN.COMPARISON_EQUAL' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.COMPARISON_GREATER' IN TYPEOF(arg)) OR
      ('STRUCTURAL_ANALYSIS_DESIGN.COMPARISON_GREATER_EQUAL' IN TYPEOF(
```



```
      arg)) OR ('STRUCTURAL_ANALYSIS_DESIGN.COMPARISON_LESS' IN TYPEOF(
      arg)) OR ('STRUCTURAL_ANALYSIS_DESIGN.COMPARISON_LESS_EQUAL' IN
      TYPEOF(arg)) OR ('STRUCTURAL_ANALYSIS_DESIGN.COMPARISON_NOT_EQUAL'
      IN TYPEOF(arg)) OR ('STRUCTURAL_ANALYSIS_DESIGN.LIKE_EXPRESSION' IN
       TYPEOF(arg)) THEN
    RETURN(is_sql_mappable(arg\comparison_expression.operands[1]) AND
        is_sql_mappable(arg\comparison_expression.operands[2]));
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.INTERVAL_EXPRESSION' IN TYPEOF(arg)
       THEN
    RETURN(is_sql_mappable(arg\interval_expression.interval_low) AND
        is_sql_mappable(arg\interval_expression.interval_high) AND
        is_sql_mappable(arg\interval_expression.interval_item));
  END_IF;
  IF ('STRUCTURAL_ANALYSIS_DESIGN.NUMERIC_DEFINED_FUNCTION' IN TYPEOF(
      arg)) OR ('STRUCTURAL_ANALYSIS_DESIGN.BOOLEAN_DEFINED_FUNCTION' IN
      TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.STRING_DEFINED_FUNCTION' IN TYPEOF(arg))
       THEN
    RETURN(FALSE);
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.SIMPLE_STRING_EXPRESSION' IN TYPEOF(arg)
       THEN
    RETURN(TRUE);
  END_IF;
  IF ('STRUCTURAL_ANALYSIS_DESIGN.INDEX_EXPRESSION' IN TYPEOF(arg)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.SUBSTRING_EXPRESSION' IN TYPEOF(arg))
      OR ('STRUCTURAL_ANALYSIS_DESIGN.CONCAT_EXPRESSION' IN TYPEOF(arg))
      OR ('STRUCTURAL_ANALYSIS_DESIGN.FORMAT_FUNCTION' IN TYPEOF(arg))
       THEN
    RETURN(FALSE);
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- is_sql_mappable

FUNCTION item_in_context(
             item: representation_item;
             cntxt: representation_context
    ): BOOLEAN;

  LOCAL
    i : INTEGER;
    y : BAG OF representation_item;
  END_LOCAL;
  IF SIZEOF(USEDIN(item,
      'STRUCTURAL_ANALYSIS_DESIGN.REPRESENTATION.ITEMS') * cntxt.
      representations_in_context) > 0 THEN
    RETURN(TRUE);
  ELSE
    y := QUERY ( z <* USEDIN(item,'') | (
        'STRUCTURAL_ANALYSIS_DESIGN.REPRESENTATION_ITEM' IN TYPEOF(z)) );
    IF SIZEOF(y) > 0 THEN
      REPEAT i := 1 TO HIINDEX(y) BY 1;
        IF item_in_context(y[i],cntxt) THEN
          RETURN(TRUE);
        END_IF;
      END_REPEAT;
    END_IF;
  END_IF;
  RETURN(FALSE);
```

```
END_FUNCTION; -- item_in_context

FUNCTION leap_year(
             year: year_number
    ): BOOLEAN;
  IF (((year MOD 4) = 0) AND ((year MOD 100) <> 0)) OR ((year MOD 400) =
      0) THEN
    RETURN(TRUE);
  ELSE
    RETURN(FALSE);
  END_IF;

END_FUNCTION; -- leap_year

FUNCTION list_face_loops(
             f: face
    ): LIST [0:?] OF loop;

  LOCAL
    loops : LIST [0:?] OF loop := [];
  END_LOCAL;
  REPEAT i := 1 TO SIZEOF(f.bounds) BY 1;
    loops := loops + f.bounds[i].bound;
  END_REPEAT;
  RETURN(loops);

END_FUNCTION; -- list_face_loops

FUNCTION list_of_topology_reversed(
             a_list: list_of_reversible_topology_item
    ): list_of_reversible_topology_item;

  LOCAL
    the_reverse : list_of_reversible_topology_item;
  END_LOCAL;
  the_reverse := [];
  REPEAT i := 1 TO SIZEOF(a_list) BY 1;
    the_reverse := topology_reversed(a_list[i]) + the_reverse;
  END_REPEAT;
  RETURN(the_reverse);

END_FUNCTION; -- list_of_topology_reversed

FUNCTION list_to_array(
             lis: LIST [0:?] OF GENERIC:t;
             low, u: INTEGER
    ): ARRAY [low:u] OF GENERIC:t;

  LOCAL
    n   : INTEGER;
    res : ARRAY [low:u] OF GENERIC:t;
  END_LOCAL;
  n := SIZEOF(lis);
  IF n <> ((u - low) + 1) THEN
    RETURN(?);
```



```
  ELSE
    res := [lis[1],n];
    REPEAT i := 2 TO n BY 1;
      res[(low + i) - 1] := lis[i];
    END_REPEAT;
    RETURN(res);
  END_IF;

END_FUNCTION; -- list_to_array

FUNCTION list_to_set(
             l: LIST [0:?] OF GENERIC:t
    ): SET OF GENERIC:t;

  LOCAL
    s : SET OF GENERIC:t := [];
  END_LOCAL;
  REPEAT i := 1 TO SIZEOF(l) BY 1;
    s := s + l[i];
  END_REPEAT;
  RETURN(s);

END_FUNCTION; -- list_to_set

FUNCTION make_array_of_array(
             lis: LIST [1:?] OF LIST [1:?] OF GENERIC:t;
             low1, u1, low2, u2: INTEGER
    ): ARRAY [low1:u1] OF ARRAY [low2:u2] OF GENERIC:t;

  LOCAL
    res : ARRAY [low1:u1] OF ARRAY [low2:u2] OF GENERIC:t;
  END_LOCAL;
  IF ((u1 - low1) + 1) <> SIZEOF(lis) THEN
    RETURN(?);
  END_IF;
  IF ((u2 - low2) + 1) <> SIZEOF(lis[1]) THEN
    RETURN(?);
  END_IF;
  res := [list_to_array(lis[1],low2,u2),(u1 - low1) + 1];
  REPEAT i := 2 TO HIINDEX(lis) BY 1;
    IF ((u2 - low2) + 1) <> SIZEOF(lis[i]) THEN
      RETURN(?);
    END_IF;
    res[(low1 + i) - 1] := list_to_array(lis[i],low2,u2);
  END_REPEAT;
  RETURN(res);

END_FUNCTION; -- make_array_of_array

FUNCTION mixed_loop_type_set(
             l: SET [0:?] OF loop
    ): LOGICAL;

  LOCAL
    poly_loop_type : LOGICAL;
  END_LOCAL;
  IF SIZEOF(l) <= 1 THEN
    RETURN(FALSE);
  END_IF;
  poly_loop_type := 'STRUCTURAL_ANALYSIS_DESIGN.POLY_LOOP' IN TYPEOF(l[1]);
  REPEAT i := 2 TO SIZEOF(l) BY 1;
```

```
    IF ('STRUCTURAL_ANALYSIS_DESIGN.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 msb_shells(
             brep: manifold_solid_brep
    ): SET [1:?] OF closed_shell;
  IF SIZEOF(QUERY ( msbtype <* TYPEOF(brep) | (msbtype LIKE
      '*BREP_WITH_VOIDS') )) >= 1 THEN
    RETURN(brep\brep_with_voids.voids + brep.outer);
  ELSE
    RETURN([brep.outer]);
  END_IF;

END_FUNCTION; -- msb_shells

FUNCTION msf_curve_check(
             cv: representation_item
    ): BOOLEAN;
  IF SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.BOUNDED_CURVE',
      'STRUCTURAL_ANALYSIS_DESIGN.CONIC',
      'STRUCTURAL_ANALYSIS_DESIGN.CURVE_REPLICA',
      'STRUCTURAL_ANALYSIS_DESIGN.LINE',
      'STRUCTURAL_ANALYSIS_DESIGN.OFFSET_CURVE_3D'] * TYPEOF(cv)) > 1
       THEN
    RETURN(FALSE);
  END_IF;
  IF (('STRUCTURAL_ANALYSIS_DESIGN.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(['STRUCTURAL_ANALYSIS_DESIGN.CONIC',
        'STRUCTURAL_ANALYSIS_DESIGN.LINE'] * TYPEOF(cv)) = 1 THEN
      RETURN(TRUE);
    ELSE
      IF 'STRUCTURAL_ANALYSIS_DESIGN.CURVE_REPLICA' IN TYPEOF(cv) THEN
        RETURN(msf_curve_check(cv\curve_replica.parent_curve));
      ELSE
        IF ('STRUCTURAL_ANALYSIS_DESIGN.OFFSET_CURVE_3D' IN TYPEOF(cv))
            AND ((cv\offset_curve_3d.self_intersect = FALSE) OR (cv\
            offset_curve_3d.self_intersect = UNKNOWN)) AND (NOT (
            'STRUCTURAL_ANALYSIS_DESIGN.POLYLINE' IN TYPEOF(cv\
            offset_curve_3d.basis_curve))) THEN
          RETURN(msf_curve_check(cv\offset_curve_3d.basis_curve));
        ELSE
          IF 'STRUCTURAL_ANALYSIS_DESIGN.PCURVE' IN TYPEOF(cv) THEN
            RETURN(msf_curve_check(cv\pcurve.reference_to_curve\
                representation.items[1]) AND msf_surface_check(cv\pcurve.
                basis_surface));
          ELSE
            IF 'STRUCTURAL_ANALYSIS_DESIGN.SURFACE_CURVE' IN TYPEOF(cv)
                 THEN
              IF msf_curve_check(cv\surface_curve.curve_3d) THEN
                REPEAT i := 1 TO SIZEOF(cv\surface_curve.
                    associated_geometry) BY 1;
```



```
                    IF 'STRUCTURAL_ANALYSIS_DESIGN.SURFACE' IN TYPEOF(cv\
                        surface_curve.associated_geometry[i]) THEN
                      IF NOT msf_surface_check(cv\surface_curve.
                          associated_geometry[i]) THEN
                        RETURN(FALSE);
                      END_IF;
                    ELSE
                      IF 'STRUCTURAL_ANALYSIS_DESIGN.PCURVE' IN TYPEOF(cv\
                          surface_curve.associated_geometry[i]) THEN
                        IF NOT msf_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 'STRUCTURAL_ANALYSIS_DESIGN.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; -- msf_curve_check

  FUNCTION msf_surface_check(
               surf: surface
      ): BOOLEAN;
    IF 'STRUCTURAL_ANALYSIS_DESIGN.ELEMENTARY_SURFACE' IN TYPEOF(surf)
         THEN
      RETURN(TRUE);
    ELSE
      IF 'STRUCTURAL_ANALYSIS_DESIGN.SWEPT_SURFACE' IN TYPEOF(surf) THEN
        RETURN(msf_curve_check(surf\swept_surface.swept_curve));
      ELSE
        IF (('STRUCTURAL_ANALYSIS_DESIGN.OFFSET_SURFACE' IN TYPEOF(surf))
            AND (surf\offset_surface.self_intersect = FALSE)) OR (surf\
            offset_surface.self_intersect = UNKNOWN) THEN
          RETURN(msf_surface_check(surf\offset_surface.basis_surface));
        ELSE
          IF 'STRUCTURAL_ANALYSIS_DESIGN.SURFACE_REPLICA' IN TYPEOF(surf)
               THEN
            RETURN(msf_surface_check(surf\surface_replica.parent_surface));
          ELSE
            IF (('STRUCTURAL_ANALYSIS_DESIGN.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; -- msf_surface_check

FUNCTION necessary_value_coordinate_system(
             cs_value: field_value;
             coordinate_system: GENERIC
    ): BOOLEAN;
  IF SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.TENSOR1_2D',
      'STRUCTURAL_ANALYSIS_DESIGN.TENSOR1_3D',
      'STRUCTURAL_ANALYSIS_DESIGN.SYMMETRIC_TENSOR2_2D',
      'STRUCTURAL_ANALYSIS_DESIGN.SYMMETRIC_TENSOR2_3D'] * TYPEOF(
      cs_value)) = 1 THEN
    IF NOT EXISTS(coordinate_system) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  RETURN(TRUE);

END_FUNCTION; -- necessary_value_coordinate_system

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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 number_of_terms(
             node_dof_list: LIST [1:?] OF LIST [1:?] OF
                  degree_of_freedom;
             matrix_type: matrix_symmetry
    ): INTEGER;

  LOCAL
    num_terms          : INTEGER;
    number_of_freedoms : INTEGER;
  END_LOCAL;
  number_of_freedoms := 0;
  REPEAT i := 1 TO SIZEOF(node_dof_list) BY 1;
    number_of_freedoms := number_of_freedoms + SIZEOF(node_dof_list[i]);
  END_REPEAT;
  IF matrix_type = symmetric THEN
    num_terms := (number_of_freedoms * (number_of_freedoms + 1)) DIV 2;
  END_IF;
  IF matrix_type = diagonal THEN
    num_terms := number_of_freedoms;
  END_IF;
  RETURN(num_terms);

END_FUNCTION; -- number_of_terms

FUNCTION open_shell_reversed(
             a_shell: open_shell
    ): oriented_open_shell;

  LOCAL
    the_reverse : oriented_open_shell;
  END_LOCAL;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.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
    ): LOGICAL;

  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 'STRUCTURAL_ANALYSIS_DESIGN.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 required_0d_nodes(
             node_list: LIST [1:?] OF node_representation
    ): BOOLEAN;
  RETURN(SIZEOF(node_list) = 1);

END_FUNCTION; -- required_0d_nodes

FUNCTION required_1d_nodes(
             node_list: LIST [1:?] OF node_representation;
             order: element_order
    ): BOOLEAN;
```



```
  LOCAL
    end_nodes        : INTEGER;
    additional_nodes : INTEGER;
  END_LOCAL;
  end_nodes := 2;
  IF order = linear THEN
    additional_nodes := 0;
  END_IF;
  IF order = quadratic THEN
    additional_nodes := 1;
  END_IF;
  IF order = cubic THEN
    additional_nodes := 2;
  END_IF;
  RETURN(SIZEOF(node_list) = (end_nodes + additional_nodes));

END_FUNCTION; -- required_1d_nodes

FUNCTION required_2d_nodes(
             node_list: LIST [1:?] OF node_representation;
             element_shape: element_2d_shape;
             order: element_order
    ): BOOLEAN;

  LOCAL
    vertex_nodes         : INTEGER;
    edge_face_body_nodes : INTEGER;
    edge_nodes           : INTEGER;
  END_LOCAL;
  IF element_shape = triangle THEN
    vertex_nodes := 3;
    IF order = linear THEN
      edge_nodes := 0;
      edge_face_body_nodes := 0;
    END_IF;
    IF order = quadratic THEN
      edge_nodes := 3;
      edge_face_body_nodes := 3;
    END_IF;
    IF order = cubic THEN
      edge_nodes := 6;
      edge_face_body_nodes := 7;
    END_IF;
  END_IF;
  IF element_shape = quadrilateral THEN
    vertex_nodes := 4;
    IF order = linear THEN
      edge_nodes := 0;
      edge_face_body_nodes := 0;
    END_IF;
    IF order = quadratic THEN
      edge_nodes := 4;
      edge_face_body_nodes := 5;
    END_IF;
    IF order = cubic THEN
      edge_nodes := 8;
      edge_face_body_nodes := 12;
    END_IF;
  END_IF;
  RETURN((SIZEOF(node_list) = (vertex_nodes + edge_nodes)) OR (SIZEOF(
      node_list) = (vertex_nodes + edge_face_body_nodes)));
```

```
END_FUNCTION; -- required_2d_nodes

FUNCTION required_3d_nodes(
             node_list: LIST [1:?] OF node_representation;
             element_shape: volume_3d_element_shape;
             order: element_order
    ): BOOLEAN;

  LOCAL
    vertex_nodes          : INTEGER;
    edge_face_body_nodes : INTEGER;
    edge_nodes            : INTEGER;
  END_LOCAL;
  IF element_shape = hexahedron THEN
    vertex_nodes := 8;
    IF order = linear THEN
      edge_nodes := 0;
      edge_face_body_nodes := 0;
    END_IF;
    IF order = quadratic THEN
      edge_nodes := 12;
      edge_face_body_nodes := 19;
    END_IF;
    IF order = cubic THEN
      edge_nodes := 24;
      edge_face_body_nodes := 56;
    END_IF;
  END_IF;
  IF element_shape = wedge THEN
    vertex_nodes := 6;
    IF order = linear THEN
      edge_nodes := 0;
      edge_face_body_nodes := 0;
    END_IF;
    IF order = quadratic THEN
      edge_nodes := 9;
      edge_face_body_nodes := 12;
    END_IF;
    IF order = cubic THEN
      edge_nodes := 18;
      edge_face_body_nodes := 34;
    END_IF;
  END_IF;
  IF element_shape = tetrahedron THEN
    vertex_nodes := 4;
    IF order = linear THEN
      edge_nodes := 0;
      edge_face_body_nodes := 0;
    END_IF;
    IF order = quadratic THEN
      edge_nodes := 6;
      edge_face_body_nodes := 6;
    END_IF;
    IF order = cubic THEN
      edge_nodes := 12;
      edge_face_body_nodes := 16;
    END_IF;
  END_IF;
  IF element_shape = pyramid THEN
    vertex_nodes := 5;
    IF order = linear THEN
```



```
      edge_nodes := 0;
      edge_face_body_nodes := 0;
    END_IF;
    IF order = quadratic THEN
      edge_nodes := 8;
      edge_face_body_nodes := 9;
    END_IF;
    IF order = cubic THEN
      edge_nodes := 16;
      edge_face_body_nodes := 25;
    END_IF;
  END_IF;
  RETURN((SIZEOF(node_list) = (vertex_nodes + edge_nodes)) OR (SIZEOF(
      node_list) = (vertex_nodes + edge_face_body_nodes)));

END_FUNCTION; -- required_3d_nodes

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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.OPEN_SHELL' IN TYPEOF(a_shell) THEN
    RETURN(open_shell_reversed(a_shell));
  ELSE
    IF 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.EDGE' IN TYPEOF(an_item) THEN
    RETURN(edge_reversed(an_item));
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.PATH' IN TYPEOF(an_item) THEN
    RETURN(path_reversed(an_item));
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.FACE_BOUND' IN TYPEOF(an_item) THEN
    RETURN(face_bound_reversed(an_item));
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.FACE' IN TYPEOF(an_item) THEN
    RETURN(face_reversed(an_item));
  END_IF;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.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 unique_version_change_order(
             ad: action_directive
    ): BOOLEAN;

  LOCAL
    i      : INTEGER;
    k      : INTEGER;
    vers   : SET OF product_definition_formation := [];
    varset : SET OF versioned_action_request := ad.requests;
    aara   : SET OF applied_action_request_assignment := [];
  END_LOCAL;
  REPEAT i := 1 TO SIZEOF(varset) BY 1;
    aara := aara + QUERY ( ara <* bag_to_set(USEDIN(varset[i],
        'STRUCTURAL_ANALYSIS_DESIGN.' +
        'ACTION_REQUEST_ASSIGNMENT.ASSIGNED_ACTION_REQUEST')) | ((
        'STRUCTURAL_ANALYSIS_DESIGN.' +
        'APPLIED_ACTION_REQUEST_ASSIGNMENT') IN TYPEOF(ara)) );
  END_REPEAT;
  REPEAT k := 1 TO SIZEOF(aara) BY 1;
    vers := vers + bag_to_set(QUERY ( aarai <* aara[k].items | (
        'STRUCTURAL_ANALYSIS_DESIGN.PRODUCT_DEFINITION_FORMATION' IN
        TYPEOF(aarai)) ));
  END_REPEAT;
  RETURN(SIZEOF(QUERY ( ver <* vers | (NOT (SIZEOF(QUERY ( other_ver <*
      (vers - ver) | (ver.of_product :=: other_ver.of_product) )) = 0)) ))
      = 0);

END_FUNCTION; -- unique_version_change_order

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,'')) | ((
      'STRUCTURAL_ANALYSIS_DESIGN.REPRESENTATION_ITEM' IN TYPEOF(z)) OR (
      'STRUCTURAL_ANALYSIS_DESIGN.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,
      'STRUCTURAL_ANALYSIS_DESIGN.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],
          'STRUCTURAL_ANALYSIS_DESIGN.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;
  IF NOT ((1 <= date.day_component) AND (date.day_component <= 31))
       THEN
    RETURN(FALSE);
  END_IF;
  CASE date.month_component OF
    4        :         RETURN((1 <= date.day_component) AND (date.
          day_component <= 30));
    6        :         RETURN((1 <= date.day_component) AND (date.
          day_component <= 30));
    9        :         RETURN((1 <= date.day_component) AND (date.
          day_component <= 30));
    11       :         RETURN((1 <= date.day_component) AND (date.
          day_component <= 30));
    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;
    OTHERWISE  :         RETURN(TRUE);
    END_CASE;

END_FUNCTION; -- valid_calendar_date

FUNCTION valid_geometrically_bounded_wf_curve(
             crv: curve
    ): BOOLEAN;
  IF SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.POLYLINE',
      'STRUCTURAL_ANALYSIS_DESIGN.B_SPLINE_CURVE',
      'STRUCTURAL_ANALYSIS_DESIGN.ELLIPSE',
      'STRUCTURAL_ANALYSIS_DESIGN.CIRCLE'] * TYPEOF(crv)) = 1 THEN
    RETURN(TRUE);
  ELSE
    IF 'STRUCTURAL_ANALYSIS_DESIGN.TRIMMED_CURVE' IN TYPEOF(crv) THEN
      IF SIZEOF(['STRUCTURAL_ANALYSIS_DESIGN.LINE',
          'STRUCTURAL_ANALYSIS_DESIGN.PARABOLA',
          'STRUCTURAL_ANALYSIS_DESIGN.HYPERBOLA'] * TYPEOF(crv\
          trimmed_curve.basis_curve)) = 1 THEN
        RETURN(TRUE);
      ELSE
        RETURN(valid_geometrically_bounded_wf_curve(crv\trimmed_curve.
            basis_curve));
      END_IF;
    ELSE
      IF 'STRUCTURAL_ANALYSIS_DESIGN.OFFSET_CURVE_3D' IN TYPEOF(crv)
           THEN
        RETURN(valid_geometrically_bounded_wf_curve(crv\offset_curve_3d.
            basis_curve));
      ELSE
        IF 'STRUCTURAL_ANALYSIS_DESIGN.CURVE_REPLICA' IN TYPEOF(crv)
             THEN
          RETURN(valid_geometrically_bounded_wf_curve(crv\curve_replica.
              parent_curve));
        ELSE
          IF 'STRUCTURAL_ANALYSIS_DESIGN.COMPOSITE_CURVE' IN TYPEOF(crv)
               THEN
            RETURN(SIZEOF(QUERY ( ccs <* crv\composite_curve.segments |
```

```
                (NOT valid_geometrically_bounded_wf_curve(ccs.
                parent_curve)) )) = 0);
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- valid_geometrically_bounded_wf_curve

FUNCTION valid_geometrically_bounded_wf_point(
             pnt: point
    ): BOOLEAN;
  IF 'STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT' IN TYPEOF(pnt) THEN
    RETURN(TRUE);
  ELSE
    IF 'STRUCTURAL_ANALYSIS_DESIGN.POINT_ON_CURVE' IN TYPEOF(pnt) THEN
      RETURN(valid_geometrically_bounded_wf_curve(pnt\point_on_curve.
          basis_curve));
    ELSE
      IF 'STRUCTURAL_ANALYSIS_DESIGN.POINT_REPLICA' IN TYPEOF(pnt) THEN
        RETURN(valid_geometrically_bounded_wf_point(pnt\point_replica.
            parent_pt));
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- valid_geometrically_bounded_wf_point

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_parametric_coordinate(
             coordinates: LIST [1:3] OF parameter_value
    ): BOOLEAN;

  LOCAL
    i : INTEGER;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(coordinates) BY 1;
    IF (1 < coordinates[i]) OR (coordinates[i] < (-1)) THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  RETURN(TRUE);

END_FUNCTION; -- valid_parametric_coordinate
```



```
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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.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(['STRUCTURAL_ANALYSIS_DESIGN.LINE',
      'STRUCTURAL_ANALYSIS_DESIGN.CONIC',
      'STRUCTURAL_ANALYSIS_DESIGN.B_SPLINE_CURVE',
      'STRUCTURAL_ANALYSIS_DESIGN.POLYLINE'] * TYPEOF(crv)) = 1 THEN
    RETURN(TRUE);
  ELSE
    IF 'STRUCTURAL_ANALYSIS_DESIGN.CURVE_REPLICA' IN TYPEOF(crv) THEN
      RETURN(valid_wireframe_edge_curve(crv\curve_replica.parent_curve));
    ELSE
      IF 'STRUCTURAL_ANALYSIS_DESIGN.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 'STRUCTURAL_ANALYSIS_DESIGN.CARTESIAN_POINT' IN TYPEOF(pnt) THEN
    RETURN(TRUE);
  ELSE
    IF 'STRUCTURAL_ANALYSIS_DESIGN.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 variable_value_type(
             variable: GENERIC
    ): STRING;

  LOCAL
    svt             : STRING;
    feacr           : STRING;
    variable_typeof : SET [1:?] OF STRING;
  END_LOCAL;
  svt := 'STRUCTURAL_ANALYSIS_DESIGN.';
  feacr := 'STRUCTURAL_ANALYSIS_DESIGN.';
  variable_typeof := TYPEOF(variable);
  IF SIZEOF([feacr + 'CURVE_SCALAR_VARIABLE',feacr +
      'SURFACE_SCALAR_VARIABLE',feacr + 'VOLUME_SCALAR_VARIABLE',feacr +
      'BOUNDARY_CURVE_SCALAR_VARIABLE',feacr +
      'BOUNDARY_SURFACE_SCALAR_VARIABLE',feacr +
      'AGGREGATED_SCALAR_VARIABLE',feacr + 'VOLUME_ANGULAR_VARIABLE',
      feacr + 'AGGREGATED_ANGULAR_VARIABLE',feacr +
      'APPLICATION_DEFINED_SCALAR_VARIABLE'] * variable_typeof) = 1 THEN
    RETURN(svt + 'SCALAR');
  END_IF;
```

```
  IF SIZEOF([feacr + 'CURVE_VECTOR_2D_VARIABLE',feacr +
      'SURFACE_VECTOR_2D_VARIABLE',feacr +
      'APPLICATION_DEFINED_VECTOR_2D_VARIABLE'] * variable_typeof) = 1
       THEN
    RETURN(svt + 'TENSOR1_2D');
  END_IF;
  IF SIZEOF([feacr + 'CURVE_VECTOR_3D_VARIABLE',feacr +
      'SURFACE_VECTOR_3D_VARIABLE',feacr + 'VOLUME_VECTOR_3D_VARIABLE',
      feacr + 'BOUNDARY_CURVE_VECTOR_3D_VARIABLE',feacr +
      'BOUNDARY_SURFACE_VECTOR_3D_VARIABLE',feacr +
      'AGGREGATED_VECTOR_3D_VARIABLE',feacr +
      'APPLICATION_DEFINED_VECTOR_3D_VARIABLE'] * variable_typeof) = 1
       THEN
    RETURN(svt + 'TENSOR1_3D');
  END_IF;
  IF SIZEOF([feacr + 'SURFACE_TENSOR2_2D_VARIABLE',feacr +
      'APPLICATION_DEFINED_TENSOR2_2D_VARIABLE'] * variable_typeof) = 1
       THEN
    RETURN(svt + 'SYMMETRIC_TENSOR2_3D');
  END_IF;
  IF SIZEOF([feacr + 'VOLUME_TENSOR2_3D_VARIABLE',feacr +
      'AGGREGATED_TENSOR2_3D_VARIABLE',feacr +
      'APPLICATION_DEFINED_TENSOR2_3D_VARIABLE'] * variable_typeof) = 1
       THEN
    RETURN(svt + 'SYMMETRIC_TENSOR2_3D');
  END_IF;
  RETURN('NO_MATCH');

END_FUNCTION; -- variable_value_type

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 'STRUCTURAL_ANALYSIS_DESIGN.VECTOR' IN TYPEOF(arg1) THEN
        mag1 := arg1.magnitude;
        vec1 := arg1.orientation;
      ELSE
        mag1 := 1;
        vec1 := arg1;
      END_IF;
      IF 'STRUCTURAL_ANALYSIS_DESIGN.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

END_SCHEMA; -- structural_analysis_design


(*
```

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 names	Short names
ACTION	ACTION
ACTION_ASSIGNMENT	ACTASS
ACTION_DIRECTIVE	ACTDRC
ACTION_METHOD	ACTMTH
ACTION_REQUEST_ASSIGNMENT	ACRQAS
ACTION_REQUEST_SOLUTION	ACRQSL
ACTION_REQUEST_STATUS	ACRQST
ACTION_STATUS	ACTSTT
ADDRESS	ADDRSS
ADVANCED_BREP_SHAPE_REPRESENTATION	ABSR
ADVANCED_FACE	ADVFC
ALIGNED_AXIS_TOLERANCE	ALAXTL
ALIGNED_CURVE_3D_ELEMENT_COORDINATE_SYSTEM	AC3ECS
ALIGNED_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM	AS2ECS
ALIGNED_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM	AS3ECS
ALTERNATE_PRODUCT_RELATIONSHIP	ALPRRL
AMOUNT_OF_SUBSTANCE_UNIT	AOSU
AMOUNT_OF_SUBSTANCE_MEASURE_WITH_UNIT	AOSMWU
ANALYSIS_ITEM_WITHIN_REPRESENTATION	AIWR
ANALYSIS_MESSAGE	ANLMSS
ANALYSIS_STEP	ANLSTP
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_CERTIFICATION_ASSIGNMENT	APCRAS
APPLIED_CONTRACT_ASSIGNMENT	APCNAS
APPLIED_DATE_AND_TIME_ASSIGNMENT	ADATA

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
APPLIED_DOCUMENT_REFERENCE	APDCRF
APPLIED_NAME_ASSIGNMENT	APNMAS
APPLIED_ORGANIZATION_ASSIGNMENT	APORAS
APPLIED_PERSON_AND_ORGANIZATION_ASSIGNMENT	APAOA
APPLIED_SECURITY_CLASSIFICATION_ASSIGNMENT	ASCA
APPROVAL	APPRVL
APPROVAL_ASSIGNMENT	APPASS
APPROVAL_DATE_TIME	APDTTM
APPROVAL_PERSON_ORGANIZATION	APPROR
APPROVAL_RELATIONSHIP	APPRLT
APPROVAL_ROLE	APPRL
APPROVAL_STATUS	APPSTT
ARBITRARY_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM	AV2ECS
ARBITRARY_VOLUME_3D_ELEMENT_COORDINATE_SYSTEM	AV3ECS
AREA_MEASURE_WITH_UNIT	AMWU
AREA_UNIT	ARUNT
ASSEMBLY_COMPONENT_USAGE	ASCMUS
ASSEMBLY_COMPONENT_USAGE_SUBSTITUTE	ACUS
AXIS1_PLACEMENT	AX1PLC
AXIS2_PLACEMENT_2D	A2PL2D
AXIS2_PLACEMENT_3D	A2PL3D
AXISYMMETRIC_2D_ELEMENT_PROPERTY	A2EP
AXISYMMETRIC_CURVE_2D_ELEMENT_DESCRIPTOR	AC2ED
AXISYMMETRIC_CURVE_2D_ELEMENT_REPRESENTATION	AC2ER
AXISYMMETRIC_SURFACE_2D_ELEMENT_DESCRIPTOR	AS2ED
AXISYMMETRIC_SURFACE_2D_ELEMENT_REPRESENTATION	AS2ER
AXISYMMETRIC_VOLUME_2D_ELEMENT_REPRESENTATION	AV2ER
AXISYMMETRIC_VOLUME_2D_ELEMENT_DESCRIPTOR	AV2ED
B_SPLINE_CURVE	BSPCR
B_SPLINE_CURVE_WITH_KNOTS	BSCWK
B_SPLINE_SURFACE	BSPSR
B_SPLINE_SURFACE_WITH_KNOTS	BSSWK
BEVELED_SHEET_REPRESENTATION	BVSHRP
BEZIER_CURVE	BZRCRV
BEZIER_SURFACE	BZRSRF
BINARY_GENERIC_EXPRESSION	BNGNEX
BINARY_NUMERIC_EXPRESSION	BNNMEX
BOOLEAN_EXPRESSION	BLNEXP
BOUNDARY_CURVE	BNDCR
BOUNDED_CURVE	BNDCRV

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
BOUNDED_SURFACE	BNDSRF
BREP_WITH_VOIDS	BRWTVD
CALCULATED_STATE	CLCSTT
CALENDAR_DATE	CLNDT
CARTESIAN_POINT	CRTPNT
CARTESIAN_TRANSFORMATION_OPERATOR_3D	CTO3
CARTESIAN_TRANSFORMATION_OPERATOR_2D	CTO2
CARTESIAN_TRANSFORMATION_OPERATOR	CRTROP
CERTIFICATION	CRTFCT
CERTIFICATION_ASSIGNMENT	CRTASS
CERTIFICATION_TYPE	CRTTYP
CHARACTERIZED_OBJECT	CHROBJ
CIRCLE	CIRCLE
COMPARISON_EQUAL	CMPEQL
COMPARISON_EXPRESSION	CMPEXP
CLOSED_SHELL	CLSSHL
COMPOSITE_ASSEMBLY_DEFINITION	CMASDF
COMPOSITE_ASSEMBLY_SEQUENCE_DEFINITION	CASD
COMPOSITE_CURVE	CMPCRV
COMPOSITE_CURVE_ON_SURFACE	CCOS
COMPOSITE_CURVE_SEGMENT	CMCRSG
COMPOSITE_SHEET_REPRESENTATION	CMSHRP
CONFIGURATION_DESIGN	CNFDSG
CONFIGURATION_EFFECTIVITY	CNFEFF
CONFIGURATION_ITEM	CNFITM
CONIC	CONIC
CONICAL_SURFACE	CNCSRF
CONNECTED_EDGE_SET	CNEDST
CONNECTED_FACE_SET	CNFCST
CONSTANT_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM	CS3ECS
CONSTRAINT_ELEMENT	CNSELM
CONTEXT_DEPENDENT_SHAPE_REPRESENTATION	CDSR
CONTEXT_DEPENDENT_UNIT	CNDPUN
CONTRACT	CNTRCT
CONTRACT_ASSIGNMENT	CNTASS
CONTRACT_TYPE	CNTTYP
CONTROL	CNTRL
CONTROL_ANALYSIS_STEP	CNANST
CONTROL_LINEAR_MODES_AND_FREQUENCIES_PROCESS	CLMAFP
CONTROL_LINEAR_MODES_AND_FREQUENCIES_ANALYSIS_STEP	CLMAFA

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
CONTROL_LINEAR_STATIC_ANALYSIS_STEP	CLSAS
CONTROL_LINEAR_STATIC_ANALYSIS_STEP_WITH_HARMONIC	CLSASW
CONTROL_LINEAR_STATIC_LOAD_INCREMENT_PROCESS	CLSLIP
CONTROL_PROCESS	CNTPRC
CONTROL_RESULT_RELATIONSHIP	CNRSRL
CONVERSION_BASED_UNIT	CNBSUN
COORDINATED_UNIVERSAL_TIME_OFFSET	CUTO
CURVE	CURVE
CURVE_2D_ELEMENT_BASIS	C2EB
CURVE_2D_ELEMENT_CONSTANT_SPECIFIED_VARIABLE_VALUE	C2CSV
CURVE_2D_ELEMENT_COORDINATE_SYSTEM	C2ECS
CURVE_2D_ELEMENT_CONSTANT_SPECIFIED_VOLUME_VARIABLE_-VALUE	C2ECSV
CURVE_2D_ELEMENT_FIELD_VARIABLE_DEFINITION	C2EFVD
CURVE_2D_ELEMENT_GROUP	C2EG
CURVE_2D_ELEMENT_INTEGRATED_MATRIX	C2EIM
CURVE_2D_ELEMENT_INTEGRATED_MATRIX_WITH_DEFINITION	C2EIMW
CURVE_2D_ELEMENT_INTEGRATION	C2EI
CURVE_2D_ELEMENT_LOCATION_POINT_VARIABLE_VALUES	C2E0
CURVE_2D_ELEMENT_LOCATION_POINT_VOLUME_VARIABLE_VALUES	C2ELPV
CURVE_2D_ELEMENT_PROPERTY	C2EP
CURVE_2D_ELEMENT_VALUE_AND_VOLUME_LOCATION	C2EVAV
CURVE_2D_ELEMENT_VALUE_AND_LOCATION	C2EVAL
CURVE_2D_NODE_FIELD_AGGREGATED_VARIABLE_VALUES	C2NFAV
CURVE_2D_NODE_FIELD_SECTION_VARIABLE_VALUES	C2NFSV
CURVE_2D_NODE_FIELD_VARIABLE_DEFINITION	C2NFVD
CURVE_2D_SUBSTRUCTURE_ELEMENT_REFERENCE	C2SER
CURVE_2D_WHOLE_ELEMENT_VARIABLE_VALUE	C2WEVV
CURVE_3D_ELEMENT_BASIS	C3EB
CURVE_3D_ELEMENT_CONSTANT_SPECIFIED_VARIABLE_VALUE	C3ECSV
CURVE_3D_ELEMENT_CONSTANT_SPECIFIED_VOLUME_VARIABLE_-VALUE	C3CSV
CURVE_3D_ELEMENT_DESCRIPTOR	C3ED
CURVE_3D_ELEMENT_FIELD_VARIABLE_DEFINITION	C3EFVD
CURVE_3D_ELEMENT_GROUP	C3EG
CURVE_3D_ELEMENT_INTEGRATION	C3EI
CURVE_3D_ELEMENT_INTEGRATED_MATRIX	C3EIM
CURVE_3D_ELEMENT_INTEGRATED_MATRIX_WITH_DEFINITION	C3EIMW
CURVE_3D_ELEMENT_LENGTH_INTEGRATION_RULE	C3ELIR
CURVE_3D_ELEMENT_LENGTH_INTEGRATION_EXPLICIT	C3ELIE

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
CURVE_3D_ELEMENT_LOCATION_POINT_VOLUME_VARIABLE_VALUES	C3ELPV
CURVE_3D_ELEMENT_LOCATION_POINT_VARIABLE_VALUES	C3LPV
CURVE_3D_ELEMENT_NODAL_SPECIFIED_VARIABLE_VALUES	C3ENSV
CURVE_3D_ELEMENT_POSITION_WEIGHT	C3EPW
CURVE_3D_ELEMENT_PROPERTY	C3EP
CURVE_3D_ELEMENT_REPRESENTATION	C3ER
CURVE_3D_ELEMENT_VALUE_AND_VOLUME_LOCATION	C3EVAV
CURVE_3D_ELEMENT_VALUE_AND_LOCATION	C3EVAL
CURVE_3D_NODE_FIELD_SECTION_VARIABLE_VALUES	C3NFSV
CURVE_3D_NODE_FIELD_AGGREGATED_VARIABLE_VALUES	C3NFAV
CURVE_3D_NODE_FIELD_VARIABLE_DEFINITION	C3NFVD
CURVE_3D_SUBSTRUCTURE_ELEMENT_REFERENCE	C3SER
CURVE_3D_WHOLE_ELEMENT_VARIABLE_VALUE	C3WEVV
CURVE_BOUNDED_SURFACE	CRBNSR
CURVE_CONSTRAINT	CRVCNS
CURVE_ELEMENT_END_OFFSET	CEEO
CURVE_ELEMENT_END_RELEASE	CEER
CURVE_ELEMENT_END_RELEASE_PACKET	CEERP
CURVE_ELEMENT_INTERVAL_CONSTANT	CEIC
CURVE_ELEMENT_INTERVAL_LINEARLY_VARYING	CEILV
CURVE_ELEMENT_INTERVAL	CRELIN
CURVE_ELEMENT_LOCATION	CRELLC
CURVE_ELEMENT_SECTION_DEFINITION	CESD
CURVE_ELEMENT_SECTION_DERIVED_DEFINITIONS	CESDD
CURVE_FREEDOM_ACTION_DEFINITION	CFAD
CURVE_FREEDOM_AND_VALUE_DEFINITION	CFAVD
CURVE_FREEDOM_VALUES	CRFRVL
CURVE_REPLICA	CRVRPL
CURVE_SECTION_ELEMENT_LOCATION	CSEL
CURVE_SECTION_INTEGRATION_EXPLICIT	CSIE
CURVE_VOLUME_ELEMENT_LOCATION	CVEL
CYLINDRICAL_POINT	CYLPNT
CYLINDRICAL_SURFACE	CYLSRF
CYLINDRICAL_SYMMETRY_CONTROL	CYSYCN
DATA_ENVIRONMENT	DTENV
DATA_ENVIRONMENT_RELATIONSHIP	DTENRL
DATE	DATE
DATE_AND_TIME	DTANTM
DATE_AND_TIME_ASSIGNMENT	DATA
DATE_ASSIGNMENT	DTASS

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
DATE_ROLE	DTRL
DATE_TIME_ROLE	DTTMRL
DATED_EFFECTIVITY	DTDEFF
DEFINITIONAL_REPRESENTATION	DFNRPR
DEGENERATE_PCURVE	DGNPCR
DEGENERATE_TOROIDAL_SURFACE	DGTRSR
DERIVED_UNIT	DRVUNT
DERIVED_UNIT_ELEMENT	DRUNEL
DESCRIPTIVE_REPRESENTATION_ITEM	DSRPIT
DESIGN_MAKE_FROM_RELATIONSHIP	DMFR
DIMENSIONAL_EXPONENTS	DMNEXP
DIMENSIONAL_SIZE	DMNSZ
DIRECTED_ACTION	DRCACT
DIRECTION	DRCTN
DIRECTION_NODE	DRCND
DIRECTIONALLY_EXPLICIT_ELEMENT_COEFFICIENT	DEEC
DIRECTIONALLY_EXPLICIT_ELEMENT_COORDINATE_SYSTEM_ALIGNED	DCS
DIRECTIONALLY_EXPLICIT_ELEMENT_COORDINATE_SYSTEM_-ARBITRARY	DEECSA
DIRECTIONALLY_EXPLICIT_ELEMENT_REPRESENTATION	DEER
DIV_EXPRESSION	DVEXP
DOCUMENT	DCMNT
DOCUMENT_REFERENCE	DCMRFR
DOCUMENT_RELATIONSHIP	DCMRLT
DOCUMENT_TYPE	DCMTYP
DOCUMENT_USAGE_CONSTRAINT	DCUSCN
DOCUMENT_WITH_CLASS	DCWTCL
DRAPED_DEFINED_TRANSFORMATION	DRDFTR
DUMMY_NODE	DMMND
EDGE	EDGE
EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION	EBWSR
EDGE_BASED_WIREFRAME_MODEL	EBWM
EDGE_CURVE	EDGCRV
EDGE_LOOP	EDGLP
EFFECTIVITY	EFFCTV
ELEMENT_ANALYSIS_MESSAGE	ELANMS
ELEMENT_DEFINITION	ELMDFN
ELEMENT_DESCRIPTOR	ELMDSC
ELEMENT_GEOMETRIC_RELATIONSHIP	ELGMRL
ELEMENT_GROUP	ELMGRP

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
ELEMENT_GROUP_ANALYSIS_MESSAGE	EGAM
ELEMENT_MATERIAL	ELMMTR
ELEMENT_NODAL_FREEDOM_ACTIONS	ENFA
ELEMENT_NODAL_FREEDOM_TERMS	ENFT
ELEMENT_REPRESENTATION	ELMRPR
ELEMENT_SEQUENCE	ELMSQN
ELEMENTARY_SURFACE	ELMSRF
ELLIPSE	ELLPS
ENVIRONMENT	ENVRNM
EULER_ANGLES	ELRANG
EVALUATED_DEGENERATE_PCURVE	EVDGPC
EXECUTED_ACTION	EXCACT
EXPANDED_UNCERTAINTY	EXPUNC
EXPLICIT_ELEMENT_MATRIX	EXELMT
EXPLICIT_ELEMENT_REPRESENTATION	EXELRP
EXPRESSION	EXPRSS
EXPRESSION_CONVERSION_BASED_UNIT	ECBU
FACE	FACE
FACE_BOUND	FCBND
FACE_OUTER_BOUND	FCOTBN
FACE_SURFACE	FCSRF
FACETED_BREP	FCTBR
FACETED_BREP_SHAPE_REPRESENTATION	FBSR
FEA_AREA_DENSITY	FARDN
FEA_AXIS2_PLACEMENT_2D	FAP2
FEA_AXIS2_PLACEMENT_3D	FAP3
FEA_CURVE_SECTION_GEOMETRIC_RELATIONSHIP	FCSGR
FEA_GROUP	FGRP
FEA_GROUP_RELATION	FGRRL
FEA_LINEAR_ELASTICITY	FLNEL
FEA_MASS_DENSITY	FMSDN
FEA_MATERIAL_PROPERTY_REPRESENTATION	FMPR
FEA_MATERIAL_PROPERTY_REPRESENTATION_ITEM	FMPRI
FEA_MATERIAL_PROPERTY_GEOMETRIC_RELATIONSHIP	FMPGR
FEA_MODEL	FMDL
FEA_MODEL_2D	FMD2D
FEA_MODEL_3D	FMD3D
FEA_MODEL_DEFINITION	FMDDF
FEA_MOISTURE_ABSORPTION	FMSAB

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
FEA_PARAMETRIC_POINT	FPRPN
FEA_REPRESENTATION_ITEM	FRPIT
FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION	FSCOLT
FEA_SHELL_BENDING_STIFFNESS	FSBS
FEA_SHELL_MEMBRANE_BENDING_COUPLING_STIFFNESS	FSMBCS
FEA_SHELL_MEMBRANE_STIFFNESS	FSMS
FEA_SHELL_SHEAR_STIFFNESS	FSSS
FEA_SURFACE_SECTION_GEOMETRIC_RELATIONSHIP	FSSGR
FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION	FTCOLT
FIELD_VARIABLE_DEFINITION	FLVRDF
FIELD_VARIABLE_ELEMENT_DEFINITION	FVED
FIELD_VARIABLE_ELEMENT_GROUP_VALUE	FVEGV
FIELD_VARIABLE_NODE_DEFINITION	FVND
FIELD_VARIABLE_WHOLE_MODEL_VALUE	FVWMV
FLAT_PATTERN_PLY_REPRESENTATION_RELATIONSHIP	FPPRR
FOUNDED_ITEM	FNDITM
FREEDOM_AND_COEFFICIENT	FRANCF
FREEDOMS_LIST	FRDLST
FUNCTIONALLY_DEFINED_TRANSFORMATION	FNDFTR
GENERIC_EXPRESSION	GNREXP
GENERIC_LITERAL	GNRLTR
GENERIC_VARIABLE	GNRVRB
GEOMETRIC_CURVE_SET	GMCRST
GEOMETRIC_NODE	GMTND
GEOMETRIC_REPRESENTATION_CONTEXT	GMRPCN
GEOMETRIC_REPRESENTATION_ITEM	GMRPIT
GEOMETRIC_SET	GMTST
GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION	GBSSR
GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_REPRESENTATION	GBWSR
GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT	GC
GLOBAL_UNIT_ASSIGNED_CONTEXT	GUAC
GROUNDED_DAMPER	GRNDMP
GROUNDED_SPRING	GRNSPR
GROUP	GROUP
GROUP_RELATIONSHIP	GRPRLT
HYPERBOLA	HYPRBL
INTERSECTION_CURVE	INTCRV
INTERVAL_EXPRESSION	INTEXP
ITEM_DEFINED_TRANSFORMATION	ITDFTR
LAID_DEFINED_TRANSFORMATION	LDDFTR

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
LENGTH_MEASURE_WITH_UNIT	LMWU
LENGTH_UNIT	LNGUNT
LINE	LINE
LINEAR_CONSTRAINT_EQUATION_NODAL_TERM	LCENT
LINEAR_CONSTRAINT_EQUATION_ELEMENT_VALUE	LCEEV
LINEAR_CONSTRAINT_EQUATION_ELEMENT	LCEE
LINEARLY_SUPERIMPOSED_STATE	LNSPST
LITERAL_NUMBER	LTRNMB
LOCAL_TIME	LCLTM
LOOP	LOOP
LOT_EFFECTIVITY	LTEFF
MAKE_FROM_USAGE_OPTION	MFUO
MANIFOLD_SOLID_BREP	MNSLBR
MANIFOLD_SURFACE_SHAPE_REPRESENTATION	MSSR
MAPPED_ITEM	MPPITM
MASS_MEASURE_WITH_UNIT	MMWU
MASS_UNIT	MSSUNT
MATERIAL_DESIGNATION	MTRDSG
MATERIAL_DESIGNATION_CHARACTERIZATION	MTDSCH
MATERIAL_PROPERTY	MTRPRP
MATERIAL_PROPERTY_REPRESENTATION	MTPRRP
MEASURE_QUALIFICATION	MSRQLF
MEASURE_REPRESENTATION_ITEM	MSRPIT
MEASURE_WITH_UNIT	MSWTUN
MINUS_EXPRESSION	MNSEXP
MULTIPLE_ARITY_BOOLEAN_EXPRESSION	MABE
MULTIPLE_ARITY_GENERIC_EXPRESSION	MAGE
MULTIPLE_ARITY_NUMERIC_EXPRESSION	MANE
MULT_EXPRESSION	MLTEXP
NAME_ASSIGNMENT	NMASS
NAMED_UNIT	NMDUNT
NAMED_UNIT_VARIABLE	NMUNVR
NEXT_ASSEMBLY_USAGE_OCCURRENCE	NAUO
NO_SYMMETRY_CONTROL	NSYCN
NODAL_DOF_REDUCTION	NDDFRD
NODAL_FREEDOM_ACTION_DEFINITION	NFAD
NODAL_FREEDOM_AND_VALUE_DEFINITION	NFAVD
NODAL_FREEDOM_VALUES	NDFRVL
NODE	NODE
NODE_ANALYSIS_MESSAGE	NDANMS

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
NODE_DEFINITION	NDDFN
NODE_GEOMETRIC_RELATIONSHIP	NDGMRL
NODE_GROUP	NDGRP
NODE_REPRESENTATION	NDRPR
NODE_SEQUENCE	NDSQN
NODE_SET	NDST
NODE_WITH_SOLUTION_COORDINATE_SYSTEM	NWSCS
NODE_WITH_VECTOR	NDWTVC
NUMERIC_EXPRESSION	NMREXP
NUMERIC_VARIABLE	MNRVRB
OFFSET_CURVE_2D	OFCR2D
OFFSET_CURVE_3D	OFCR3D
OFFSET_SURFACE	OFFSRF
OPEN_SHELL	OPNSHL
ORDINAL_DATE	ORDDT
ORGANIZATION	ORGNZT
ORGANIZATION_ASSIGNMENT	ORGASS
ORGANIZATION_RELATIONSHIP	ORGRLT
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
OUTER_BOUNDARY_CURVE	OTBNCR
OUTPUT_REQUEST_STATE	OTRQST
PARABOLA	PRBL
PARAMETRIC_CURVE_3D_ELEMENT_COORDINATE_DIRECTION	PC3ECD
PARAMETRIC_CURVE_3D_ELEMENT_COORDINATE_SYSTEM	PC3ECS
PARAMETRIC_REPRESENTATION_CONTEXT	PRRPCN
PARAMETRIC_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM	PS3ECS
PARAMETRIC_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM	PS2ECS
PARAMETRIC_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM	PV2ECS
PARAMETRIC_VOLUME_3D_ELEMENT_COORDINATE_SYSTEM	PV3ECS
PATH	PATH
PCURVE	PCURVE
PERCENTAGE_LAMINATE_DEFINITION	PRLMDF
PERCENTAGE_PLY_DEFINITION	PRPLDF

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
PERSON	PERSON
PERSON_AND_ORGANIZATION	PRANOR
PERSON_AND_ORGANIZATION_ASSIGNMENT	PAOA
PERSON_AND_ORGANIZATION_ROLE	PAOR
PERSONAL_ADDRESS	PRSADD
PLACEMENT	PLCMNT
PLANE	PLANE
PLANE_2D_ELEMENT_PROPERTY	P2EP
PLANE_ANGLE_MEASURE_WITH_UNIT	PAMWU
PLANE_ANGLE_UNIT	PLANUN
PLANE_CURVE_2D_ELEMENT_DESCRIPTOR	PC2ED
PLANE_CURVE_2D_ELEMENT_REPRESENTATION	PC2ER
PLANE_SURFACE_2D_ELEMENT_DESCRIPTOR	PS2ED
PLANE_SURFACE_2D_ELEMENT_REPRESENTATION	PS2ER
PLANE_VOLUME_2D_ELEMENT_REPRESENTATION	PV2ER
PLANE_VOLUME_2D_ELEMENT_DESCRIPTOR	PV2ED
PLY_LAMINATE_DEFINITION	PLLMDF
PLY_LAMINATE_SEQUENCE_DEFINITION	PLS0
PLUS_EXPRESSION	PLSEXP
POINT	POINT
POINT_AND_VECTOR	PNANVC
POINT_CONSTRAINT	PNTCNS
POINT_ELEMENT_MATRIX	PNELMT
POINT_ELEMENT_REPRESENTATION	PNELRP
POINT_FREEDOM_ACTION_DEFINITION	PFAD
POINT_FREEDOM_AND_VALUE_DEFINITION	PFAVD
POINT_FREEDOM_VALUES	PNFRVL
POINT_ON_CURVE	PNONCR
POINT_ON_SURFACE	PNONSR
POINT_PATH	PNTPTH
POINT_REPLICA	PNTRPL
POINT_REPRESENTATION	PNTRPR
POLY_LOOP	PLYLP
POLYLINE	PLYLN
PRECISION_QUALIFIER	PRCQLF
PRODUCT	PRDCT
PRODUCT_CATEGORY	PRDCTG
PRODUCT_CATEGORY_RELATIONSHIP	PRCTRL
PRODUCT_CONCEPT	PRDCNC
PRODUCT_CONCEPT_CONTEXT	PRCNCN

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
PRODUCT_CONTEXT	PRDCNT
PRODUCT_DEFINITION	PRDDFN
PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE	PDFWSS
PRODUCT_DEFINITION_SHAPE	PRDFSH
PRODUCT_DEFINITION_EFFECTIVITY	PRDFEF
PRODUCT_DEFINITION_RELATIONSHIP	PRDFRL
PRODUCT_DEFINITION_CONTEXT	PRDFCN
PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS	PDWAD
PRODUCT_DEFINITION_FORMATION	PRDFFR
PRODUCT_DEFINITION_FORMATION_RELATIONSHIP	PDFR
PRODUCT_DEFINITION_USAGE	PRDFUS
PRODUCT_MATERIAL_COMPOSITION_RELATIONSHIP	PMCR
PRODUCT_RELATED_PRODUCT_CATEGORY	PRPC
PROMISSORY_USAGE_OCCURRENCE	PRUSOC
PROPERTY_DEFINITION	PRPDFN
PROPERTY_DEFINITION_REPRESENTATION	PRDFRP
QUALIFIED_REPRESENTATION_ITEM	QLRPIT
QUALITATIVE_UNCERTAINTY	QLTUNC
QUANTIFIED_ASSEMBLY_COMPONENT_USAGE	QACU
QUASI_UNIFORM_CURVE	QSUNCR
QUASI_UNIFORM_SURFACE	QSUNSR
RATIO_UNIT	RTUNT
RATIONAL_B_SPLINE_CURVE	RBSC
RATIONAL_B_SPLINE_SURFACE	RBSS
REAL_LITERAL	RLLTR
REAL_NUMERIC_VALUE	RLNMVR
RECTANGULAR_COMPOSITE_SURFACE	RCCMSR
RECTANGULAR_TRIMMED_SURFACE	RCTRSR
REPARAMETRISED_COMPOSITE_CURVE_SEGMENT	RCCS
REPRESENTATION	RPRSNT
REPRESENTATION_CONTEXT	RPRCNT
REPRESENTATION_ITEM	RPRITM
REPRESENTATION_MAP	RPRMP
REPRESENTATION_RELATIONSHIP	RPRRLT
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION	RRWT
RESULT	RESULT
RESULT_ANALYSIS_STEP	RSANST
RESULT_LINEAR_MODES_AND_FREQUENCIES_ANALYSIS_SUB_STEP	RLMAFA
RESULT_LINEAR_STATIC_ANALYSIS_SUB_STEP	RLSASS
RETENTION	RTNTN

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
RETENTION_ASSIGNMENT	RTNASS
SEAM_CURVE	SMCRV
SECURITY_CLASSIFICATION_LEVEL	SCCLLV
SECURITY_CLASSIFICATION_ASSIGNMENT	SCCLAS
SECURITY_CLASSIFICATION	SCRCLS
SERIAL_NUMBERED_EFFECTIVITY	SRNMEF
SHAPE_ASPECT	SHPASP
SHAPE_ASPECT_RELATIONSHIP	SHASRL
SHAPE_DEFINITION_REPRESENTATION	SHDFRP
SHAPE_DIMENSION_REPRESENTATION	SHDMRP
SHAPE_REPRESENTATION_RELATIONSHIP	SHRPRL
SHAPE_REPRESENTATION	SHPRPR
SHELL_BASED_SURFACE_MODEL	SBSM
SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION	SBWSR
SHELL_BASED_WIREFRAME_MODEL	SBWM
SI_UNIT	SUNT
SIMPLE_GENERIC_EXPRESSION	SMGNEX
SIMPLE_NUMERIC_EXPRESSION	SMNMEX
SIMPLE_PLANE_2D_ELEMENT_PROPERTY	SP2EP
SINGLE_POINT_CONSTRAINT_ELEMENT	SPCE
SINGLE_POINT_CONSTRAINT_ELEMENT_VALUES	SPCEV
SMEARED_MATERIAL_DEFINITION	SMMTDF
SOLID_ANGLE_MEASURE_WITH_UNIT	SAMWU
SOLID_ANGLE_UNIT	SLANUN
SOLID_CONSTRAINT	SLDCNS
SOLID_FREEDOM_ACTION_DEFINITION	SFD
SOLID_FREEDOM_AND_VALUE_DEFINITION	SFVD
SOLID_FREEDOM_VALUES	SLFRVL
SOLID_MODEL	SLDMDL
SPECIFIED_HIGHER_USAGE_OCCURRENCE	SHUO
SPECIFIED_STATE	SPCSTT
SPHERICAL_POINT	SPHPNT
SPHERICAL_SURFACE	SPHSRF
STANDARD_UNCERTAINTY	STNUNC
STATE	STATE
STATE_COMPONENT	STTCMP
STATE_DEFINITION	STTDFN
STATE_RELATIONSHIP	STTRLT
STATE_WITH_HARMONIC	STWTHR
STATIONARY_MASS	STTMSS

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
STRUCTURAL_RESPONSE_PROPERTY_DEFINITION_REPRESENTATION	SRPDR
STRUCTURAL_RESPONSE_PROPERTY	STRSPR
SUBSTRUCTURE_ELEMENT_REPRESENTATION	SBELRP
SUBSTRUCTURE_NODE_RELATIONSHIP	SBNDRL
SUBSTRUCTURE_NODE_REFERENCE	SBNDRF
SUPPLIED_PART_RELATIONSHIP	SPPRRL
SURFACE	SRFC
SURFACE_2D_ELEMENT_BOUNDARY_EDGE_LOCATION_POINT_-VARIABLE_VALUES	S2EBEL
SURFACE_2D_ELEMENT_BOUNDARY_EDGE_CONSTANT_SPECIFIED_-VARIABLE_VALUE	S2EBEC
SURFACE_2D_ELEMENT_INTEGRATION	S2EI
SURFACE_2D_ELEMENT_BOUNDARY_EDGE_NODAL_SPECIFIED_-VARIABLE_VALUES	S2EBEN
SURFACE_2D_ELEMENT_BOUNDARY_EDGE_WHOLE_EDGE_VARIABLE_-VALUE	S2EBEW
SURFACE_2D_ELEMENT_BASIS	S2EB
SURFACE_2D_ELEMENT_BOUNDARY_LOCATION_POINT_SURFACE_-VARIABLE_VALUES	S2EBLP
SURFACE_2D_ELEMENT_BOUNDARY_NODAL_SPECIFIED_VARIABLE_-VALUES	S2EBNS
SURFACE_2D_ELEMENT_BOUNDARY_EDGE_CONSTANT_SPECIFIED_-SURFACE_VARIABLE_VALUE	S2BC
SURFACE_2D_ELEMENT_BOUNDARY_CONSTANT_SPECIFIED_-VARIABLE_VALUE	S2EBCS
SURFACE_2D_ELEMENT_BOUNDARY_WHOLE_FACE_VARIABLE_VALUE	S2EBWF
SURFACE_2D_ELEMENT_CONSTANT_SPECIFIED_VARIABLE_VALUE	S2ECSV
SURFACE_2D_ELEMENT_FIELD_VARIABLE_DEFINITION	S2EFVD
SURFACE_2D_ELEMENT_GROUP	S2EG
SURFACE_2D_ELEMENT_BOUNDARY_EDGE_LOCATION_POINT_-SURFACE_VARIABLE_VALUES	S2BL
SURFACE_2D_ELEMENT_INTEGRATED_MATRIX	S2EIM
SURFACE_2D_ELEMENT_INTEGRATED_MATRIX_WITH_DEFINITION	S2EIMW
SURFACE_2D_ELEMENT_LENGTH_INTEGRATION_EXPLICIT	S2ELIE
SURFACE_2D_ELEMENT_LENGTH_INTEGRATION_RULE	S2ELIR
SURFACE_2D_ELEMENT_LOCATION_POINT_VARIABLE_VALUES	S2ELPV
SURFACE_2D_ELEMENT_NODAL_SPECIFIED_VARIABLE_VALUES	S2ENSV
SURFACE_2D_ELEMENT_VALUE_AND_LOCATION	S2EVAL
SURFACE_2D_ELEMENT_VALUE_AND_VOLUME_LOCATION	S2EVAV
SURFACE_2D_ELEMENT_LOCATION_POINT_VOLUME_VARIABLE_-VALUES	S2LPV

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
SURFACE_2D_ELEMENT_BOUNDARY_CONSTANT_SPECIFIED_SURFACE_-VARIABLE_VALUE	S2BCS
SURFACE_2D_ELEMENT_CONSTANT_SPECIFIED_VOLUME_VARIABLE_-VALUE	S2CSV
SURFACE_2D_NODE_FIELD_VARIABLE_DEFINITION	S2NFVD
SURFACE_2D_NODE_FIELD_SECTION_VARIABLE_VALUES	S2NFSV
SURFACE_2D_NODE_FIELD_AGGREGATED_VARIABLE_VALUES	S2NFAV
SURFACE_2D_SUBSTRUCTURE_ELEMENT_REFERENCE	S2SER
SURFACE_2D_WHOLE_ELEMENT_VARIABLE_VALUE	S2WEVV
SURFACE_3D_ELEMENT_VALUE_AND_VOLUME_LOCATION	S3EVAV
SURFACE_3D_ELEMENT_BOUNDARY_EDGE_CONSTANT_SPECIFIED_-VARIABLE_VALUE	S3BC
SURFACE_3D_ELEMENT_BOUNDARY_CONSTANT_SPECIFIED_-VARIABLE_VALUE	S3BCS
SURFACE_3D_ELEMENT_BOUNDARY_EDGE_LOCATION_POINT_-SURFACE_VARIABLE_VALUES	S3BL
SURFACE_3D_ELEMENT_CONSTANT_SPECIFIED_VOLUME_VARIABLE_-VALUE	S3CSV
SURFACE_3D_ELEMENT_BASIS	S3EB
SURFACE_3D_ELEMENT_BOUNDARY_CONSTANT_SPECIFIED_SURFACE_-VARIABLE_VALUE	S3EBCS
SURFACE_3D_ELEMENT_BOUNDARY_EDGE_CONSTANT_SPECIFIED_-SURFACE_VARIABLE_VALUE	S3EBEC
SURFACE_3D_ELEMENT_BOUNDARY_EDGE_LOCATION_POINT_-VARIABLE_VALUES	S3EBEL
SURFACE_3D_ELEMENT_BOUNDARY_EDGE_NODAL_SPECIFIED_-VARIABLE_VALUES	S3EBEN
SURFACE_3D_ELEMENT_LOCATION_POINT_VOLUME_VARIABLE_-VALUES	S3LPV
SURFACE_3D_ELEMENT_BOUNDARY_NODAL_SPECIFIED_VARIABLE_-VALUES	S3EBNS
SURFACE_3D_ELEMENT_VALUE_AND_LOCATION	S3EVAL
SURFACE_3D_ELEMENT_REPRESENTATION	S3ER
SURFACE_3D_ELEMENT_NODAL_SPECIFIED_VARIABLE_VALUES	S3ENSV
SURFACE_3D_ELEMENT_LOCATION_POINT_VARIABLE_VALUES	S3ELPV
SURFACE_3D_ELEMENT_BOUNDARY_EDGE_WHOLE_EDGE_VARIABLE_-VALUE	S3EBEW
SURFACE_3D_ELEMENT_INTEGRATION	S3EI
SURFACE_3D_ELEMENT_GROUP	S3EG
SURFACE_3D_ELEMENT_BOUNDARY_LOCATION_POINT_SURFACE_-VARIABLE_VALUES	S3EBLP
SURFACE_3D_ELEMENT_FIELD_INTEGRATION_EXPLICIT	S3EFIE

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
SURFACE_3D_ELEMENT_CONSTANT_SPECIFIED_VARIABLE_VALUE	S3ECSV
SURFACE_3D_ELEMENT_BOUNDARY_WHOLE_FACE_VARIABLE_VALUE	S3EBWF
SURFACE_3D_ELEMENT_INTEGRATED_MATRIX	S3EIM
SURFACE_3D_ELEMENT_INTEGRATED_MATRIX_WITH_DEFINITION	S3EIMW
SURFACE_3D_ELEMENT_FIELD_INTEGRATION_RULE	S3EFIR
SURFACE_3D_ELEMENT_FIELD_VARIABLE_DEFINITION	S3EFVD
SURFACE_3D_ELEMENT_DESCRIPTOR	S3ED
SURFACE_3D_NODE_FIELD_VARIABLE_DEFINITION	S3NFVD
SURFACE_3D_NODE_FIELD_AGGREGATED_VARIABLE_VALUES	S3NFAV
SURFACE_3D_NODE_FIELD_SECTION_VARIABLE_VALUES	S3NFSV
SURFACE_3D_SUBSTRUCTURE_ELEMENT_REFERENCE	S3SER
SURFACE_3D_WHOLE_ELEMENT_VARIABLE_VALUE	S3WEVV
SURFACE_CONSTRAINT	SRFCNS
SURFACE_CURVE	SRFCRV
SURFACE_ELEMENT_LOCATION	SRELLC
SURFACE_ELEMENT_PROPERTY	SRELPR
SURFACE_FREEDOM_ACTION_DEFINITION	SFAD
SURFACE_FREEDOM_AND_VALUE_DEFINITION	SFAVD
SURFACE_FREEDOM_VALUES	SRFRVL
SURFACE_OF_LINEAR_EXTRUSION	SL
SURFACE_OF_REVOLUTION	SROFRV
SURFACE_PATCH	SRFPTC
SURFACE_POSITION_WEIGHT	SRPSWG
SURFACE_REPLICA	SRFRPL
SURFACE_SECTION	SRFSCT
SURFACE_SECTION_ELEMENT_LOCATION_DIMENSIONLESS	SSELD
SURFACE_SECTION_ELEMENT_LOCATION	SSEL
SURFACE_SECTION_ELEMENT_LOCATION_ABSOLUTE	SSELA
SURFACE_SECTION_FIELD_VARYING	SSFV
SURFACE_SECTION_FIELD_CONSTANT	SSFC
SURFACE_SECTION_FIELD	SRSCFL
SURFACE_SECTION_INTEGRATION_EXPLICIT	SSIE
SURFACE_SECTION_INTEGRATION_RULE	SSIR
SURFACE_SECTION_POSITION_WEIGHT	SSPW
SURFACE_VOLUME_ELEMENT_LOCATION	SVEL
SWEPT_SURFACE	SWPSRF
SYMMETRY_CONTROL	SYMCNT
SYSTEM_AND_FREEDOM	SYANFR
TENSOR_REPRESENTATION_ITEM	TNRPIT
THERMODYNAMIC_TEMPERATURE_MEASURE_WITH_UNIT	TTMWU

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
THERMODYNAMIC_TEMPERATURE_UNIT	THTMUN
THICKNESS_LAMINATE_DEFINITION	THLMDF
TIME_MEASURE_WITH_UNIT	TMWU
TIME_UNIT	TMUNT
TOPOLOGICAL_REPRESENTATION_ITEM	TPRPIT
TOROIDAL_SURFACE	TRDSRF
TRANSFORMATION_WITH_DERIVED_ANGLE	TWDA
TRIMMED_CURVE	TRMCRV
TYPE_QUALIFIER	TYPQLF
UNARY_GENERIC_EXPRESSION	UNGNEX
UNARY_NUMERIC_EXPRESSION	UNNMEX
UNCERTAINTY_MEASURE_WITH_UNIT	UMWU
UNCERTAINTY_QUALIFIER	UNCQLF
UNIFORM_CURVE	UNFCRV
UNIFORM_SURFACE	UNFSRF
UNIFORM_SURFACE_SECTION_LAYERED	USSL
UNIFORM_SURFACE_SECTION	UNSRSC
VARIABLE	VRBL
VARIABLE_SEMANTICS	VRBSMN
VECTOR	VECTOR
VERSIONED_ACTION_REQUEST	VRACRQ
VERTEX	VERTEX
VERTEX_LOOP	VRTLP
VERTEX_POINT	VRTPNT
VERTEX_SHELL	VRTSHL
VOLUME_2D_ELEMENT_BOUNDARY_NODAL_SPECIFIED_VARIABLE_-VALUES	V2EBNS
VOLUME_2D_ELEMENT_BOUNDARY_EDGE_LOCATION_POINT_-VOLUME_VARIABLE_VALUES	V2EBEL
VOLUME_2D_ELEMENT_BOUNDARY_EDGE_CONSTANT_SPECIFIED_-VOLUME_VARIABLE_VALUE	V2EBEC
VOLUME_2D_ELEMENT_BOUNDARY_EDGE_WHOLE_EDGE_VARIABLE_-VALUE	V2EBEW
VOLUME_2D_ELEMENT_BOUNDARY_EDGE_NODAL_SPECIFIED_-VARIABLE_VALUES	V2EBEN
VOLUME_2D_ELEMENT_BOUNDARY_WHOLE_FACE_VARIABLE_VALUE	V2EBWF
VOLUME_2D_ELEMENT_BOUNDARY_CONSTANT_SPECIFIED_VARIABLE_-VALUE	V2EBCS
VOLUME_2D_ELEMENT_BASIS	V2EB
VOLUME_2D_ELEMENT_BOUNDARY_LOCATION_POINT_VARIABLE_-VALUES	V2EBLP

Table B.1 - AIM short names of entities (continued)

Entity names	Short names
VOLUME_2D_ELEMENT_CONSTANT_SPECIFIED_VARIABLE_VALUE	V2ECSV
VOLUME_2D_ELEMENT_FIELD_INTEGRATION_RULE	V2EFIR
VOLUME_2D_ELEMENT_FIELD_VARIABLE_DEFINITION	V2EFVD
VOLUME_2D_ELEMENT_FIELD_INTEGRATION_EXPLICIT	V2EFIE
VOLUME_2D_ELEMENT_GROUP	V2EG
VOLUME_2D_ELEMENT_INTEGRATED_MATRIX_WITH_DEFINITION	V2EIMW
VOLUME_2D_ELEMENT_INTEGRATED_MATRIX	V2EIM
VOLUME_2D_ELEMENT_LOCATION_POINT_VARIABLE_VALUES	V2ELPV
VOLUME_2D_ELEMENT_NODAL_SPECIFIED_VARIABLE_VALUES	V2ENSV
VOLUME_2D_ELEMENT_VALUE_AND_LOCATION	V2EVAL
VOLUME_2D_NODE_FIELD_VARIABLE_DEFINITION	V2NFVD
VOLUME_2D_SUBSTRUCTURE_ELEMENT_REFERENCE	V2SER
VOLUME_2D_WHOLE_ELEMENT_VARIABLE_VALUE	V2WEVV
VOLUME_3D_ELEMENT_BASIS	V3EB
VOLUME_3D_ELEMENT_BOUNDARY_WHOLE_FACE_VARIABLE_VALUE	V3EBWF
VOLUME_3D_ELEMENT_BOUNDARY_EDGE_NODAL_SPECIFIED_-VARIABLE_VALUES	V3EBEN
VOLUME_3D_ELEMENT_BOUNDARY_EDGE_CONSTANT_SPECIFIED_-VOLUME_VARIABLE_VALUE	V3EBEC
VOLUME_3D_ELEMENT_BOUNDARY_CONSTANT_SPECIFIED_VARIABLE_-VALUE	V3EBCS
VOLUME_3D_ELEMENT_BOUNDARY_EDGE_WHOLE_EDGE_VARIABLE_-VALUE	V3EBEW
VOLUME_3D_ELEMENT_BOUNDARY_LOCATION_POINT_VARIABLE_-VALUES	V3EBLP
VOLUME_3D_ELEMENT_BOUNDARY_EDGE_LOCATION_POINT_-VOLUME_VARIABLE_VALUES	V3EBEL
VOLUME_3D_ELEMENT_BOUNDARY_NODAL_SPECIFIED_VARIABLE_-VALUES	V3EBNS
VOLUME_3D_ELEMENT_CONSTANT_SPECIFIED_VARIABLE_VALUE	V3ECSV
VOLUME_3D_ELEMENT_DESCRIPTOR	V3ED
VOLUME_3D_ELEMENT_FIELD_INTEGRATION_RULE	V3EFIR

Table B.1 - AIM short names of entities (concluded)

Entity names	Short names
VOLUME_3D_ELEMENT_FIELD_VARIABLE_DEFINITION	V3EFVD
VOLUME_3D_ELEMENT_GROUP	V3EG
VOLUME_3D_ELEMENT_INTEGRATED_MATRIX	V3EIM
VOLUME_3D_ELEMENT_INTEGRATED_MATRIX_WITH_DEFINITION	V3EIMW
VOLUME_3D_ELEMENT_LOCATION_POINT_VARIABLE_VALUES	V3ELPV
VOLUME_3D_ELEMENT_NODAL_SPECIFIED_VARIABLE_VALUES	V3ENSV
VOLUME_3D_ELEMENT_REPRESENTATION	V3ER
VOLUME_3D_ELEMENT_VALUE_AND_LOCATION	V3EVAL
VOLUME_3D_NODE_FIELD_VARIABLE_DEFINITION	V3NFVD
VOLUME_3D_SUBSTRUCTURE_ELEMENT_REFERENCE	V3SER
VOLUME_3D_WHOLE_ELEMENT_VARIABLE_VALUE	V3WEVV
VOLUME_ELEMENT_LOCATION	VLELLC
VOLUME_MEASURE_WITH_UNIT	VMWU
VOLUME_POSITION_WEIGHT	VLPSWG
VOLUME_UNIT	VLMUNT
WEEK_OF_YEAR_AND_DAY_DATE	WOYADD
WHOLE_MODEL_ANALYSIS_MESSAGE	WMAM
WHOLE_MODEL_MODES_AND_FREQUENCIES_ANALYSIS_MESSAGE	WMMAFA
WIRE_SHELL	WRSHL

Annex C
(normative)

Implementation method specific requirements

Conformance to this part of ISO 10303 shall be realized in one or more implementation methods. The implementation methods define what types of exchange behavior is required with respect to this part of ISO 10303. The requirements of ISO 10303-21 shall be supported by implementation of this part.

For an exchange structure, the file format shall be encoded according to the syntax and EXPRESS language mapping defined in ISO 10303-21 and 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 'structural_-analysis_design'.

Annex D
(normative)

Protocol Implementation Conformance Statement (PICS) proforma

Overview

The Protocol Information and Conformance Statement (PICS) Proforma supports the conformance assessment of implementations requesting evaluation to this part of ISO 10303. The PICS proforma contains a series of questions which capture static information about the implementation under test (IUT). This information is used in both statically assessing valid option selection, and in configuring an appropriate conformance test (dynamic assessment) session.

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:

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.

1. Product/system identifier (or name): ___________________________

There are ten classes defined in this international standard. Each class specifies a subset of ISO 10303-209 AIM constructs. These classes are detailed in 6.1.2 of this document. Conformance to this part of ISO 10303 requires conformance to at least one of the primary conformance classes 7 through 10.

2. Claimed classes of conformance (functionality) - circle choices:

   Class 1: Support for configuration control without shape information.

   Class 2: Support for Class 1 plus shapes represented by non_topological_surface_and_wireframe.

   Class 3: Support for Class 1 plus shapes represented by wireframe_with_topology.

   Class 4: Support for Class 1 plus shapes represented by manifold_surface_with_topology.

   Class 5: Support for Class 1 plus shapes represented by faceted_boundary_representation.

   Class 6: Support for Class 1 plus shapes represented by advanced_boundary_representation.

   Class 7: Support for part_identifiation, part_shape, material, part_composite_constituents, composite_constituent_representation, part_laminate_table, and zone_composite_constituents_-and_their_representation.

   Class 8: Support for part_identification, part_shape, material, fea_model, and analysis_report.

Class 9: Support for Class 8, plus fe_analysis_control.

Class 10: Support for Class 9, plus fe_analysis_results.

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.

3. Claimed implementation forms - circle choices:

   exchange structure (ISO 10303-21)

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.

4. Default Response: ________________________

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).

5. Does the IUT provide some user discretion over the changing and setting of static options?

   Yes or No

6. If yes, which ones?

   (a) Conformance class(es): ________________________

   (b) Default Response: ________________________

A statement of conformance shall include identification of at least one party deeming conformance for the implementation.

7. 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(209) version(0) }

is assigned to this part of ISO 10303. The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

E.2 Schema identification

To provide for unambiguous identification of the schema specifications given in this application protocol structural_analysis_design_schema in an open information system, object identifiers are assigned as follows:

{ iso standard 10303 part(209) version(1) object(1) structural-analysis-design-schema(1) }

is assigned to the structural_analysis_design schema expanded schema (see annex A);

{ iso standard 10303 part(209) version(1) object(1) structural-analysis-design-schema(2) }

is assigned to the structural_analysis_design schema short form schema (see 5.2).

The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

Annex F
(informative)

Application activity model

The application activity model (AAM) is provided 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. The application activity model is given in Figures F.4 through F.47. Activities and data flows that are out of scope are presented with dashed lines.

The viewpoint of the application activity model is that of the analyst who is responsible for performing a structural analysis of a composite structural part product. The analyst may transfer shape (geometric and topological), and material (composite and metallic) constituent and representation information from the design function to aid in the creation of analysis models, and may transfer this information back to the design function to provide recommended changes.

F.1 Application activity model background

The IDEF0 diagrams were developed from a composites structural part definition perspective. Homogenous material structural parts (for example metallic parts) are included in this AP. This is due to the fact that composite material structural part definitions typically have more extensive definition requirements than homogenous material structural part definitions, so therefore homogenous parts are considered a subset of composite parts. This perspective can be seen in many of the lower level IDEF0 diagrams where composites are split from homogenous material definitions.

The IDEF0 diagrams were developed from an aerospace manufacturer perspective to help to bound the AAM and the scope of the AAM activities. It is perceived that the activities found in the AAM for an aerospace manufacturer are more strict than most industrial design and analysis practices, and are therefore more encompassing than most commercial requirements.

This part of ISO 10303 is a member of a suite of application protocols to support the exchange of product data within the life cycle of a product. The information exchange problem has been decomposed from a high level perspective, and then several areas were selected that have a high payback potential for development. This AAM is representative of one of these areas and is a decomposition for the analysis function, and the transfer of detail design information between the design and analysis functions. There has been some generalization at the lower levels from the overall AAM to accommodate the aspect of homogenous material parts since they are considered a subset of the total realm. There has also been some generalization on the lower levels relative to the types of composite parts of interest. AP 209 is a member of the suite of APs in which common information exists. Figure F.1 shows this common information set as the overlap between the boxes. The unique information that each of the suite application protocols addresses is also defined within its respective boxes.

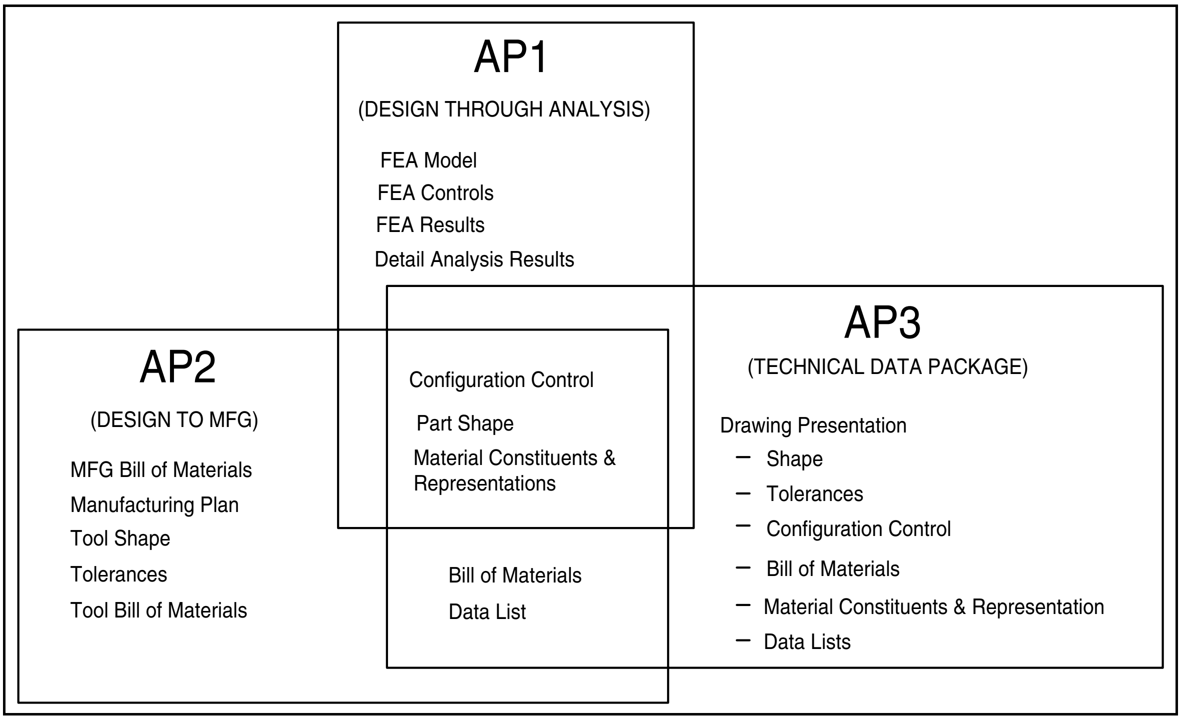

Figure F.1 - PAS-C Suite Concept

F.2 Information exchange scenario

Information exchange scenarios are illustrated in Figure F.2 and Figure F.3 to provide a usage perspective for the interpretation of the activity model. Throughout these scenarios it is assumed that the STEP implementation method could be either an exchange structure as specified in ISO 10303-21 or database. If a database application is used, then the translation task would use the implementation method specified in ISO 10303-22. Typical applications are delineated by dashed boxes surrounding a series of functional boxes to avoid presenting the impression of coupled applications and translators. A typical analysis application information exchange scenario as shown in Figure F.2 would proceed as follows:

— The part shape (points, curves, surfaces, faces, and shells), and material constituents and representation (laminate table, material property and specification, and the associated product constituent information and shape) are translated into a STEP implementation method from CAD and materials database applications. This information is then translated from a STEP implementation method into finite element analysis (FEA) preprocessor format. The discretized geometry (nodes) and elements are created in the FEA preprocessor based upon the imported information. The part shape (geometry and topology) and associated material constituent and representation information (such as for plies) could be modified to reflect the level of idealization necessary to model the structure with finite elements.

— The element thicknesses and material property attribute information are then automatically calculated with an application. The property attribute generation application could use element (discretized) geometry, part shape, and material constituent and representation information to calculate the element thicknesses and material properties. Metallic structures do not have associated composite materials constituents and representation information. The resulting element thickness and material attributes are then assigned to the elements.

— The loads, boundary conditions, and analysis controls are assigned using the FEA preprocessor application to the nodes and elements of the model as appropriate. The finite element model and controls (loads, boundary conditions, and analysis controls) information are then translated into a STEP implementation method.

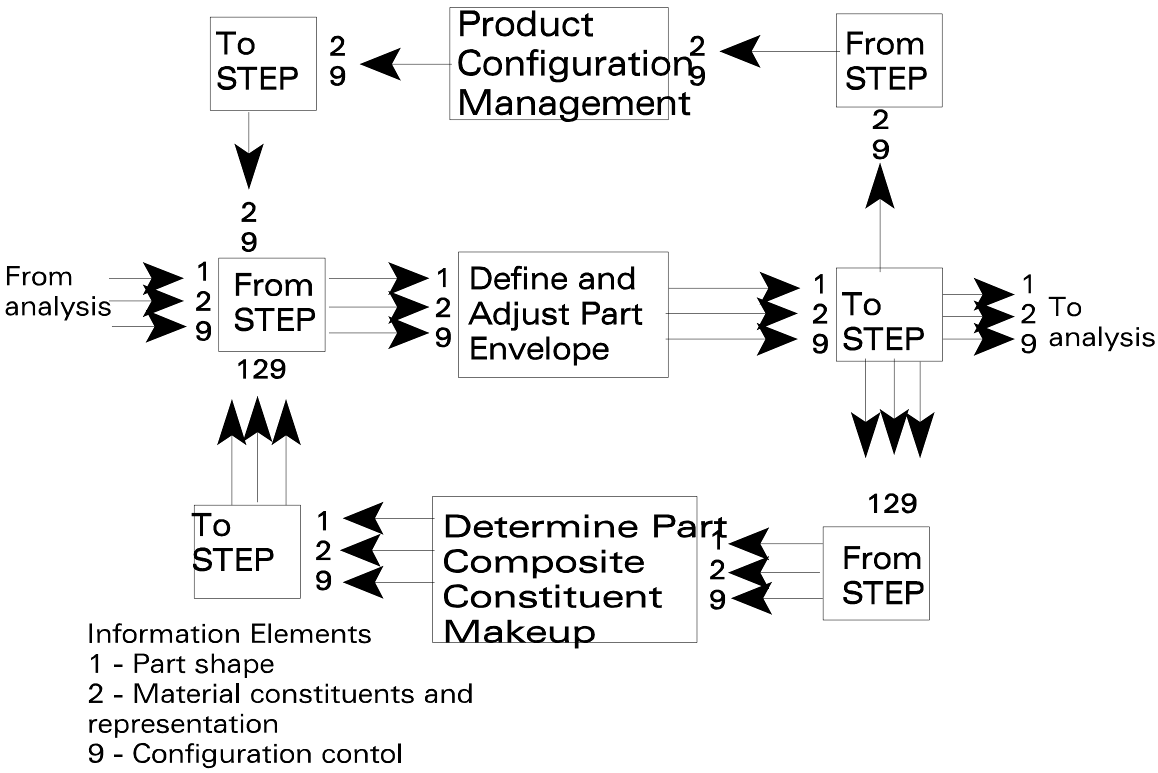

Figure F.2 - AP209 scenario from a structural designer's viewpoint

— The FEA model and controls information are then translated into the analysis application input format. The FEA (generation and solution of equations, output of resulting information) are performed using the analysis application. The analysis output information is combined with model and controls information, and then translated into a STEP implementation method.

— The FEA model, controls, and analysis output information are then translated from STEP implementation method into FEA postprocessor format. This information could be graphically and tabularly displayed using the capabilities of the FEA postprocessor. The FEA postprocessor display information is captured in a non-STEP implementation method as text tabular output, and simple 2D line and alphanumeric graphics. A reference to this postprocessor display information is then established in the related STEP implementation method. The tabular and graphical information does not have to be stored in an organized form such as a report. The postprocessor could also be used to calculate the critical load cases for further detail analysis by use of such criteria as minimum and maximum strains.

— Detail bolted joint and buckling structural analyses (among others) could also be performed based upon the same STEP input provided for the FEA postprocessor. Typically only the analysis output information from critical load cases would be analyzed. This information could then be graphically and tabularly displayed. Material constituent and representation information (for example ply tables and boundaries) could be translated to a STEP implementation method in preparation for another analysis iteration. The detail analysis output information is captured in a STEP implementation method in the same manner as the similar information from FEA postprocessing.

— The preprocessor application could then be used to update the part shape, material constituent and representation information, and element attributes to reflect modifications dictated by finite element and detail analyses. The altered part shape and material constituent and representation information are then translated to a STEP implementation method for incorporation by the design function.

— Once the design function has incorporated the recommended design changes, new finite element models are created, and finite element and detail analyses are redone to perform another check upon the updated design. The iterations would continue until the process converges to a satisfactory design, or time and/or money has run out. Configuration control information is exchanged throughout the scenario in order to track the history of the iterations and capture the approvals of each of the stages.

In practice one or more of the transfers of information, for example from FEA preprocessing to FE Analysis, would be made in native application format. This International Standard provides the flexibility of potentially integrating a totally heterogenous set of applications to perform the design through analysis task.

Some of the tasks from the design perspective of the information exchange scenario are summarized in Figure F.3. The (potential) modification of the inputs and output reflect the iterative nature of the design though analysis tasks. The activities in this diagram reflect the details of the interior of the CAD box in diagram Figure F.2.

The product configuration management application reflects the establishment of the initial configuration of the part and the updating of that information. This configuration control information includes part identification, assembly information, effective, work order and work request information. All or a portion of this information could be passed from the application into a STEP implementation method to form the outline that ties the other product information together.

The define and adjust part envelope application is where the initial part shape (base surfaces and edges of part) are defined. This information is passed from this application to a STEP implementation method and then to a FE Model Preprocessor application. The define and adjust part envelope application also adjusts or creates new part shape information based on inputs from the determine part composite constituent makeup application or the FE Model Preprocessor application.

The determine part composite constituent makeup application is where the details of the part and its constituent parts are created. Information such as stock size can come from a material data base. Information such as types, quantities, and location of material can be obtained from analysis applications. The output from the Determine Part Composite Constituent Makeup application are descriptions of such things as plies, core, filament laminates, and assembly information of constituents. This information is passed to a STEP implementation method and then passed to analysis applications for further evaluation.

F.3 Application activity model abbreviations

AMR advanced material requests

CDR critical design review

CSP composite structural part

I/F interface

M/D/B/S manage/design/build/support

M&P materials & processes

PASC PDES application protocol suite for composites

QA quality assurance

SP structural part

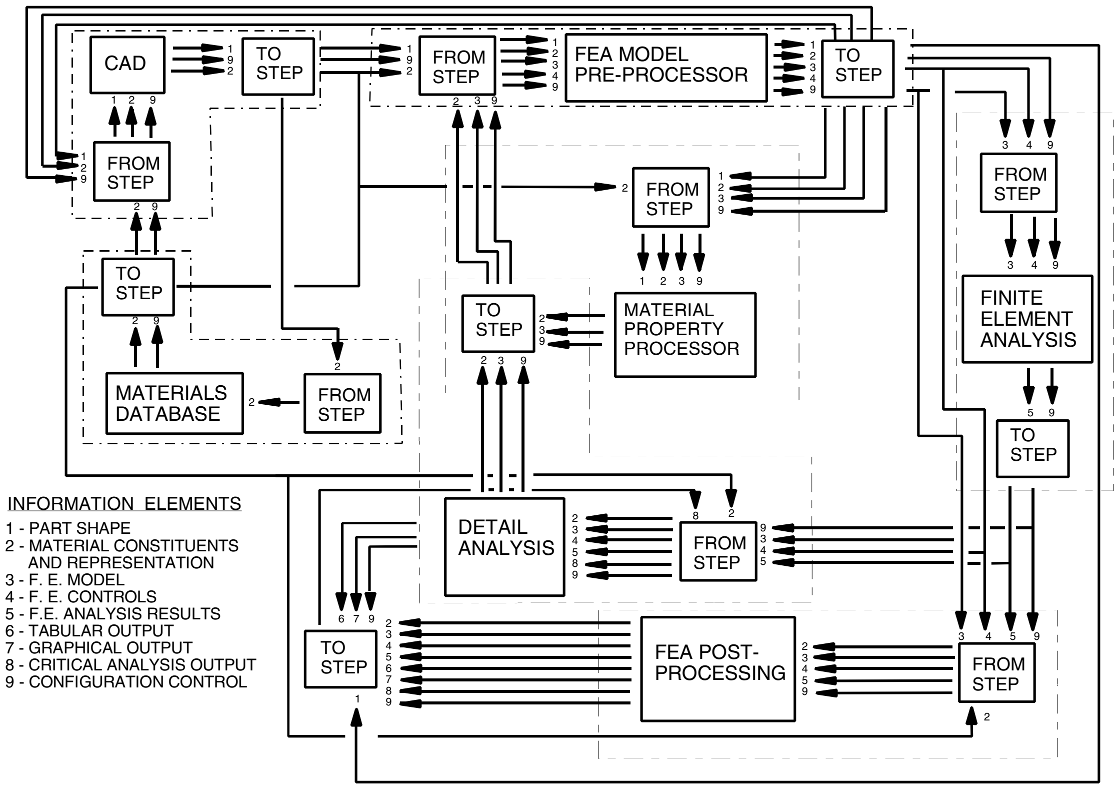

Figure F.3 - AP209 scenario from an analysis viewpoint

F.4 Application activity model definitions

The following terms are used in the application activity model. Terms marked with an asterik are outside the scope of this application protocol. The definitions given in this annex do not supersede the definitions givem in the main body of the text.

F.4.1
Analysis Decision Data
The data that records the decisions and idealizations made during the stress analysis of the structural part.

F.4.2
Analysis Materials Property Database
A database of all materials and processes (M&P) data required to perform structural analyses.

F.4.3
Analysis Output Database
A database of finite element (FE) analysis output data.

F.4.4
Analysis Staff and Tools
The analysis staff and the tools to analyze composite structure. The tools can include manuals such as handbooks and computerized tools such as finite element analysis programs and various detail analysis programs.

F.4.5
Analyze SP Manufacturing Discrepancies*
Inspect, gather analysis input data, research and apply analyses, and recommend and document the disposition of discrepant parts.

F.4.6
Anisotropic Material Properties
The two- or three- dimensional anisotropic elastic and thermal expansion property matrices that describe the response of a composite material.

F.4.7
Anomaly Reports
Reports analyzing the rejection of production parts for feedback to design and production planning.

F.4.8
Apply SP Loads/Boundary Conditions from Coarse Grid Model
Use data from the internal loads or stress database to provide applied loads and displacements for the fine grid analysis.

F.4.9
Approved SP Data
The functional and cross-functional approved structural part data that have been reviewed to be specific to a structural part. They include the technical performance constraints of the various functions along with the envelope features that are desired. The pertinent functions are from the manage, design, build and support activities.

F.4.10
Approved SP Requirements
The functional and cross-functional requirements that have been reviewed to be specific to a structural part. They include the technical performance constraints of the various functions along with the envelope features that are desired. The pertinent functions are from the manage, design, build, and support activities.

F.4.11
Assign SP Factors of Safety/Damage Tolerance Allowables
Assign acceptable factors of safety and durability/damage tolerance allowables for elements.

F.4.12
Assign SP Material, Geometric, Material Coordinate System/Angle Attributes to Elements
Assign the material, geometric, material coordinate system or angle attributes as appropriate to elements.

F.4.13
Assigned Allowables
The factor of safety and durability or damage tolerance allowables that have been assigned to an element.

F.4.14
Attach CSP Fiber Orientation
Attach the fiber orientation to the composite structural part ply table.

F.4.15
Attach CSP Material Flagnotes
Attach the various composite structural part material flagnotes to the composite structural part ply table.

F.4.16
Attach CSP Part Numbers
Attach the assigned composite structural part number to the composite structural part ply table.

F.4.17
Attach CSP Ply Callouts
Attach the composite structural part ply callouts to each ply in the composite structural part's laminate by assigning a part number to each of the plies.

F.4.18
Attach CSP Ply Numbers
Attach the assigned composite structural part ply number to the composite structural part ply table.

F.4.19
Attach CSP Revision Letter
Attach the appropriate revision letter, of the change status, to the composite structural part ply table.

F.4.20
Attach CSP Splice Flagnote
Attach the splice flagnotes to the composite structural part ply table.

F.4.21
Attach SP Dimensions & Tolerances*
Attach all the necessary dimensions and tolerances to the geometry of the drawing.

F.4.22
Attach SP Engineering Notes*
Attach all the structural part engineering notes on the drawing. They specify process specifications, change notes, material callouts, etc.

F.4.23
Boundary Constraints and Releases
The constraints and releases applied to the nodes of the finite element model to simulate the presence of connecting structure mountings or attachments.

F.4.24
Build and QA an SP*
The conversion of a design into a finished product and quality assurance functions that assure that the product meets requirements. This is usually a repetitive function, continuing throughout the product's life cycle. It receives the design from design functions and outputs the products, spare and repair parts, and technical data on each instance of the product.

F.4.25
Build SP Layouts & Models
Build the structural part layouts and models using the various geometry inputs.

F.4.26
Build SP Model & Drawing Tree*
A model or drawing tree is developed for the structural part, which specifies the combinations of items used to create the structural part.

F.4.27
Build SP Parts List*
Build a structural part parts list of the components that make up the structural part.

F.4.28
Calculate SP Margins of Safety Based upon Fine Grid Analysis Results
Use data from overall structural part and fine grid finite element analyses to assign margins of safety for structural details of the structural part.

F.4.29
Candidate Materials
The materials initially selected for the preliminary design and analysis of the structural part.

F.4.30
Coarse Grid Finite Element Analysis Model
The finite element model used for the overall structural part static stress analysis.

F.4.31
Coarse Grid Stiffness
The stiffness matrix (substructure or superelement) that represents the stiffness of the coarse grid model at attachment points to the fine grid model to supply proper flexible boundary conditions.

F.4.32
Collect & Layout CSP Core Geometry
Collect all the necessary geometry inputs necessary to layout the core in a core stiffened panel.

F.4.33
Collect Baseline SP Design Data*
The collections of baseline structural part design data includes the selected preliminary design, test data, producibility and maintainability studies.

F.4.34
Collect SP Existing Material Data*
Collect existing data needed to support baseline and trade analyses, and the definition of design criteria.

F.4.35
Collect, Review, Define & Distribute Structural Part Requirements*
Collecting, reviewing, defining and distributing structural, cross-functional engineering, build, quality assurance and logistic support requirements of the structural part.

F.4.36
Collected SP Baseline Data
The collected structural part baseline data consists of the required input geometry, system, and drawing data.

F.4.37
Collected SP Baseline Drawing Data
The collected structural part baseline drawing data consists of all the drawing data from the selected structural part preliminary design, test data and producibility and maintainability studies.

F.4.38
Collected SP Baseline Geometry
The collected structural part baseline geometry consists of all the selected structural part preliminary design geometry and other associated producibility and maintainability studies geometry.

F.4.39
Collected SP Baseline System Data
The collected baseline system data for structural part development are the standard features of the available CAD systems hardware and software configurations.

F.4.40
Company & Customer Review Standards
The structural part models and drawings are all reviewed in a manner that ensures that the model has satisfied certain company and customer standards for quality and completeness.

F.4.41
Conduct Beam Buckling and Crippling Analyses
Conduct stiffener buckling and crippling analyses to augment the finite element analyses of the structural part.

F.4.42
Conduct Beam Composite Cutout Analyses
Conduct cutout analyses to augment the finite element analyses of the structural part.

F.4.43
Conduct Beam Composite Fastener Pull-Through Analyses
Conduct fastener pull through analyses to augment the finite element analyses of the structural part.

F.4.44
Conduct Beam Composite Joint Analyses
Conduct joint analyses to augment the finite element analyses of the structural part.

F.4.45
Conduct Beam Composite Point Stress Analysis
Conduct point stress analyses to augment the finite element analyses of the structural part.

F.4.46
Conduct Beam Static Strength Analyses
Conduct various detail analyses such as joint and cutout analyses to augment the finite element analyses of the structural part.

F.4.47
Conduct Beam Stiffener Pull-off Analyses
Conduct stiffener pull-off analyses to augment the finite element analyses of the structural part.

F.4.48
Conduct Detail SP Analyses
Conduct the detail structural part structural analyses based on design loads from the weight, static, dynamic and thermal environments.

F.4.49
Conduct Panel Analyses
Conduct panel analyses such as buckling, crippling, transverse loadings, and panel flutter to augment the finite element analyses of the structural part.

F.4.50
Conduct Panel Composite & Metallic Cutout Analyses
Conduct cutout analyses to augment the finite element analyses of the structural part.

F.4.51
Conduct Panel Composite & Metallic Fastener Pull-Through Analyses
Conduct fastener pull- through analyses to augment the finite element analyses of the structural part.

F.4.52
Conduct Panel Composite & Metallic Joint Analyses
Conduct joint analyses to augment the finite element analyses of the structural part.

F.4.53
Conduct Panel Composite & Metallic Point Stress Analyses
Conduct point stress analyses to augment the finite element analyses of the structural part.

F.4.54
Conduct Panel Static Strength Analyses
Conduct various detail analyses such as joint and cutout analyses to augment the finite element analyses of the structural part.

F.4.55
Conduct SP Detail Stress Analysis
Conduct detail stress analysis of part details such as fasteners and cutouts using handbook and automated methods. The internal loads or stress database or hand generated loads are used to supply the input data for these analyses. These analyses are used to support drawing sign out, and final documentation.

F.4.56
Conduct SP Durability and Damage Tolerance Analysis*
Conduct durability and damage tolerance analysis to classify critical parts, guide material and allowables selection, and set non-destructive inspection criteria.

F.4.57
Conduct SP Dynamic Analysis*
Conduct analyses to evaluate the dynamic response of the structural part.

F.4.58
Conduct SP Fine Grid Finite Element Analysis
Conduct fine grid finite element analyses of details of the structural part that were not appropriate to include in the overall structural part (coarse grid) finite element analysis.

F.4.59
Conduct SP Finite Element Analysis
Conduct static stress analysis using finite element analysis techniques.

F.4.60
Conduct SP Mass Properties Analysis*
Conduct analyses to evaluate the total weight and mass distribution of the structural part.

F.4.61
Conduct SP Static Loads Analysis*
Conduct analyses to calculate all types of loading, such as aerodynamic, inertial, etc.

F.4.62
Conduct SP Static Strength Analyses
Conduct various detail analyses such as joint and cutout analyses to augment the finite element analyses of the structural part.

F.4.63
Conduct SP Static Stress Analysis
Stress analysis of structural part structures is a contractual requirement to ensure the integrity of the structure.

F.4.64
Conduct SP Thermal Analysis*
Conduct analyses to calculate thermal loads from such sources as aerodynamic heating and engine waste heat.

F.4.65
Conduct Structural Part Preliminary Design & Analysis
The preliminary design and analysis of various structural part concepts in order to trade performance, cost and producibility parameters for selecting an optimum structural part concept.

F.4.66
Conduct Structural Part Prototype Tests & Evaluation*
All the physical and electronically simulated tests and evaluation of structural part prototypes.

F.4.67
Configuration Data
The configuration of the structural part.

F.4.68
Create CSP Ply Tables
Create the composite structural part ply tables based on the sequence input and other pertinent design information.

F.4.69
Create Detail Structural Part Design
Create the detail structural part design based on inputs from the preliminary design phase and detail concurrent analyses.

F.4.70
Create SP 2D Envelope*
Create the structural part 2D envelope geometry using conventional 2D drawing entities within the selected construction planes.

F.4.71
Create SP 3D Wireframe*
Create the structural part 3D wireframe geometry using conventional 3D drawing entities.

F.4.72
Create SP Analysis Materials Database
Create the information structure for an analysis materials property database, and supporting software as necessary. Load the new and existing collected materials test data into the database.

F.4.73
Create SP Data
Create all the structural part design data necessary for detail drawings and associated engineering notes.

F.4.74
Create SP Drawing Data
Create all the structural part drawing data from the geometry and engineering specifications inputs using the selected systems.

F.4.75
Create SP Geometry Layouts & Models
Create all of the necessary structural part geometry layouts and models from the various inputs and prepare the data for transfer to other functions.

F.4.76
Create SP Node Geometry from Existing Geometry
Nodal geometry is created from the existing computer representation of the structural part. Computerized applications may be used to automate node generation.

F.4.77
Create SP Solid*
Create the structural part solid geometry using conventional or specialized solid entities.

F.4.78
Create SP Static Analysis Decision Record
Create a record of the decisions and idealizations made during the static stress analysis.

F.4.79
Create SP Surface*
Create the structural part surface geometry using conventional surface modelling entities.

F.4.80
Create SP Tooling Interface Drawings*
The creation of all the structural part inner mold line (IML) or outer mold line (OML) tool interfaces to the structural part. These drawings are also referred to as envelope drawings.

F.4.81
Create/Document SP Fine Grid Internal Loads/Stress Database Results
Create an internal loads and stress database by inputting data from the fine grid analysis.

F.4.82
Create/Document SP Internal Loads/Stress Database
Create and document an internal loads and stress database by inputting data from an existing solution or STEP format, and then documenting it with textual and graphical post-processing applications.

F.4.83
Cross-section Data
Data describing the extensional and beam bending behavior of a tee composite assembly.

F.4.84
Cross-sectional Area
Data describing the cross-sectional area of a tee composite assembly.

F.4.85
CSP Core Density
The composite structural part's core density as expressed in mass per unit volume.

F.4.86
CSP Core Details
The composite structural part’s core details consist of the core periphery, thickness, density, internal, transition and ribbon features.

F.4.87
CSP Core Edge Band Features
The composite structural part core edge band features are the core's edge geometry requirements based on fasteners, tolerances and the layup or placement in the manufacturing process.

F.4.88
CSP Core Filler Features
The composite structural part core filler features are the stabilizing materials as selected from either a foam, syntactic, potting compound or resin.

F.4.89
CSP Core Geometry
The composite structural part core geometry characteristics of the core as configured within the composite structural part envelope.

F.4.90
CSP Core Internal Fitting Features
The composite structural part core internal fitting features are those joining and mating requirements of other parts to the internal space of the core panel.

F.4.91
CSP Core Layouts & Models
The composite structural part core layouts and models consist of all the two dimensional and three dimensional geometry of the core. The composite structural part core models consist of the appropriate geometry required of the design.

F.4.92
CSP Core Material Features
The selected composite structural part core material features such as the material name and its associated processing specifications.

F.4.93
CSP Core Periphery
The composite structural part core periphery is the edge and internal envelope as constrained by the structural or nonstructural interfaces.

F.4.94
CSP Core Ribbon Features
The designed core ribbon direction as necessary to take the bending or axial load of the composite structural part.

F.4.95
CSP Core Thickness, Density and Material Features
The composite structural part's core thickness, density and material features.

F.4.96
CSP Core Thickness
The composite structural part's core thickness as designed to meet the various requirements.

F.4.97
CSP Core Transition Features
The composite structural part designed core top and bottom ramp radius and angle.

F.4.98
CSP Core Views
The composite structural part core views are the various selected views necessary to show the desired features of the composite core.

F.4.99
CSP Details Layouts & Models
The composite structural part layouts and models consist of all the two dimensional and three dimensional geometry required of the composite structural part details.

F.4.100
CSP Details Views
The composite structural part details views are the various selected views necessary to show the desired features of the composite item details. These include the typical front, top and side views.

F.4.101
CSP Details
The details of the composite structural part details as designed to meet the design requirements.

F.4.102
CSP Filler & Cap Periphery
The composite structural part filler and cap periphery which is dictated by core, mating part, and tooling requirements.

F.4.103
CSP Filler, Cap, & Core Periphery
The composite structural part filler, cap, and core periphery consist of the filler, cap and core periphery, thickness, density, internal, transition and ribbon features.

F.4.104
CSP Manufacturing Process Constraints
The specific composite structural part design constraints as a result of manufacturing issues which involve the layup, tooling and inspection issues.

F.4.105
CSP Periphery
The composite structural part periphery based on tooling constraints in the basic areas, and at the edges, and the geometrical interfaces to other mating parts.

F.4.106
CSP Ply Counts
The composite structural part ply counts are the desired ply counts based on the target layup orientations and the target thickness.

F.4.107
CSP Ply Periphery with Callouts
The composite structural part ply periphery as dictated by tooling and mating part constraints with unique identifying ply numbers.

F.4.108
CSP Ply Periphery
The composite structural part ply periphery as dictated by tooling and mating part constraints.

F.4.109
CSP Ply Sequence
The ply sequence of the composite structural part which shows the ply sequence in the laminate layer in the order of buildup from an inner mold line (IML) or outer mold line (OML) tool.

F.4.110
CSP Ply Tables
The composite structural part ply table is the combination of the laminates part and ply numbers, material, fiber orientations, splices and changes.

F.4.111
CSP Target Layup Orientation
The composite structural part target layup orientation is based on the laminate design rules for percent ply orientation in the subject area.

F.4.112
CSP Target Thickness
The composite structural part target thickness is based on interfaces to other parts, tooling constraints and strength considerations.

F.4.113
Define SP Material Development Program*
Define a material development and coupon test program to collect the materials data.

F.4.114
Design and Analyze an SP
The complete design and analysis life cycle from the pre-proposal phase to product support in the field.

F.4.115
Design CSP Core Ribbon Direction
Design the composite structural part's core ribbon direction to take advantage of the load paths due to bending or axial loads.

F.4.116
Design CSP Core Thickness, Density & Material
Design the composite structural part core thickness, density and material to meet the weight, stress loads and minimum manufacturing constraints.

F.4.117
Design CSP Core Thickness
Design the composite structural part core thickness based on the performance loads, manufacturing constraints and the other structural or nonstructural interfaces.

F.4.118
Design CSP Core Transition Area
Design the composite structural part core transition area to resolve the core top and bottom ramp radius along with the ramp angle.

F.4.119
Design Geometry
The three-dimensional point, curve, surface and volume information that describes the geometric representation of the structural part.

F.4.120
Detail Design Requirements Changes
The detail design requirements changes based on the analysis, producibility and maintainability results.

F.4.121
Detail SP Analyses
All analysis output data from finite element, detail, and durability/damage tolerance analyses, and the documentation of the decisions taken during those analyses.

F.4.122
Detail SP Design Data
The detail structural part design data consists of all the models, drawings and parts list that make up the structural part.

F.4.123
Detail SP Item Drawings
The structural part detail item drawings are all the design details of the subcomponents and assembly of the structural part.

F.4.124
Detail SP Test Requirements
The detail structural part test requirements based on the preliminary detail and analysis of the structural part.

F.4.125
Determine CSP Ply Counts*
Determine the composite structural part ply counts based on the guidelines established due to the target layup orientations and target thickness.

F.4.126
Develop & Provide SPs Materials
The material suppliers process of creating stock material for composite and homogenous material manufacturers. Basic material properties and allowables are addressed here.

F.4.127
Develop CSP Core Periphery
Develop the composite structural part core periphery based on its edge band and interface to structural or nonstructural parts.

F.4.128
Develop CSP Ply Periphery
Develop the composite structural part ply periphery based on tooling constraints in the basic areas and at the edges, and the geometrical interface to other mating parts.

F.4.129
Develop Structural Part Needs & Procurement*
The customer analysis of the structural part needs based on the customer's needs and the state of structural part technologies, along with the procurement process throughout the life cycle as managed by the customer.

F.4.130
Develop, Procure, Build, Use & Maintain a Structural Part
The activity covering the entire life cycle of a structural part as viewed from the combined activity groupings of the customer needs analysis and procurement, aerospace contractors, the customer's use and maintenance, and the raw material suppliers.

F.4.131
Dimensioned & Noted SP Details
The dimensioned, toleranced and noted structural part details as they would appear on a completed detail drawing.

F.4.132
Dimensioned SP Details
The dimensioned and toleranced structural part details as they would appear on the drawing.

F.4.133
Displacements
The displacements of the nodes of the finite element model that result from a finite element analysis.

F.4.134
Drawing Preparation Standards
The structural part models and administrative data are all presented in a standard format on a drawing.

F.4.135
Durability & Damage Tolerance Analysis Allowables
The detail structural part analysis is dealing with durability of the structural part and the damage tolerance of the structural part.

F.4.136
Element Attributes
The geometric and material attributes necessary to describe the various finite element continuum idealizations. For example, the curve elements require cross-section, offset, and material data for their respective attributes. The surface elements require thickness, offset and material data. The volume elements require only material data.

F.4.137
Element Connectivity
The references to the nodes that the element is connected to, and any necessary identifiers for the element itself.

F.4.138
Evaluate Detail SP Analyses*
Evaluate all the detail structural part analyses done to substantiate the weight, static, dynamic and thermal analysis; damage tolerance and producibility.

F.4.139
Feed Back SP Laminate Ply Stacking, Distribution, and Orientation to Design
Feed back any changed laminate descriptions, ply stacking sequence and orientations to design.

F.4.140
Field and Maintenance Changes & Revision History
The reviewed field/maintenance changes that result from field use of the structural part and the history of these changes in the field.

F.4.141
Field/Maintenance Changes
The reviewed field/maintenance changes that result from field use of the structural part.

F.4.142
Fine Grid Analysis Output Data
All analysis output data from the fine grid finite element analysis, and the documentation of the decisions taken during those analyses.

F.4.143
Fine Grid Finite Element Analysis Controls
The boundary constraints and releases, load sets and combinations, allowables, output requests and analysis procedure controls that are combined with a fine grid finite element model to provide input to a finite element analysis (FEA).

F.4.144
Fine Grid Finite Element Analysis Model
A finite element model that is based upon a coarse grid model that provides additional mesh refinement in a particular area of interest. The finer mesh provides greater analysis accuracy (in the fine grid area) than is otherwise feasible with a large scale coarse grid finite element model.

F.4.145
Finite Element Analysis Controls
The boundary constraints and releases, load sets and combinations, allowables, output requests and analysis procedure controls that are combined with a finite element model to provide input to a finite element analysis (FEA).

F.4.146
Finite Element Analysis Model Material Properties
The two- or three- dimensional anisotropic elastic and thermal expansion properties that describe the response of a composite material.

F.4.147
Finite Element Analysis Model
The nodes, elements, element properties, material properties and associated administrative data that are combined with the finite element analysis (FEA) controls to form input to a finite element analysis.

F.4.148
Finite Element Analysis Output Data
All analysis output data from finite element analysis, and the documentation of the decisions taken during those analyses.

F.4.149
Forces and Moments
The applied and resulting forces and moments at each of the nodes of the finite element model that are produced by the finite element analysis.

F.4.150
Generate and Assign SP Analysis Output Control Requests
Generate and assign output control requests for each of the types of data required to be output.

F.4.151
Generate and Assign SP Element Attributes
Generate and assign element geometrical, material and ply related attributes.

F.4.152
Generate and Assign SP Element Connectivities
Connect element to corner, mid-edge, mid-face and mid-volume nodes to approximate the continuum of the structural part.

F.4.153
Generate and assign SP Load Sets and Combinations
Generate and assign nodal and elemental loadings that approximate the forces, temperatures or displacements acting on the structural part, and request the combination of load sets to approximate complicated loading conditions from simpler loading components.

F.4.154
Generate Beam Equivalent Cross-sectional Area
Generate equivalent cross-sectional area of the stiffener for curve elements.

F.4.155
Generate Beam Equivalent Cross-sectional Properties
Generate equivalent cross-sectional beam properties of the stiffener for curve elements.

F.4.156
Generate Beam Equivalent Thicknesses
Generate equivalent thicknesses for surface elements used to explicitly model the stiffener.

F.4.157
Generate Beam Geometric Attributes
Generate element geometric attributes such as thicknesses for surface elements, and areas and cross-section dimensions for curve elements.

F.4.158
Generate Contoured Panel Equivalent Thicknesses
Generate equivalent thicknesses (smearing core and face sheets) for surface elements.

F.4.159
Generate Contoured Panel Geometric Attributes
Generate element geometric attributes such as thicknesses for surface elements.

F.4.160
Generate Contoured Panel Shell Offsets
Generate shell offsets for surface elements to model off thickness centroid attachment.

F.4.161
Generate Contoured Panel Solid Element Properties
Generate the equivalent core properties for solid elements.

F.4.162
Generate Core Stiffened Panel Equivalent Thicknesses
Generate equivalent thicknesses (smearing core and face sheets) for surface elements.

F.4.163
Generate Core Stiffened Panel Geometric Attributes
Generate element geometric attributes such as thicknesses for surface elements.

F.4.164
Generate Core Stiffened Panel Shear Panel Core Area Equivalents
Generate shear panel core area equivalents for surface elements.

F.4.165
Generate Core Stiffened Panel Shell Offsets
Generate shell offsets for surface elements to model off thickness centroid attachment.

F.4.166
Generate Core Stiffened Panel Solid Element Core Equivalent Properties
Generate the equivalent core properties for solid elements.

F.4.167
Generate SP Analysis Procedure Controls
Generate the necessary directives to control the analysis process in the intended analysis code.

F.4.168
Generate SP Fine Grid Finite Element Model from Coarse Grid Model
Use the existing structural part finite element model to provide a geometric basis for generating a finer grid mesh to provide more deflection and strain resolution.

F.4.169
Generate SP Finite Element Analysis Environment and Controls
Generate, set, and assign analysis environment data such as boundary constraints, loads, factors of safety, and set up the control of analysis output and the analysis procedure itself.

F.4.170
Generate SP Finite Element Models
Generate a discrete geometric approximation of the structural part. Generate and assign elemental connectivity, geometric and material attributes. Set boundary conditions and generate and assign the loading environment. Generate the directives necessary to control the analyses and resulting output.

F.4.171
Generate SP Geometric Attributes
Generate element geometric attributes such as thicknesses for surface elements, and areas and cross-section dimensions for curve elements.

F.4.172
Generate SP Graphical Analysis Output Database Documentation
Generate graphical documentation of the internal loads/stress database such as color fringe plots of strain distributions over a skin.

F.4.173
Generate SP Graphical Finite Element Model Documentation
Generate the graphical documentation of the nodes and elements, and their associated attributes.

F.4.174
Generate SP Material Orientation Angles or Coordinate Systems
Generate material orientation angles by relating elements to coordinate systems, or by individual calculations. Alternatively, a material direction may be assigned to a coordinate system reference.

F.4.175
Generate SP Material Properties from Ply Properties, Stacking Sequence and Orientations
Generate material properties from ply properties, stacking sequence and orientations.

F.4.176
Generate SP Node Geometry
Discretize the surface or volume of the structural part by creating point geometry identical or related to the structural part geometry. Placement of the nodes on or within the structural part is governed by the fineness of the mesh needed to adequately discretize the deflection and strain fields of the structural part under the applied loading environment.

F.4.177
Generate SP Textual Analysis Output Database Documentation
Generate textual documentation of the internal loads/stress database such as minimum or maximum margin of safety distributions for skin elements, or a force free body of a stiffener.

F.4.178
Generate/Collect/Reduce SP Material Test Data*
Perform a development and coupon test program to collect the materials data and reduce data to a form usable for analysis.

F.4.179
Generate/Import SP Material Properties
Either generate, import or retrieve from a database of material properties.

F.4.180
Graphical Documentation
Graphical documentation of the analysis output data from a finite element analysis.

F.4.181
Graphical Finite Element Model Documentation
Graphical documentation of the nodes and elements, and their associated attributes.

F.4.182
Hand Generate SP Node Geometry
Generate node geometry by measuring parts, scaling drawings, or freehand, and hand input the nodal coordinate data into a computer file.

F.4.183
Homogenous Material Details
The homogenous material details consist of part peripheries that affect the mating parts.

F.4.184
Homogenous Material SP Details
The homogenous material structural part details consist of the design data for the periphery, thickness, density, internal, and transition features.

F.4.185
Homogenous SP Layouts & Models
The homogenous structural part layouts consist of all the two dimensional geometry required of the design. The homogenous structural part models consist of the appropriate geometry required of the design.

F.4.186
Homogenous SP Views
The homogenous structural part views are the various selected views necessary to show the desired features of the part. These include the typical front, top and side views.

F.4.187
Import SP Material Properties from Analysis Materials Database
Import material properties from an analysis materials property database, and retrieve the necessary data.

F.4.188
Import SP Material Properties from STEP Format
Import material properties from STEP format, and retrieve the necessary data.

F.4.189
Input Beam Anisotropic Material Property Matrices
Input material property matrices data.

F.4.190
Input Beam Cross-sectional Anisotropic Material Property Matrices
Input the anisotropic cross-sectional beam properties matrices data.

F.4.191
Input Beam Equivalent Cross-sectional Area Modulus of Elasticity
Input the equivalent modulus of elasticity appropriate for idealizing the stiffener as only a curve element with extensional stiffness.

F.4.192
Input Beam Shell Element Anisotropic Material Property Matrices
Input shell element (when the stiffener walls are explicitly modelled with surface elements) material property matrices data.

F.4.193
Input Beam Solid Element Anisotropic Material Property Matrices
Input volume element material property matrices data.

F.4.194
Input Contoured Panel Anisotropic Material Property Matrices
Input material property matrices data.

F.4.195
Input Contoured Panel Shell Element Anisotropic Material Property Matrices
Input material property matrices data appropriate for surface elements.

F.4.196
Input Contoured Panel Solid Element Anisotropic Material Property Matrices
Input material property matrices data appropriate for volume elements.

F.4.197
Input Core Stiffened Panel Anisotropic Material Property Matrices
Input material property matrices data.

F.4.198
Input Core Stiffened Panel Core Anisotropic Material Property Matrices
Input core material property matrices data.

F.4.199
Input Core Stiffened Panel Combined Face Sheet (Contoured Panel) and Core Anisotropic Material Property Matrices
Input face sheet and core (smeared together) material property matrices data.

F.4.200
Input Core Stiffened Panel Face Sheet (Contoured Panel) Anisotropic Material Property Matrices
Input face sheet material property matrices data.

F.4.201
Input SP Anisotropic Material Property Matrices
Input material property matrices data.

F.4.202
Input SP Geometry from STEP Format
Import geometry from STEP format into a finite element mesh creation and editing program. Nodal geometry is then created from the computer representation of the structural part. Computer applications may be used to automate node generation.

F.4.203
Integrate & Prepare SP Assembly Drawings
Integrate and prepare all of the items that make up the structural part into an integrated assembly drawing.

F.4.204
Load Sets and Combinations
A load set or combination of load sets provides a complete set of loads data. There may be one or more load sets or combinations of load sets in a given finite element analysis.

F.4.205
Loads Data
The applied forces, moments, displacements and rotations that are applied to a finite element model in combination with boundary constraints and releases to form input to a finite element analysis.

F.4.206
Manage an SP Development*
Managing all the resources specific to the structural part through the design, build and support functions. This includes people, budgets, tools, materials, etc.

F.4.207
Manage Configuration of Structural Part Data*
The configuration management of all the data produced in the design development of the structural part.

F.4.208
Manage, Design, Build and Support an SP
All the contracted management of resources, design, build, and support of a typical structural part, as done at the prime contracting company.

F.4.209
Margins of Safety
A ratio of actual to allowable stress or strain minus one. A margin of safety greater than zero means that the value of stress or strain will meet the design criteria.

F.4.210
Mass Properties Data
The mass data of the structural part. This data contains both overall structural mass data and lumped mass data to represent non-structural mass, discretized at the nodes of the finite element model.

F.4.211
Material Angles and Coordinate Systems
Either the angle that the material 11 (x) direction makes with the element coordinate system, or a reference directly to a coordinate system that defines the material 11 (x), 22 (y) and 33 (z) directions.

F.4.212
Material Property Data
All of the data needed to describe the physical responses of a composite material or its plies.

F.4.213
Material Specs & Requirements
Definition of material composition, form and performance requirements along with orders procuring material.

F.4.214
Model & Drawing Preparation Standards
The structural part models and drawings are all prepared to a minimum standard for the preparation of the geometric models, drawings and associated engineering notes for the structural part.

F.4.215
Model Preparation Standards
The structural part models are all prepared to a minimum standard for the preparation of the geometric models and associated engineering notes for the structural part.

F.4.216
Native Finite Element Analysis Controls Format
The finite element analysis (FEA) controls in a format suitable for the analysis application.

F.4.217
Native Finite Element Analysis Model Format
The finite element analysis (FEA) model in a format suitable for the analysis application.

F.4.218
Native Finite Element Analysis Output Format
The analysis output data from a finite element analysis in the native form of the analysis application.

F.4.219
Node Geometry
The geometric position data for the node, and any necessary identifiers.

F.4.220
Obtain Detail Structural Part Material Processes & Allowables Data*
Obtain the detail structural part material processes data that is unique to the composite materials and the data that is unique to homogenous materials. Obtain the mechanical allowables of representative sections of the structural part.

F.4.221
Offsets
Any offsets needed to describe the attachment of an element to a node that is not on the axis of the element coordinate system (for curve elements) or on the plane of the element coordinate system (for surface elements) of the element.

F.4.222
Output Control Requests
Requests for the finite element analysis code to selectively output the various types of analysis output data.

F.4.223
Perform Cross-Functional SP Reviews & Customer Design Review Functions*
Perform the necessary cross-functional and customer design reviews to support the critical design review phase.

F.4.224
Perform Linear Static Mechanical and Thermo-Mechanical Finite Element Analysis
Perform the analysis.

F.4.225
Perform SP Finite Element Analysis
Perform linear or nonlinear finite element analyses of the structural part.

F.4.226
Perform SP Mechanical and Thermo-mechanical Finite Element Analysis
Perform linear or nonlinear mechanical or thermo-mechanical analyses of the structural part by submitting the completed finite element model for analysis by the appropriate finite element analysis application.

F.4.227
Perform Structural Part Detail Design & Analysis
Testing the selected preliminary structural part design concept and developing it in sufficient detail to meet the desired performance, cost and production goals.

F.4.228
Perform Structural Part Product Design & Analysis Support*
Supporting all the design and analysis needed to resolve the changes encountered in manufacturing or those from field use of the structural part.

F.4.229
Plan SP Tests and Analyze Results*
Plan and analyze the output from element and sub-component structural test of the structural part to validate analyses.

F.4.230
Ply Boundaries
The location of the outer contiguous boundary of a ply.

F.4.231
Ply Orientations
The orientations of the plies in an structural part.

F.4.232
Ply Properties
The material properties of the ply.

F.4.233
Pre-Transferred SP Layouts & Models
The structural part layouts and models that have not been prepared for transfer to another media.

F.4.234
Preliminary Design Requirements Changes
The preliminary design requirements changes based on the analysis, producibility and maintainability results.

F.4.235
Preliminary Functional Requirements
The preliminary functional structural part requirements.

F.4.236
Preliminary Material Property Data
The preliminary structural part material property data as developed in the preliminary design phase.

F.4.237
Preliminary SP Test Requirements
The preliminary structural part test requirements based on the preliminary design and analysis of the structural part.

F.4.238
Prepare & Coordinate Signature Process
All of the responsible reviews of the drawings, as noted on the signature block, are coordinated for their specific functional reviews and signature.

F.4.239
Prepare & Release SP Advanced Material Requests*
All of the advanced material requests (AMRs) needed by the engineering function are prepared and released so the material necessary for the build cycle will be available for use when required.

F.4.240
Prepare CSP Core Details
The preparation of all the composite structural part core details to meet core periphery, thickness, density, material, transition and ribbon feature requirements.

F.4.241
Prepare Detail SP Item Drawings
The preparation of the detail item's drawings that makeup the structural part.

F.4.242
Prepare SP Composite Details
The preparation of all the structural part composite details to meet the periphery, thickness, layup orientation and ply stackup requirements.

F.4.243
Prepare SP Data for Transfer
Prepare the structural part data for transfer to other functions in either paper, translated or native form.

F.4.244
Prepare SP Details
Prepare the structural part details to meet the interface, joint, size and the other detail part drawing requirements.

F.4.245
Prepare SP Homogenous Material Details
The preparation of all the homogenous structural part details to resolve the periphery, thickness, density, material, and transition features.

F.4.246
Prepare SP Installation Drawings*
All of the subassemblies or assemblies that the structural part is used on are shown on each specific installation drawing.

F.4.247
Prepare SP Models & Drawings
Prepare the structural part models and drawings using the reviewed design inputs and creating the necessary outputs for other functional use.

F.4.248
Pre-release SP Design Data
The pre-released production support design and analysis data of the structural part.

F.4.249
Pre-release SP Detail Data
The pre-released detail structural part design data consists of all the models, drawings and parts list that make up the structural part.

F.4.250
Pre-release SP Preliminary Data
The pre-released structural part preliminary design and analysis data.

F.4.251
Procured SP Information
The information that is procured with the procured parts. This would include the product definition data, design data, etc.

F.4.252
Procured SP
The actual procured structural part.

F.4.253
Procured Structural Parts & Information
The as-built structural part and information as purchased from the outside associate or subcontractors. It includes not only the procured parts, but the respective drawings, material information such as stock, material properties, etc.

F.4.254
Produce CSP Ply Stackup
Produce the composite structural part ply stackup which shows the ply sequence, ply tables and the specific ply periphery details.

F.4.255
Product Development Standards
The quality and product standards for preparation and presentation of the structural part.

F.4.256
Receive & Review Native CAD Data
Receive and review the native CAD data as received from similar CAD systems

F.4.257
Receive & Review Paper Geometry Data
Receive and review all the paper geometry data necessary to develop structural part geometry.

F.4.258
Receive & Review SP Geometry Data
Receive and review all the different forms (paper, translated, native) of structural part geometry data that will be necessary to develop the structural part geometry.

F.4.259
Receive & Verify CAD Translated Data
Receive and verify the translated CAD data as delivered from other CAD systems

F.4.260
Refined Detail SP Analyses
All analysis output data from fine grid finite element analyses, and the documentation of the decisions taken during those analyses.

F.4.261
Resolve CSP Core Density*
Resolve the core density to meet the weight and stress loads along with meeting the minimum manufacturing constraints.

F.4.262
Resolve CSP Core Edge Band Issues*
Resolve the composite structural part core edge band to meet the edge margin of fasteners, clearance for the core layup or placement process and the necessary dimensional tolerances of the core like surface flatness.

F.4.263
Resolve CSP Core Fillers*
Resolve the various stabilizing core filler based on the design requirements. Filler material types include foams, syntactic, potting compound and resins.

F.4.264
Resolve CSP Core Internal Fittings*
Resolve the interface details at the core periphery based on fittings and mating subassemblies.

F.4.265
Resolve CSP Core Material Features*
Resolve the core material features as far as the structural allowables, material compatibility and manufacturing process constraints are concerned.

F.4.266
Resolve CSP Manufacturing Process*
Resolve the composite structural part manufacturing issues which involve the layout, tooling and inspection issues.

F.4.267
Resolve CSP Periphery*
Resolve the composite structural part periphery due to tailoring, part interfaces and periphery parameters.

F.4.268
Resolve CSP Ply Sequence*
Resolve the composite structural part ply sequences by showing the laminate layer in the order of buildup from an inner mold line (IML) or outer mold line (OML) tool.

F.4.269
Resolve CSP Target Layup Orientation*
Resolve the composite structural part target layup orientation of the plies based on the design rules established for percentage ply orientation in the subject area.

F.4.270
Resolve CSP Target Thickness*
Resolve the composite structural part target thickness based on interfaces to other parts, tooling constraints and strength considerations.

F.4.271
Resolve SP Interfaces & Joints*
Resolve all the mating interfaces to the structural part that involve mechanical or bonded joints. Look at space constraints, attachment issues and material compatibility.

F.4.272
Resolve SP Size and Design Issues*
Resolve all the size issues regarding the structural part size due to tooling constraints and general design rules regarding the length and width features.

F.4.273
Reviewed 2D Baseline Geometry
The reviewed 2D baseline geometry is the data received from other sources that is deemed necessary to support the geometry development by the structural part.

F.4.274
Reviewed 3D Baseline Geometry
The reviewed 3D baseline geometry is the data received from other sources that is deemed necessary to support the geometry development by the structural part.

F.4.275
Reviewed SP Geometry Data
The reviewed structural part geometry data is the geometry that is deemed necessary to support the geometry development of the structural part.

F.4.276
Reviewed SP Native CAD Data
The reviewed structural part native CAD data is the geometry and test that was reviewed after received from a similar CAD system.

F.4.277
Reviewed SP Paper Geometry Data
The reviewed structural part paper geometry data is the data necessary to develop the structural part geometry.

F.4.278
Screen SP Available Materials*
Use cost and structural performance criteria to screen available materials.

F.4.279
Select & Detail SP Views
Select and detail the necessary structural part views based on the typical top, front, side and cross-sections needed to show the desired features.

F.4.280
Select & Prepare Model/Draft System*
Select and prepare the modelling or drafting geometry system to be used for the detail design phase of the structural part.

F.4.281
Select SP Composite or Homogeneous Material*
Use weight, cost and structural performance criteria to select a composite or homogeneous material.

F.4.282
Select SP Construction Planes*
Select the structural part construction planes that render the desired views of the structural part for top, front, side or cross-section details.

F.4.283
Selected Drawing System
The selected drawing system needed to support the detail design development.

F.4.284
Selected Mod/Drawing System
The selected modelling system and drawing system needed to support the detail design development.

F.4.285
Selected Modelling System
The selected modelling system needed to support the detail design development.

F.4.286
Selected SP Construction Planes
The selected construction planes are the associated cartesian axes of the top, front, side or cross-section views desired of the structural part.

F.4.287
Selected SP Preliminary Design
The structural part preliminary design selected from the various concepts that were traded, is now ready for the detail design phase.

F.4.288
Selected SP Views
The selected structural part views necessary to show the geometric features are the top, front, side and cross-section views.

F.4.289
Selected System Constraints
The selected system constraints are the inherent drafting or modelling technique constraints of the design toolset.

F.4.290
Selected View Constraints
The selected view constraints are the construction planes of the top, front, side or cross-sections that are location dependent.

F.4.291
Set and Assign SP Boundary Constraints and Releases
Set and assign boundary constraints and releases that approximate the support or symmetry boundary conditions for the analysis of the structural part.

F.4.292
Solid Element Properties
The properties necessary to describe the structural response of a volume element.

F.4.293
SP 2D Envelope
The structural part 2D envelope consists of just the 2D view of the structural part.

F.4.294
SP 3D Wireframe
The structural part 3D wireframe consists of a 3D view of the structural part features using conventional 3D modelling entities that show the edges.

F.4.295
SP Advanced Material Requests
The structural part advanced material requests (AMRs) are the advanced material requests by engineering of structural part components that will be available to use for the build cycle.

F.4.296
SP Analysis Constraints
The detail analysis done on the structural part has determined strength limits for the structural part.

F.4.297
SP Analysis Drawing Changes
The changes that are recommended due to the analysis is reflected as redline marks to the design drawings.

F.4.298
SP As-Built Definition Package
The as-built configuration management data of the structural part.

F.4.299
SP Assembly Drawings
Structural part assembly drawings show how the detail component items are positioned in an assembly.

F.4.300
SP Budget & Schedule
The structural part budget and schedule

F.4.301
SP Build Changes
Structural part build changes are those that are occurring during the production process of the structural part.

F.4.302
SP Business Management Data
All of the budget, cost, schedule and manpower data for the structural part development. This also includes information such as contract number, design activity, etc.

F.4.303
SP Contract
The contract for the structural part development as received from the customer.

F.4.304
SP Design Data
Structural part design data consists of all the configuration managed design and analysis information that occurs for preliminary, detail and production support phases of the structural part.

F.4.305
SP Design History
The structural part design history consists of all the similar design activities that have created data that is similar to the structural part.

F.4.306
SP Details
Structural part details consists of all the design data required of the structural part as a result of mating interfaces and joints, internal design, and manufacturing considerations. This takes the form of drawings and parts lists that are distributed to other user functions.

F.4.307
SP Drawings
Structural part drawings are all the necessary component and assembly drawings. They include tooling interface, component, assembly and installation drawings.

F.4.308
SP Installation Drawings
The structural part installation drawings show how the structural part is installed in other assemblies.

F.4.309
SP Interface and Joint Constraints
The structural part interface and joint constraints are from the mating part's envelope and joint configuration.

F.4.310
SP Interface, Joint, Size, and Design Constraints
The structural part interface, joint, size, and design constraints are from the mating parts envelope and joint configuration. The overall size and design is dictated by the area it supports in the next higher assembly.

F.4.311
SP Layouts & Models
The structural part layouts consist of all the two dimensional geometry required of the design. The structural part models consist of the appropriate geometry required of the design.

F.4.312
SP Material Allowables
The stress and strain allowables for a composite material. The allowables are established by a material test program and further influenced by durability and damage tolerance requirements.

F.4.313
SP Material Properties
The material properties of the structural part.

F.4.314
SP Materials & Processes Data
All the necessary materials and processes (M&P) data for the composite materials that make up the structural part. Homogenous materials are considered a subset of these types of parts.

F.4.315
SP Model & Drawing Tree
The structural part model and drawing tree consist of a hierarchical relationship of the drawings and models of the structural part.

F.4.316
SP Models & Drawings
The structural part models and drawings are all the geometric and associated engineering notes for the structural part.

F.4.317
SP Parts List
Structural part parts list consist of all the material and subcomponents that make up the structural part list.

F.4.318
SP Preliminary Build System Description
The structural part preliminary build system description shows the system view of the manufacturing activities and resources needed to support the structural part production.

F.4.319
SP Preliminary Business System Description
The structural part preliminary business system description contains the system relationships of the budget, schedule and costs of a preliminary structural part.

F.4.320
SP Preliminary Design System Description
The structural part preliminary design system description shows the functional, geometrical and fitup of a preliminary design of a structural part using a graphical/textual system engineering language.

F.4.321
SP Preliminary Logistics System Description
The structural part preliminary logistics system description shows the system view of the relationship between the logistics engineering, reliability and maintainability, spares and training systems of a preliminary structural part design.

F.4.322
SP Preliminary System Description
The preliminary structural part system description consists of the business, design, build and logistics system description.

F.4.323
SP Producibility & Logistics Data & Build Changes
The as-built configuration management data of the structural part that must be incorporated into the documentation of the designed structural part.

F.4.324
SP Producibility & Logistics Data
The structural part producibility and logistics data is derived from the various manufacturing, maintainability and field product support studies done for the structural part.

F.4.325
SP Product Development Information
The structural part product development information consists of the business management, design definition package, and logistics support for the structural part.

F.4.326
SP Prototypes
The structural part prototypes consist of all the pre-production physical or electronic models of the structural part.

F.4.327
SP Raw Material
The actual raw material that is used to build the structural part.

F.4.328
SP Raw Materials & Material Properties
All of the raw materials as received from the material supplier. It includes the properties of the materials, as well as the rough material stock.

F.4.329
SP Recommended Drawing Changes
The recommended changes to the structural part design that evolved as a result of the detail structural part analysis.

F.4.330
SP Review Data
Structural part review data is that which has been done in the in-house and customer reviews of the design of the structural part.

F.4.331
SP Size & Design Constraints
The structural part size constraints are the tooling and design constraints that dictate the length and width features of the structural part.

F.4.332
SP Solid
The structural part solid is the graphical and physical electronic representation of the structural part.

F.4.333
SP Spares
The spare parts required to replace the structural part.

F.4.334
SP Structural Interfaces
All the structural interfaces that mate with the structural part.

F.4.335
SP Support Logistics Data
The structural part support logistics data consists of all the logistics engineering, reliability and maintainability, technical and maintenance documents, and spares data needed to support the structural part.

F.4.336
SP Surface
The structural part surface consists of the graphical rendering of the boundary faces of the part.

F.4.337
SP Test Data
All of the test data from the structural verification test of the structural part and its subcomponents.

F.4.338
SP Tooling Interface Drawings
The structural part tooling interface drawings consist of the inner mold line (IML) or outer mold line (OML) of the tool interfaces to the structural part.

F.4.339
Stacking Sequence
The orientations of the plies in an structural part in order that the plies are laid down on the manufacturing tool.

F.4.340
Staff & Tools
The staff and tools from the contractor, the customer and materials supplier.

— **SP M/D/B/S Staff:** The specific people and tools necessary to perform the manage, design, build and support functions.

— **Customer Staff & Tools:** The specific people and tools necessary to perform the customer tasks.

— **Material Supplier Staff & Tools:** The specific people and tools necessary to perform the material development tasks.

F.4.341
STEP Exchange Format
An exchange format that conforms to the STEP international standard.

F.4.342
STEP Finite Element Analysis Controls Format
The finite element analysis (FEA) controls in STEP format.

F.4.343
STEP Finite Element Analysis Model Format
The finite element analysis (FEA) model in STEP format.

F.4.344
STEP Finite Element Analysis Output Format
The analysis output data from a finite element analysis in STEP format.

F.4.345
Structural Part
The as-built structural part.

F.4.346
Structural Part Maintenance Techniques
The maintenance techniques as practiced in the field.

F.4.347
Structural Part Material Development Techniques
The material development techniques as practiced by the material suppliers.

F.4.348
Structural Part Product Development & Maintenance Techniques
The techniques consisting of structural part product development, maintenance and material development techniques.

F.4.349
Structural Part Product Development Techniques
The structural part product development techniques for composites and homogenous materials involve the design, analysis, and build procedures and specifications used within industry.

F.4.350
Support Logistics of an SP*
The logistics engineering, reliability and maintainability design studies, technical and maintenance documents, spares and training systems that the structural part repairs.

F.4.351
Test Plan
A plan developed to describe the testing process of the test part.

F.4.352
Tested SP Prototypes
The tested structural part prototypes consist of all the pre-production physical or electronic models of the structural part that have been tested.

F.4.353
Textual Documentation
Textual documentation of the analysis output data from a finite element analysis.

F.4.354
Thermal/Moisture Data
The thermal and moisture environment of the structural part.

F.4.355
Thicknesses
The thickness of a surface element perpendicular to the surface of the element. A surface element idealizes a volume continuum as a surface with a thickness distribution.

F.4.356
Translate Finite Element Analysis Model and Finite Element Analysis Controls from STEP Format
Translate analysis input data from STEP format into the native format for the finite element analysis (FEA) application.

F.4.357
Translate Finite Element Analysis Output Data into STEP Format
Translate the finite element analysis (FEA) output data from the analysis application's native format into STEP format.

F.4.358
Translate SP Data from Finite Element Analysis Solver*
Translate analysis output data from an existing solution into an internal loads/stress database application.

F.4.359
Translate SP Data from STEP Format
Translate analysis output data from STEP format into an internal loads/stress database application.

F.4.360
Update SP Drawing & Model Data*
Update the structural part drawings and models based on the changes to the structural part.

F.4.361
Updated SP Models & Drawings
The updated structural part drawings and models are those that have incorporated the most recent design changes.

F.4.362
Use & Maintain an SP*
The customer's use and maintenance of a structural part. It also includes repair, redesign, and modification activities of a structural part at the servicing location.

F.4.363
Used SP
The structural part as a result of use in the field.

F.4.364
Verified SP Translated CAD Data
The verified structural part translated data is the geometry and text that was successfully translated and received in by the CAD toolset, and is now used to develop the structural part geometry.

F.5 Application activity model diagrams

The application activity model diagrams are given in Figures F.4 through F.47. The graphical form of the application activity model is presented in the IDEF0activity modelling format. Activities and data flows that are out of scope are shown with dashed lines.

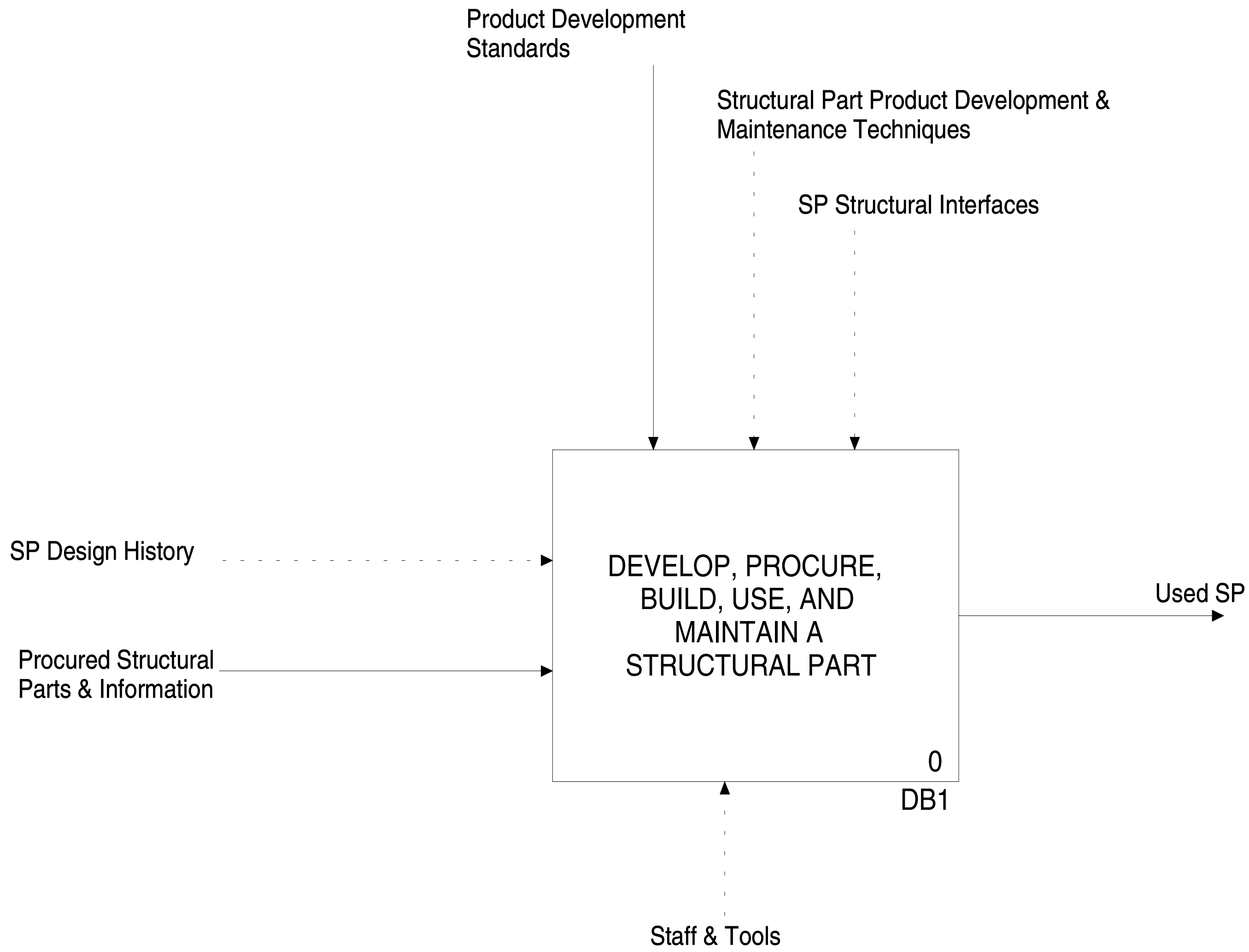

Figure F.4 - A-0 Develop, procure, build, use and maintain an SP

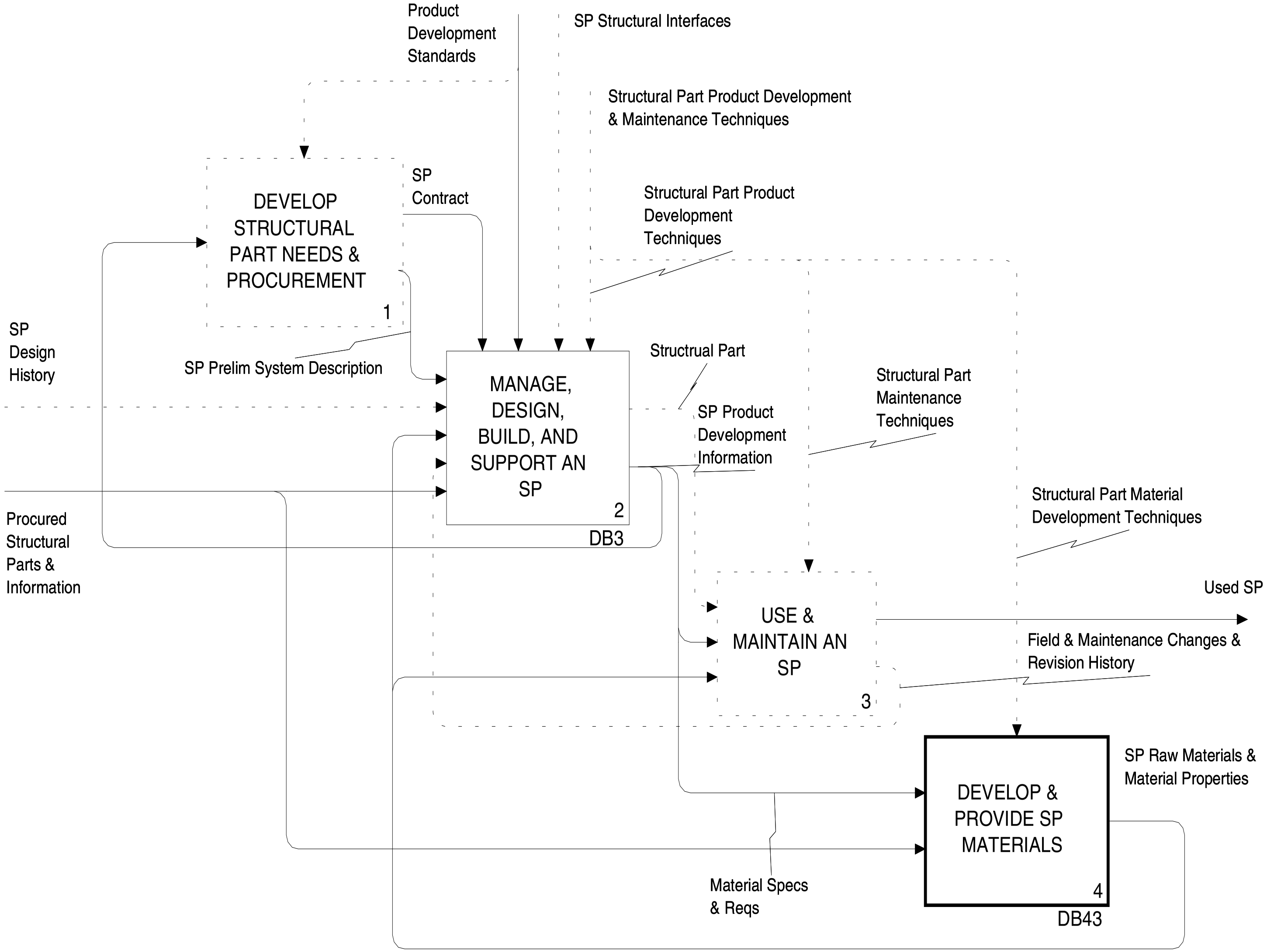

Figure F.5 - A0 Develop, procure, build, use and maintain an SP

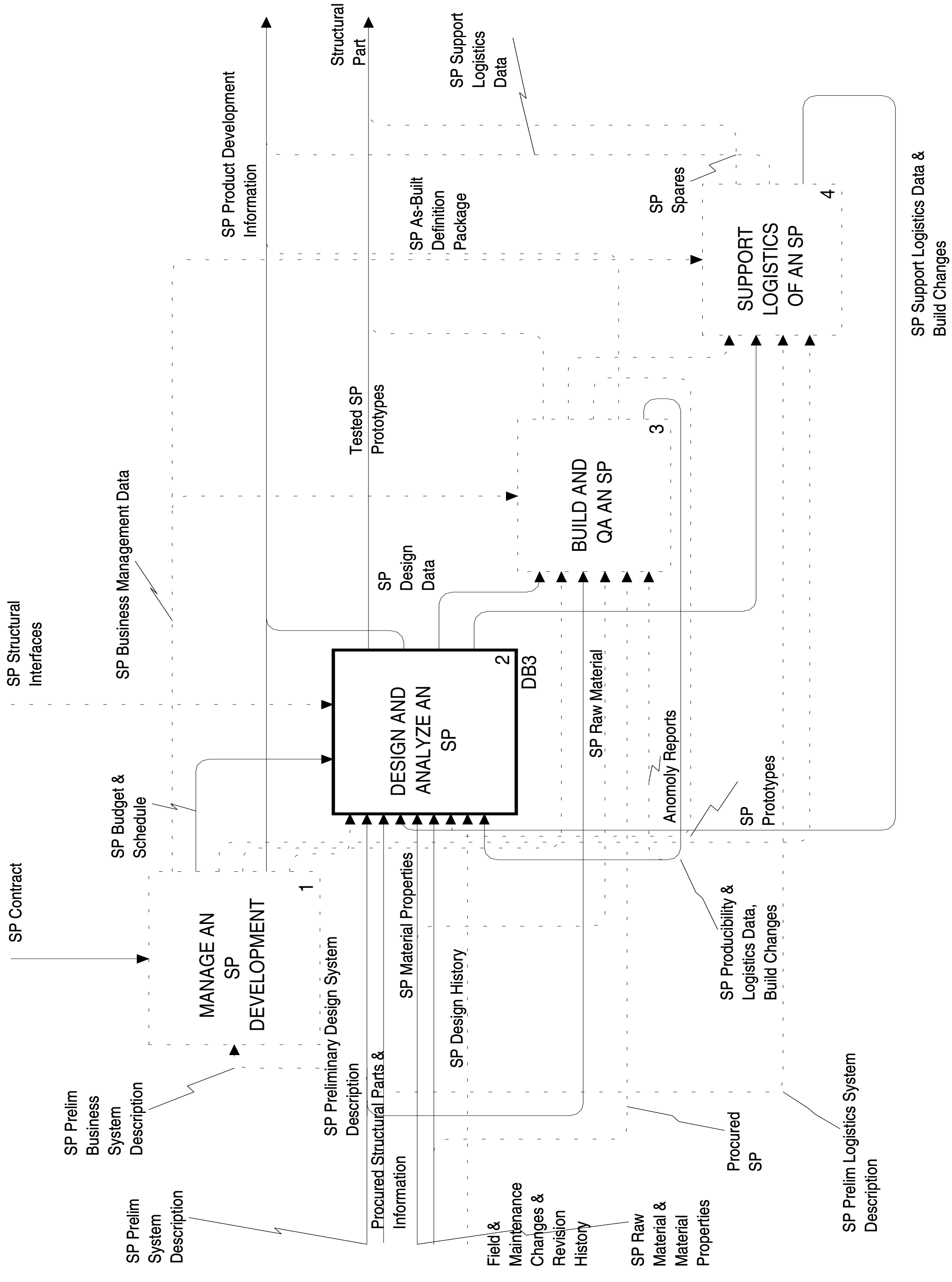

Figure F.6 - A2 manage, design, build, and support an SP

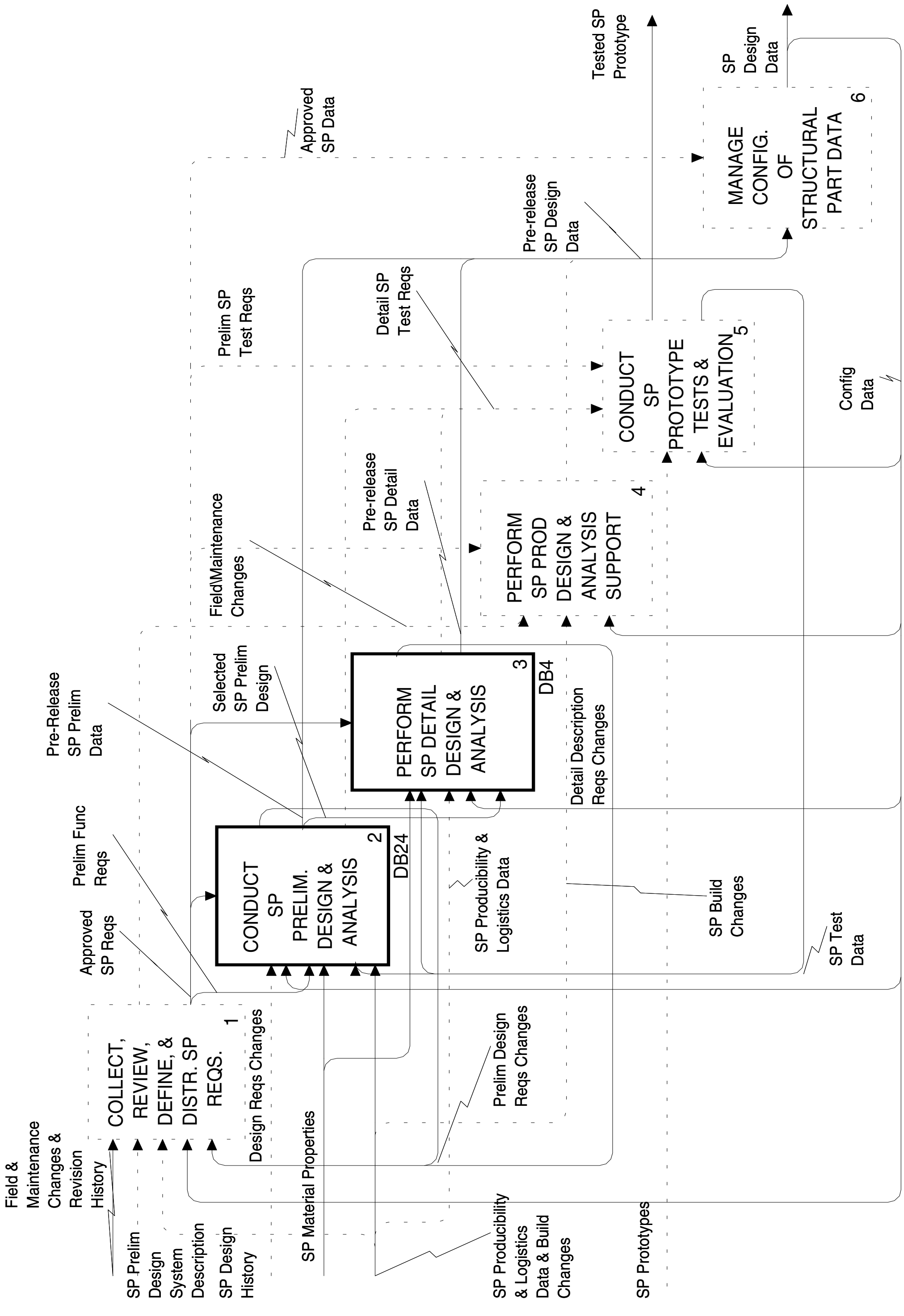

Figure F.7 - A22 design and analyze an SP

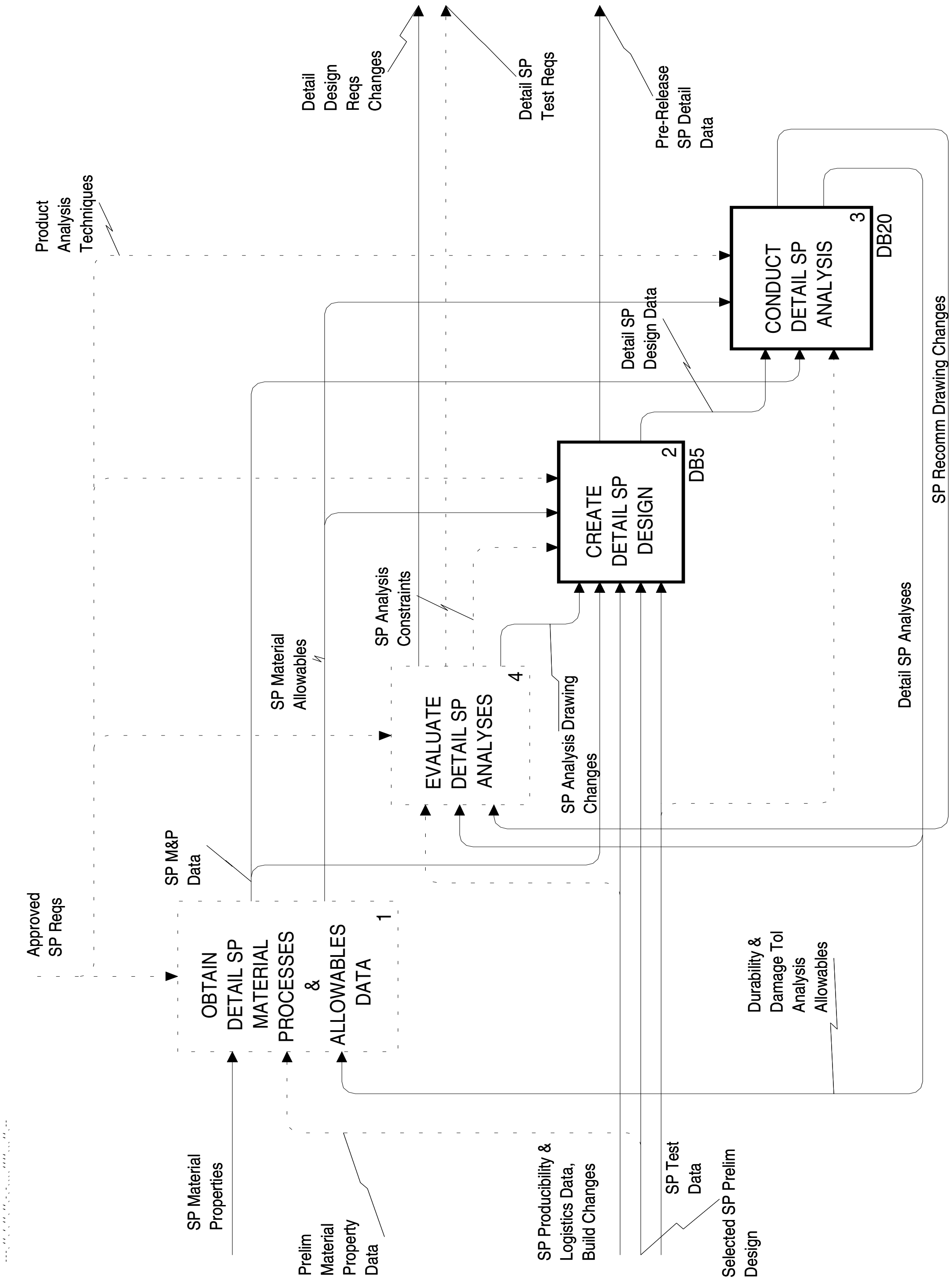

Figure F.8 - A223 perform structural part detail design and analysis

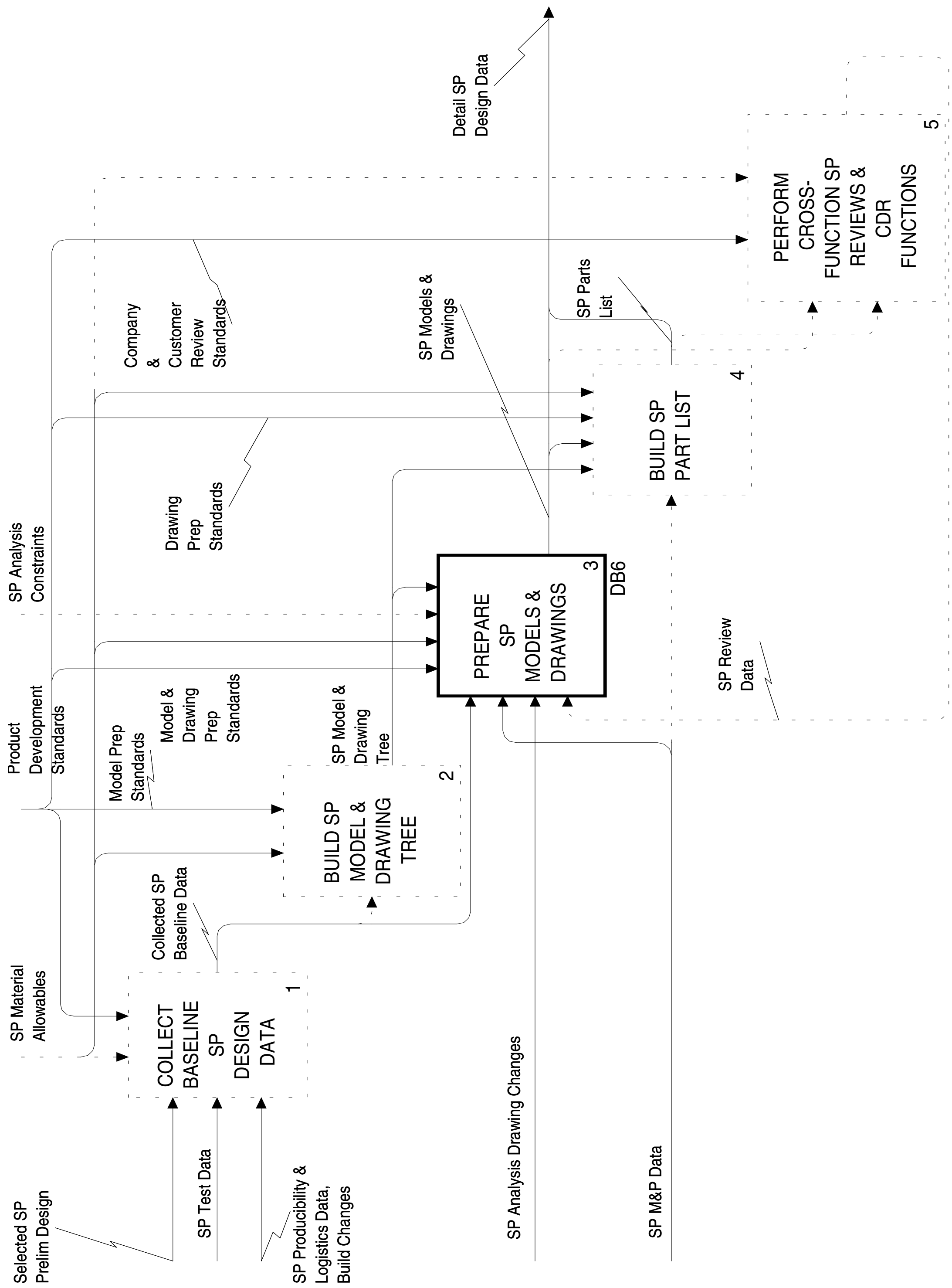

Figure F.9 - A2232 create detail structural part design

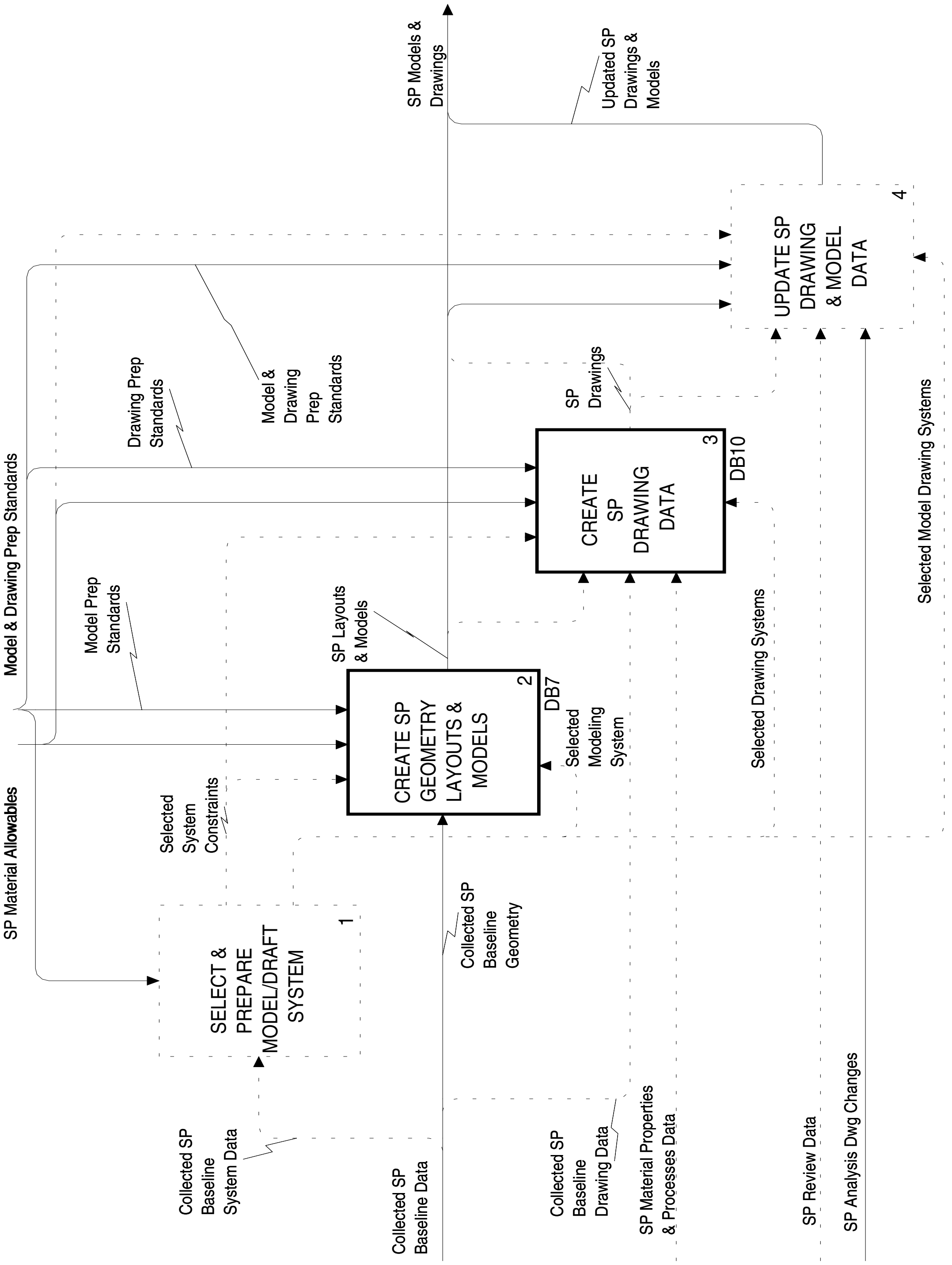

Figure F.10 - A22323 prepare SP models and drawing

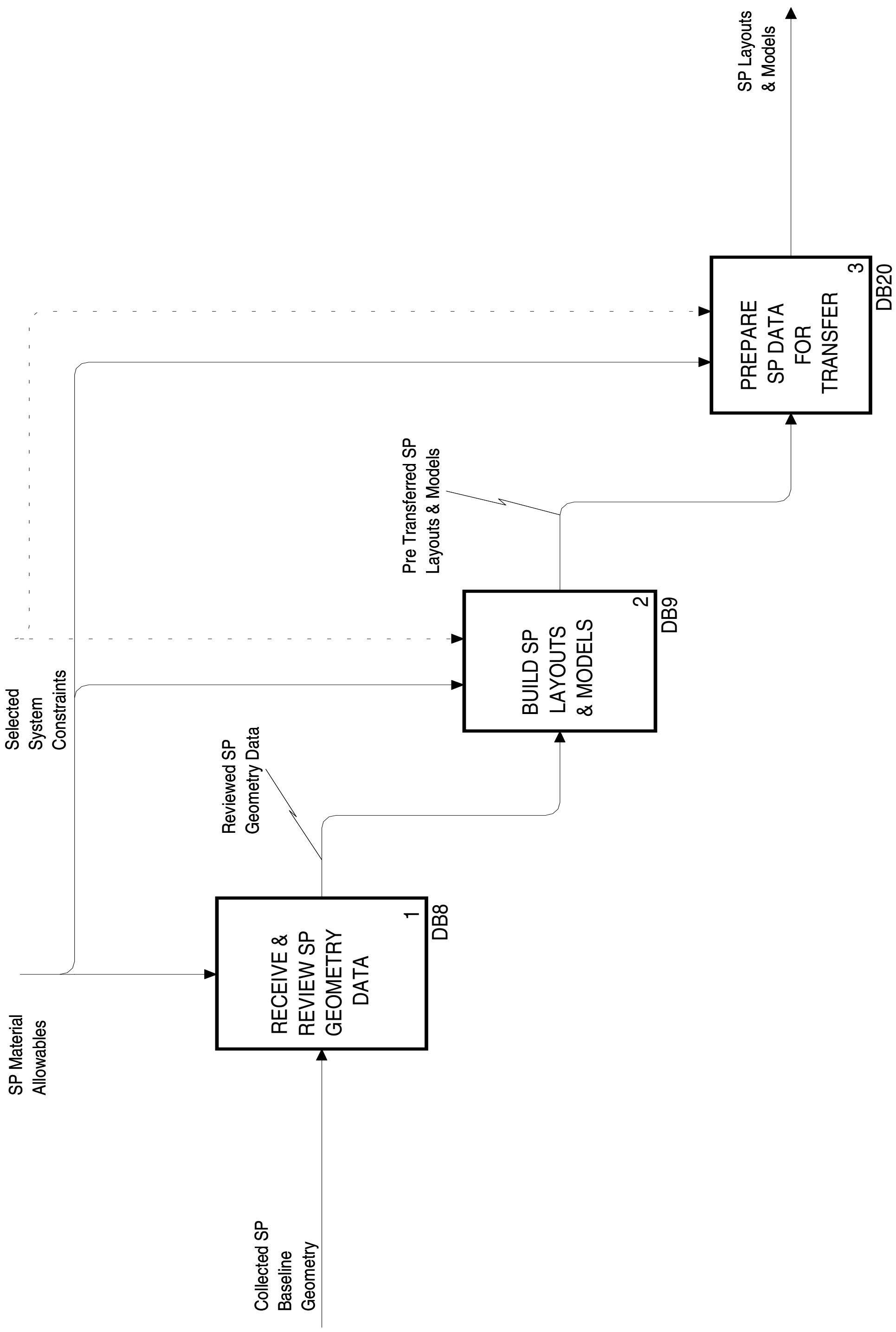

Figure F.11 - A223232 create SP geometry layouts and models

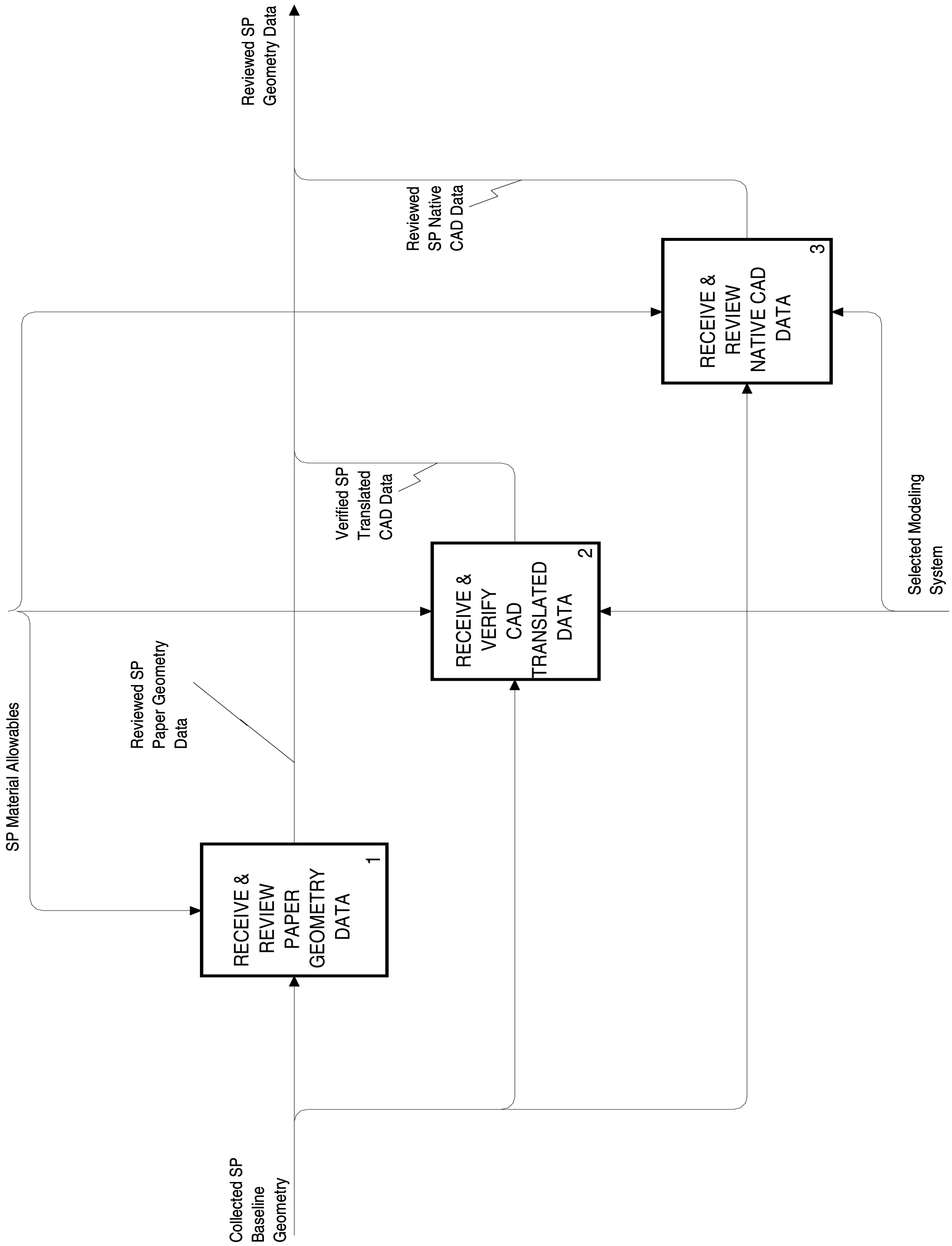

Figure F.12 - A2232321 receive and review SP geometry data

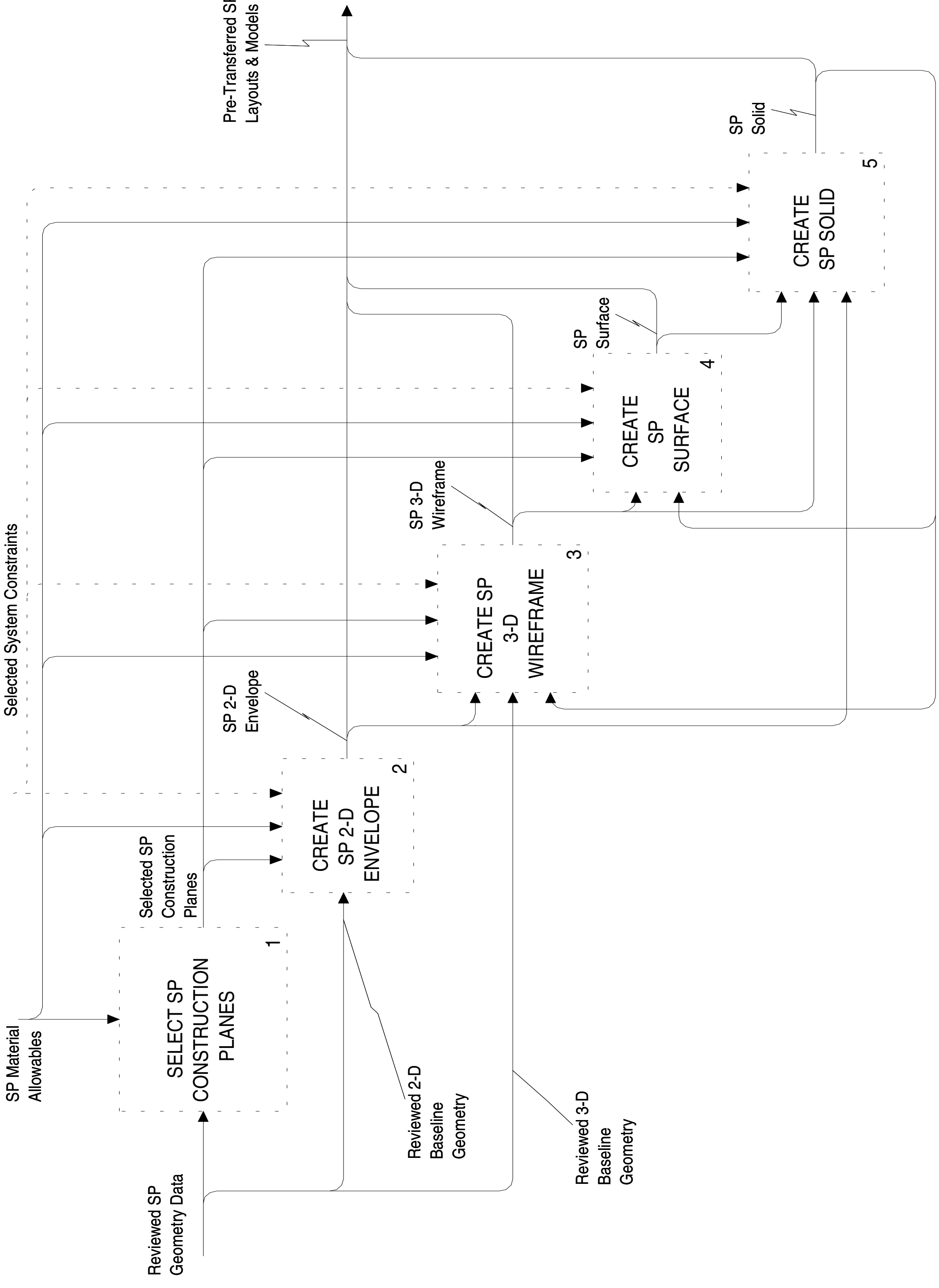

Figure F.13 - A22232322 build SP layouts and models

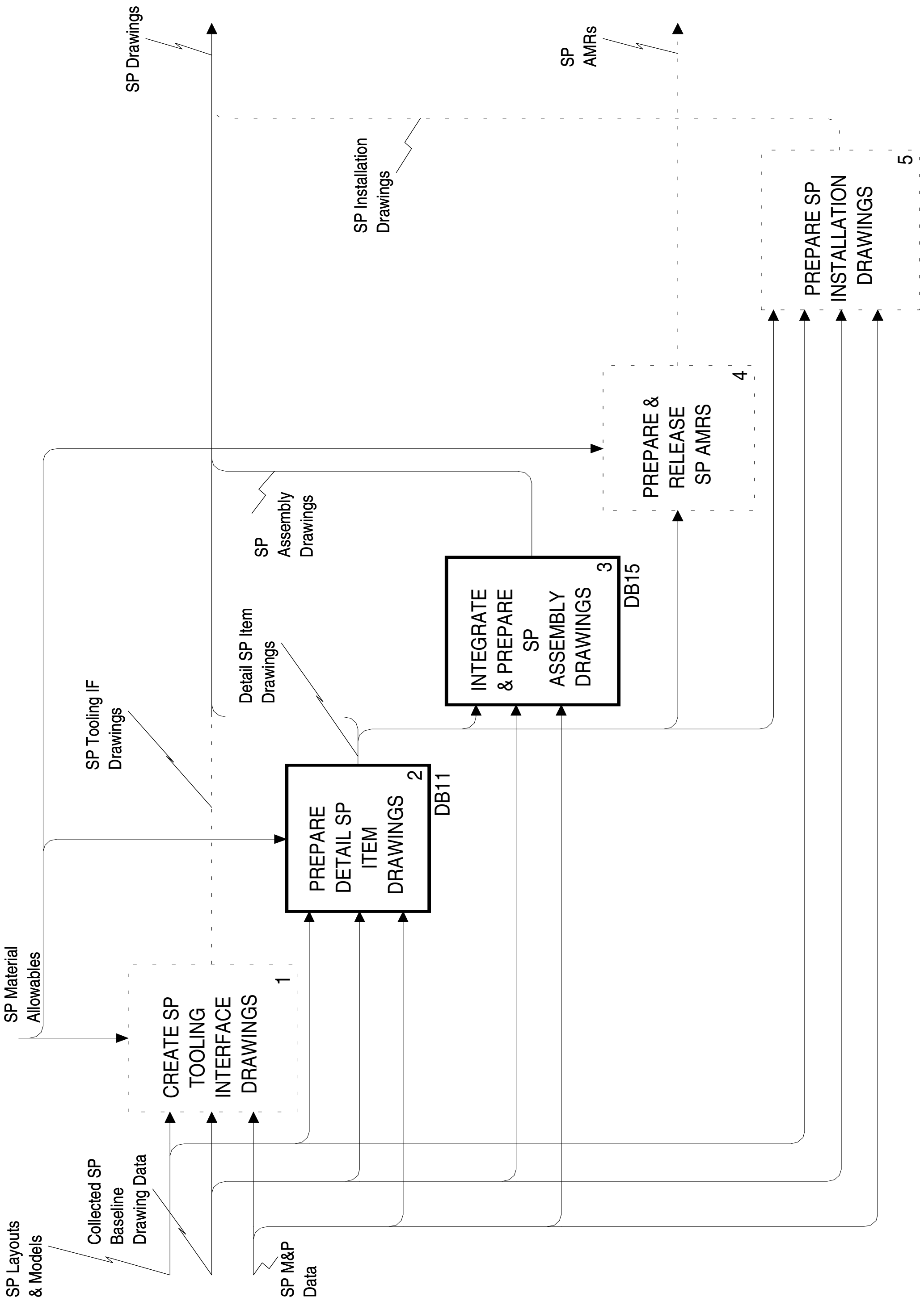

Figure F.14 - A223233 create SP drawing data

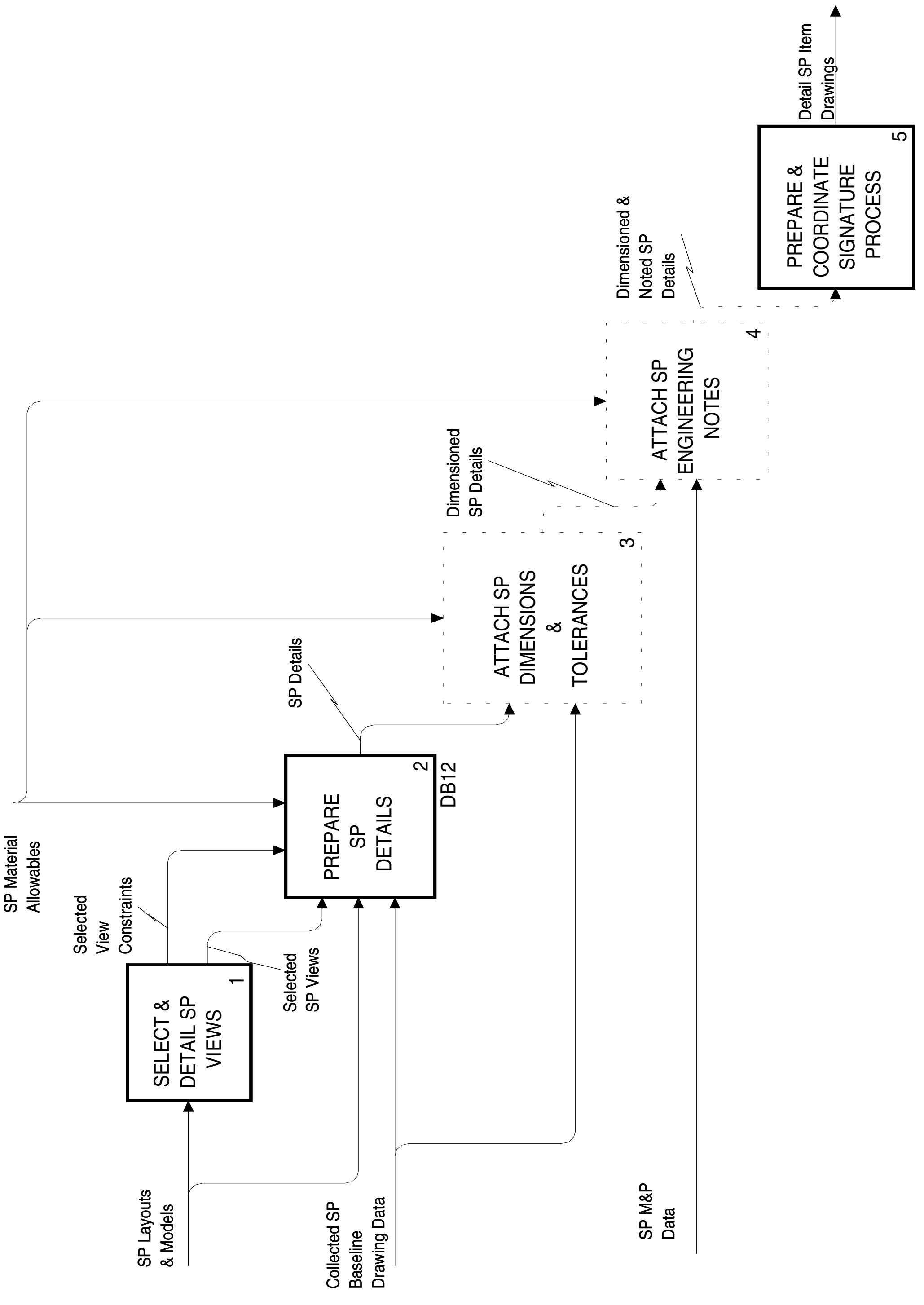

Figure F.15 - A2232332 prepare detail SP item drawings

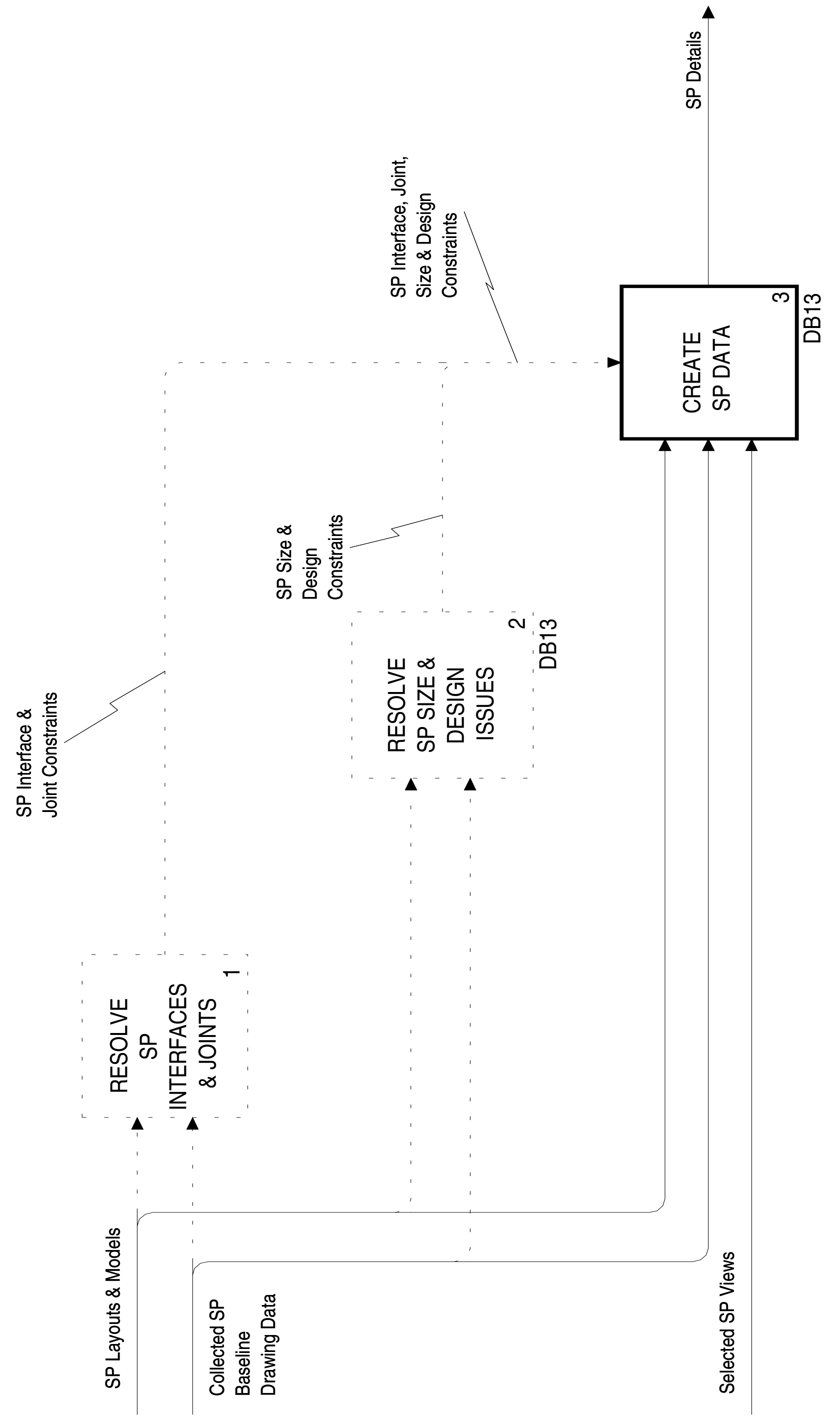

Figure F.16 - A222323322 prepare SP details

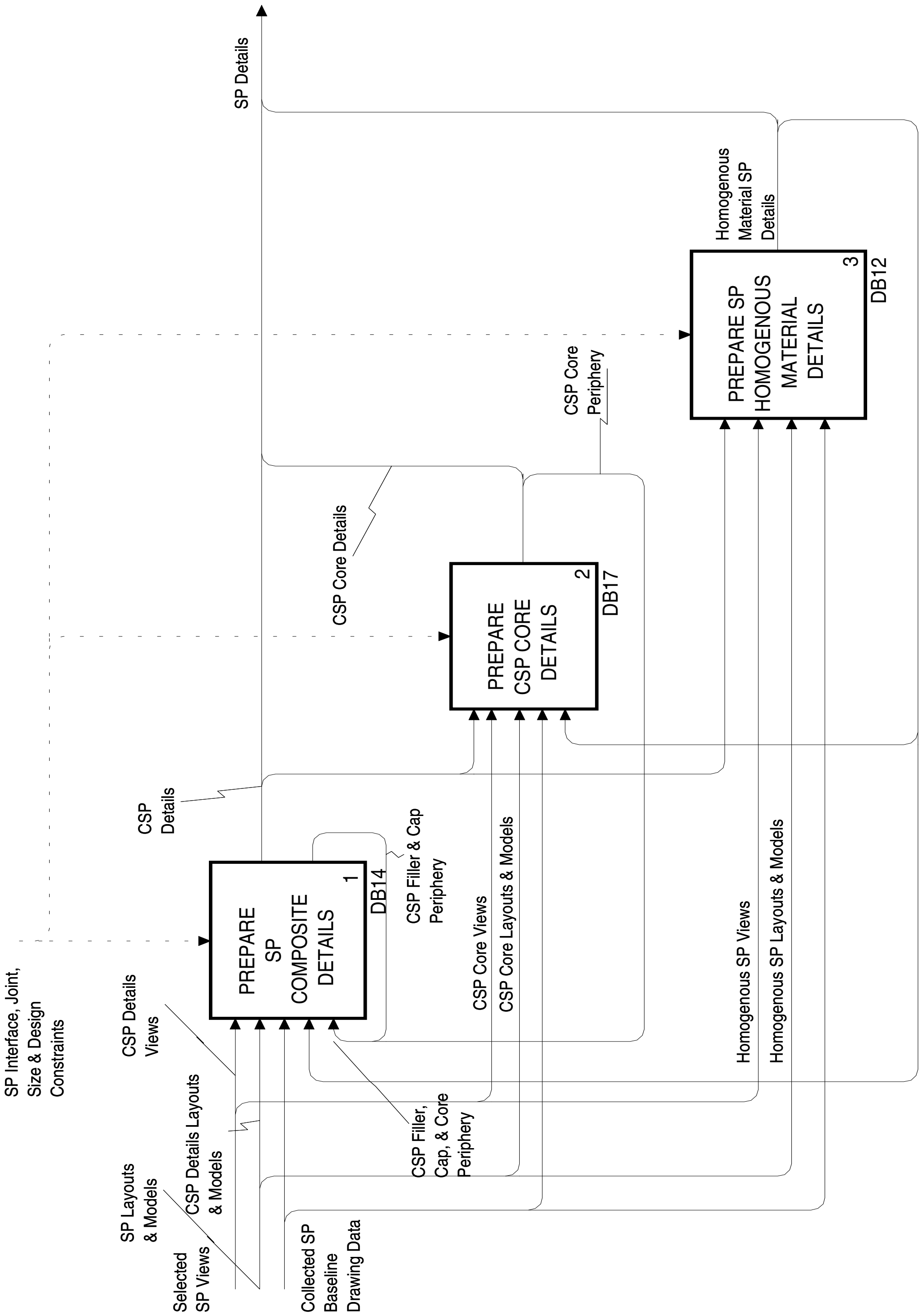

Figure F.17 - A22232332233 create SP data

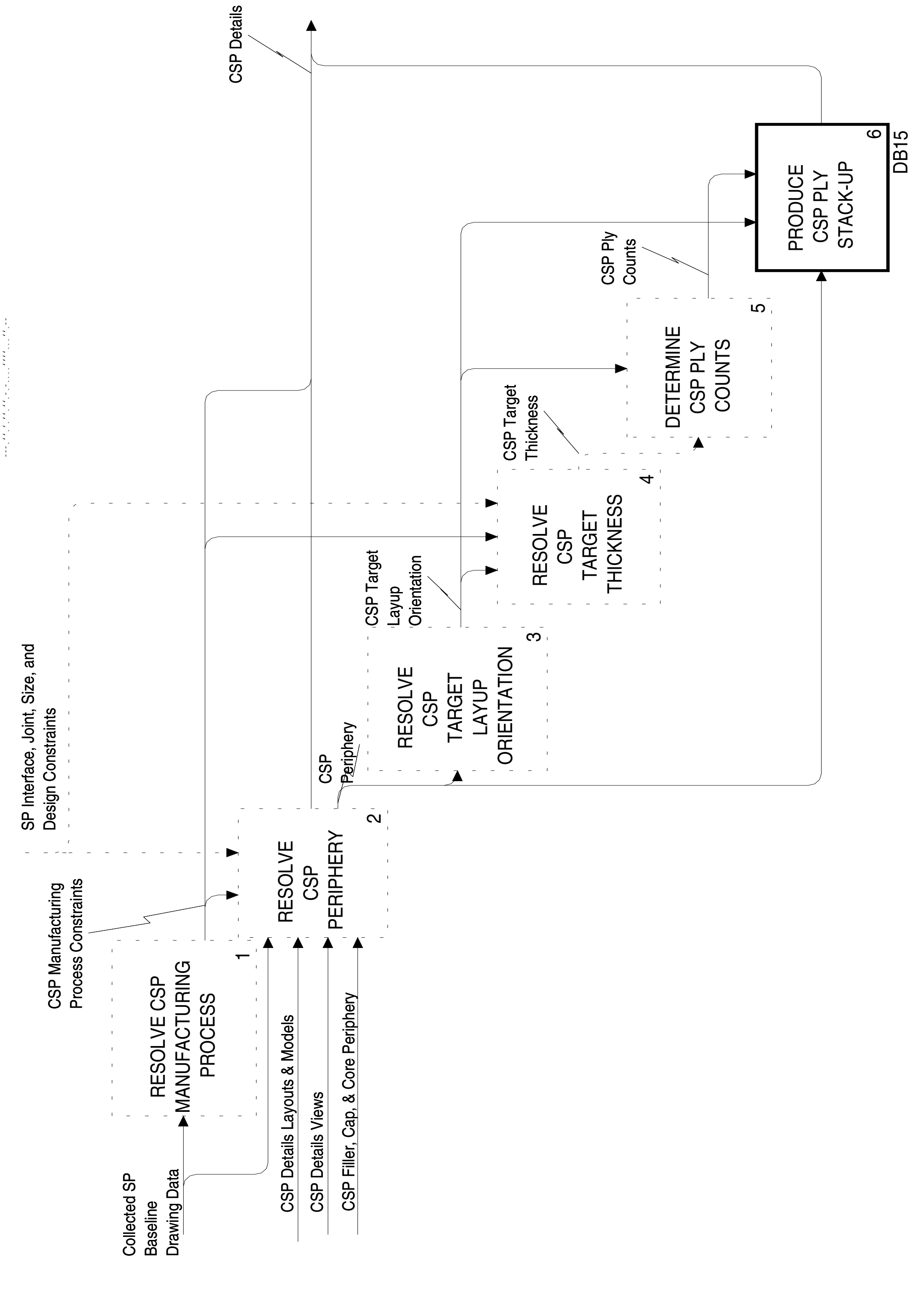

Figure F.18 - A22232332231 prepare SP composite detail

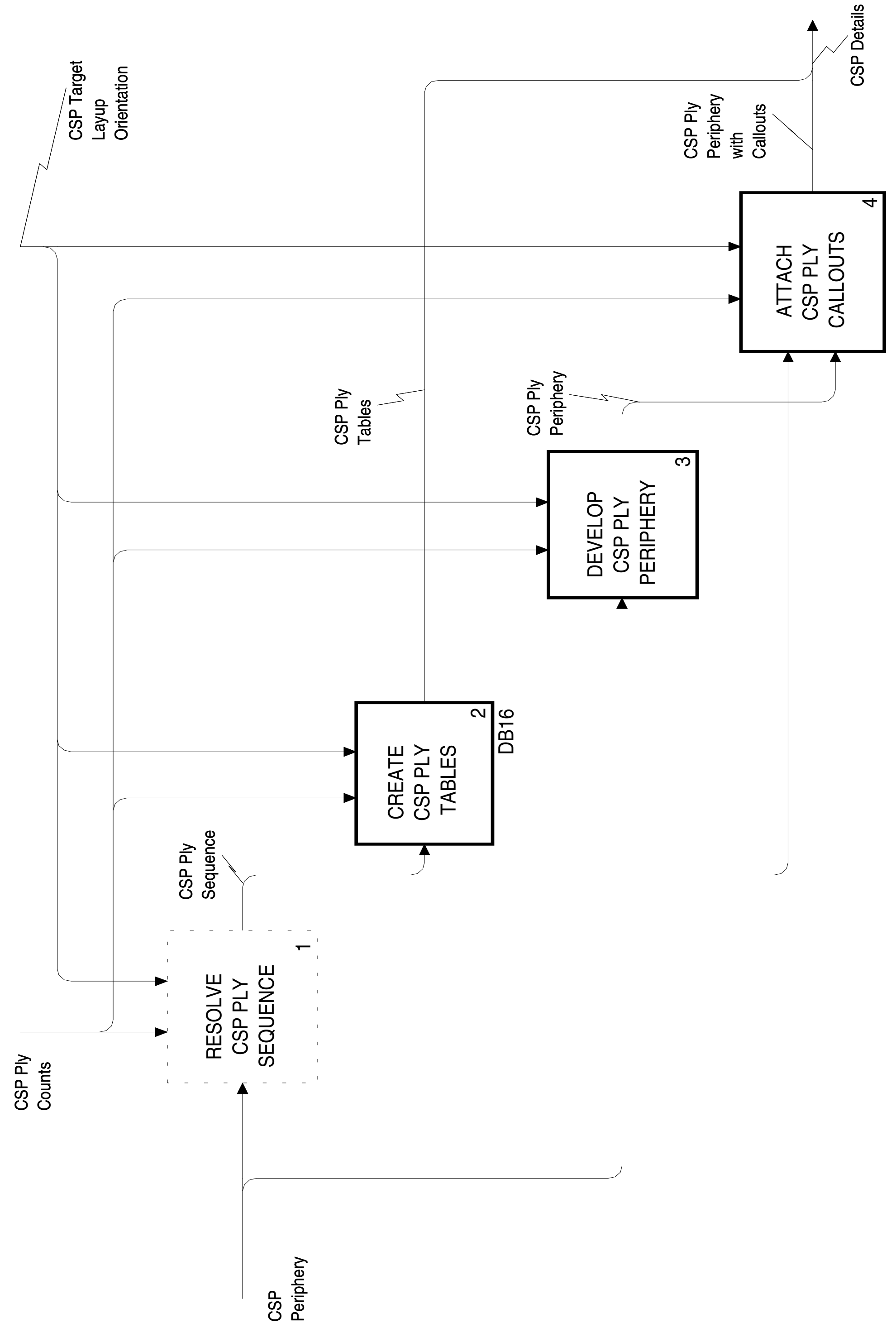

Figure F.19 - A2232323222316 produce CSP ply stack-up

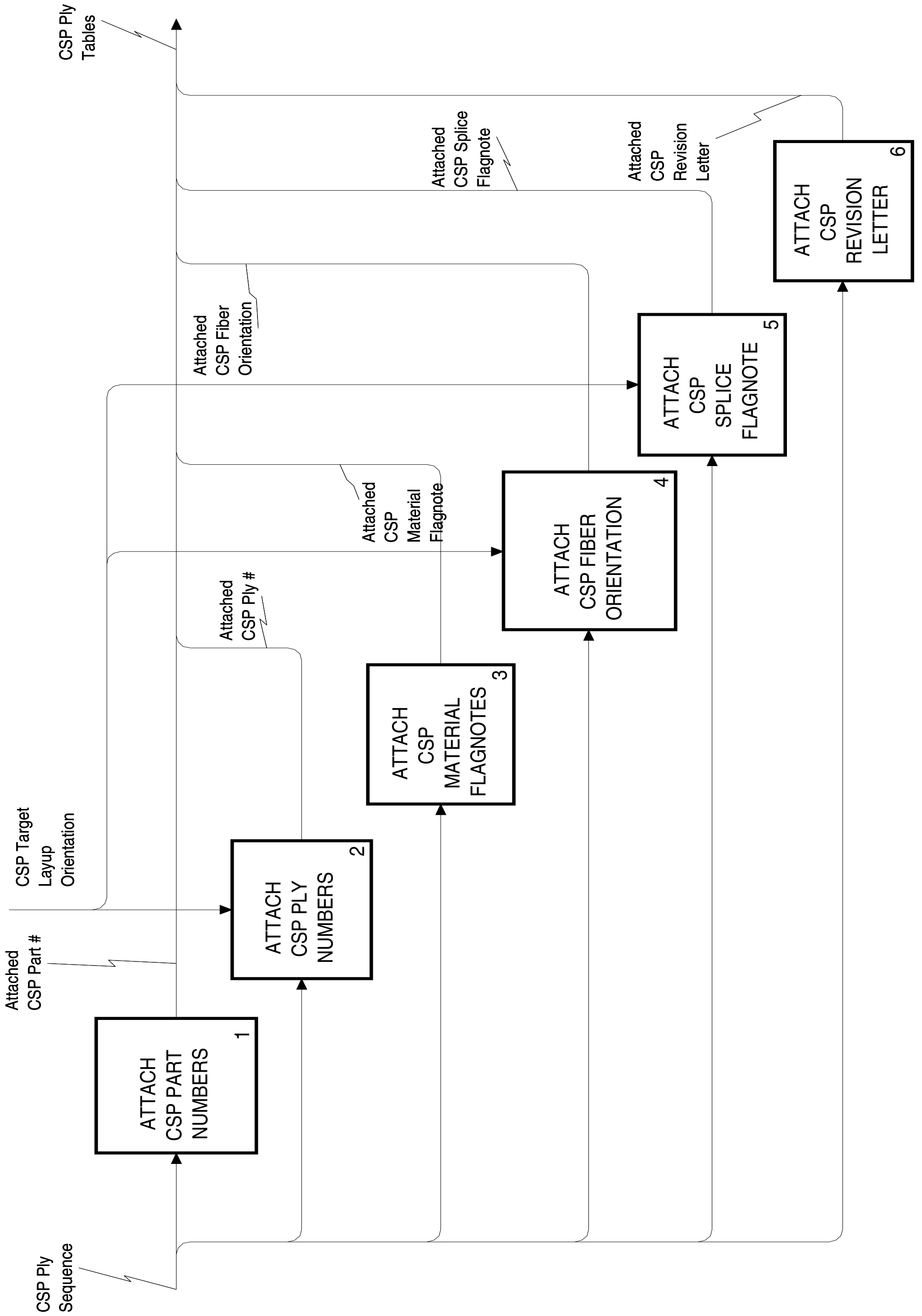

Figure F.20 - A22232322223162 create CSP ply tables

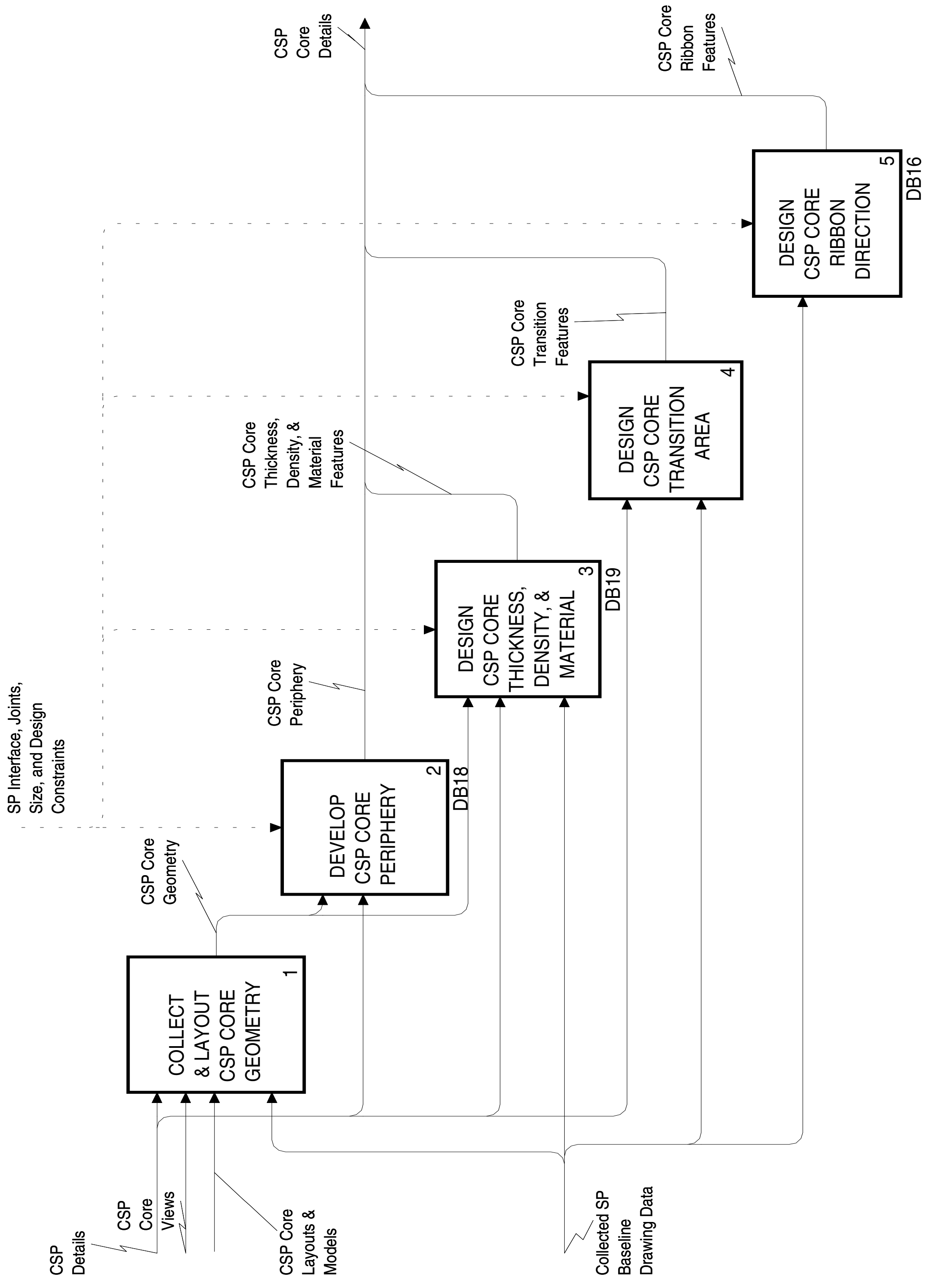

Figure F.21 - A22323332232 prepare CSP core details

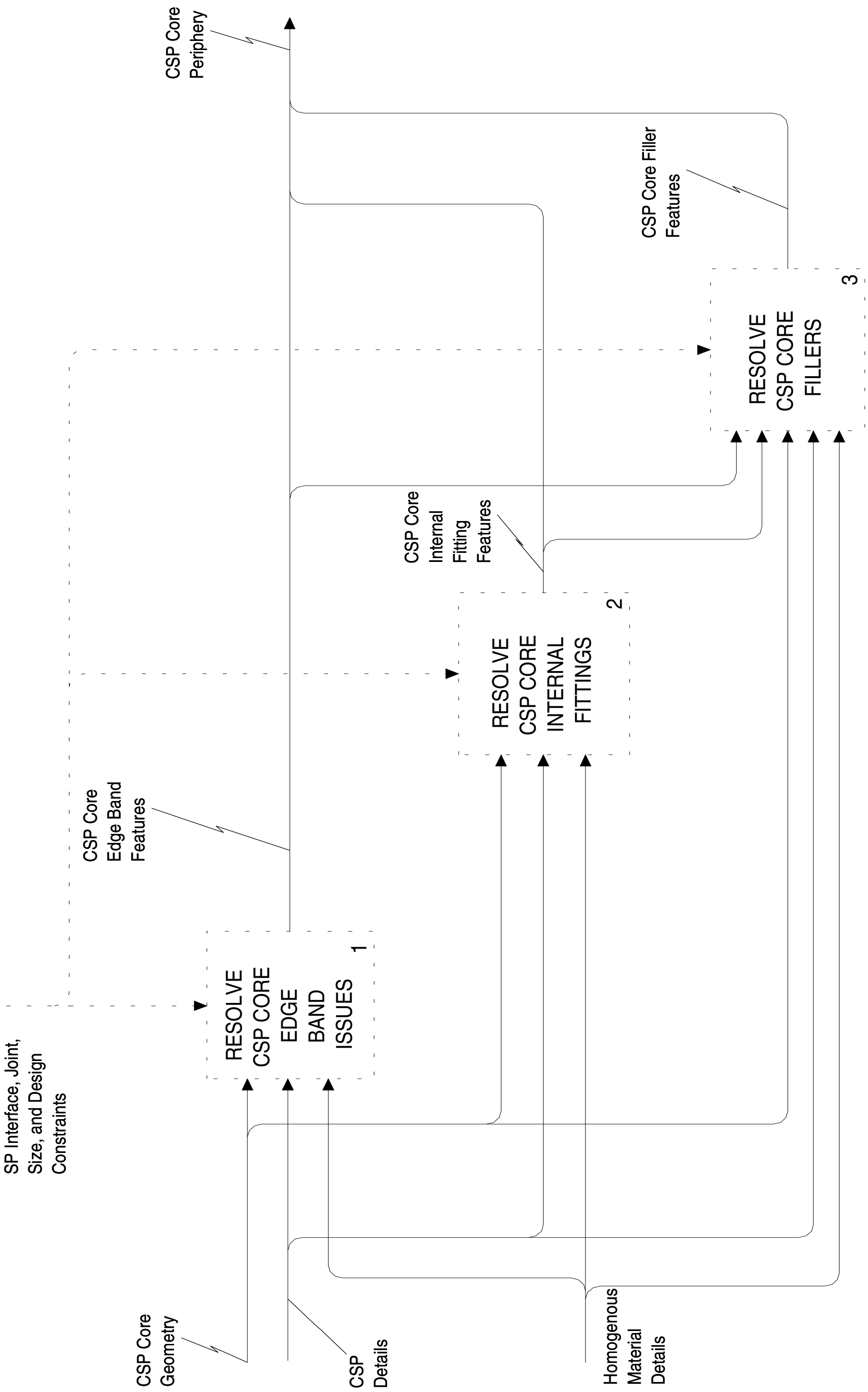

Figure F.22 - A22323322322 develop CSP core periphery

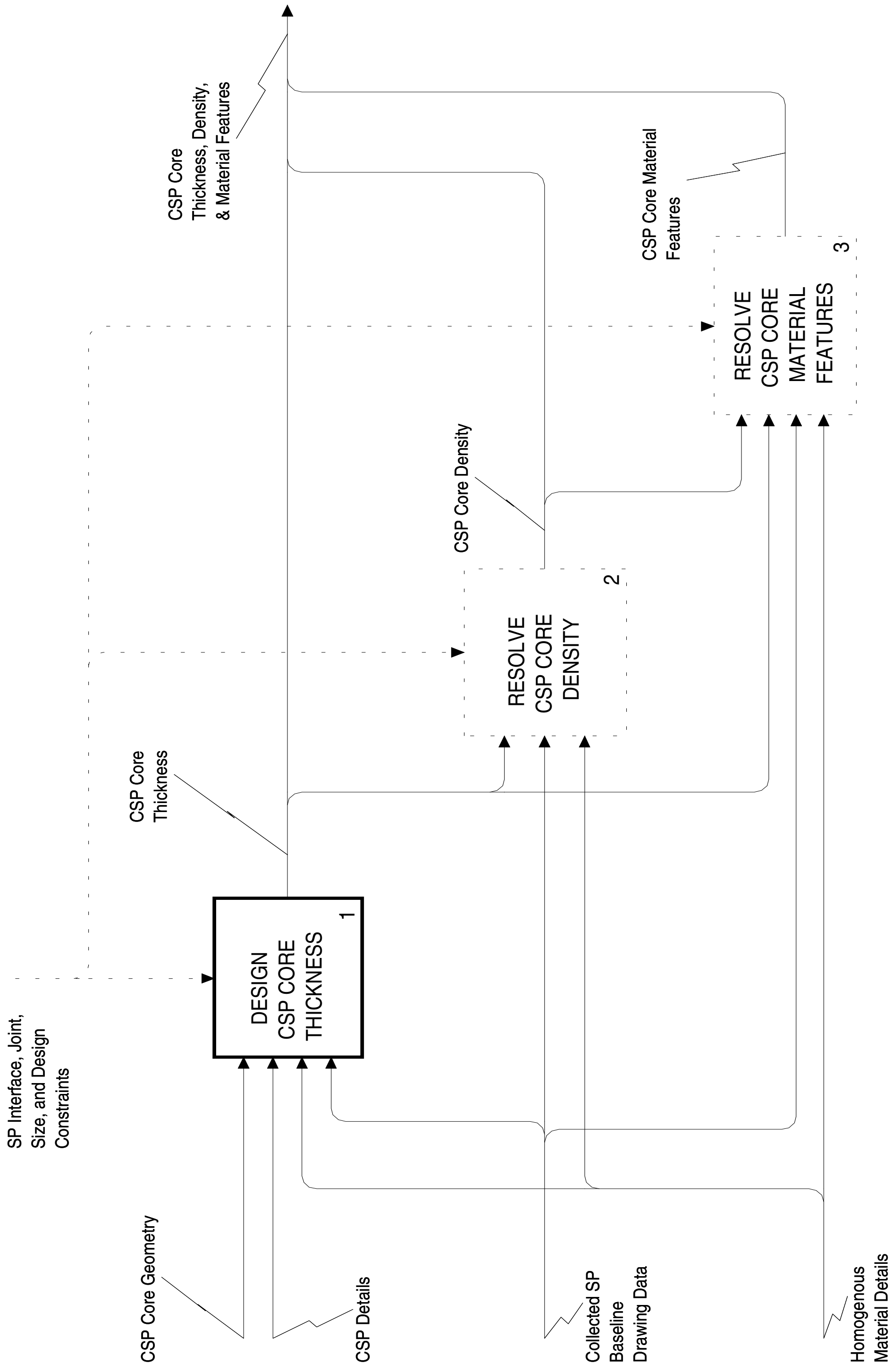

Figure F.23 - A22323322323 Design CPS core thickness, density, and material features

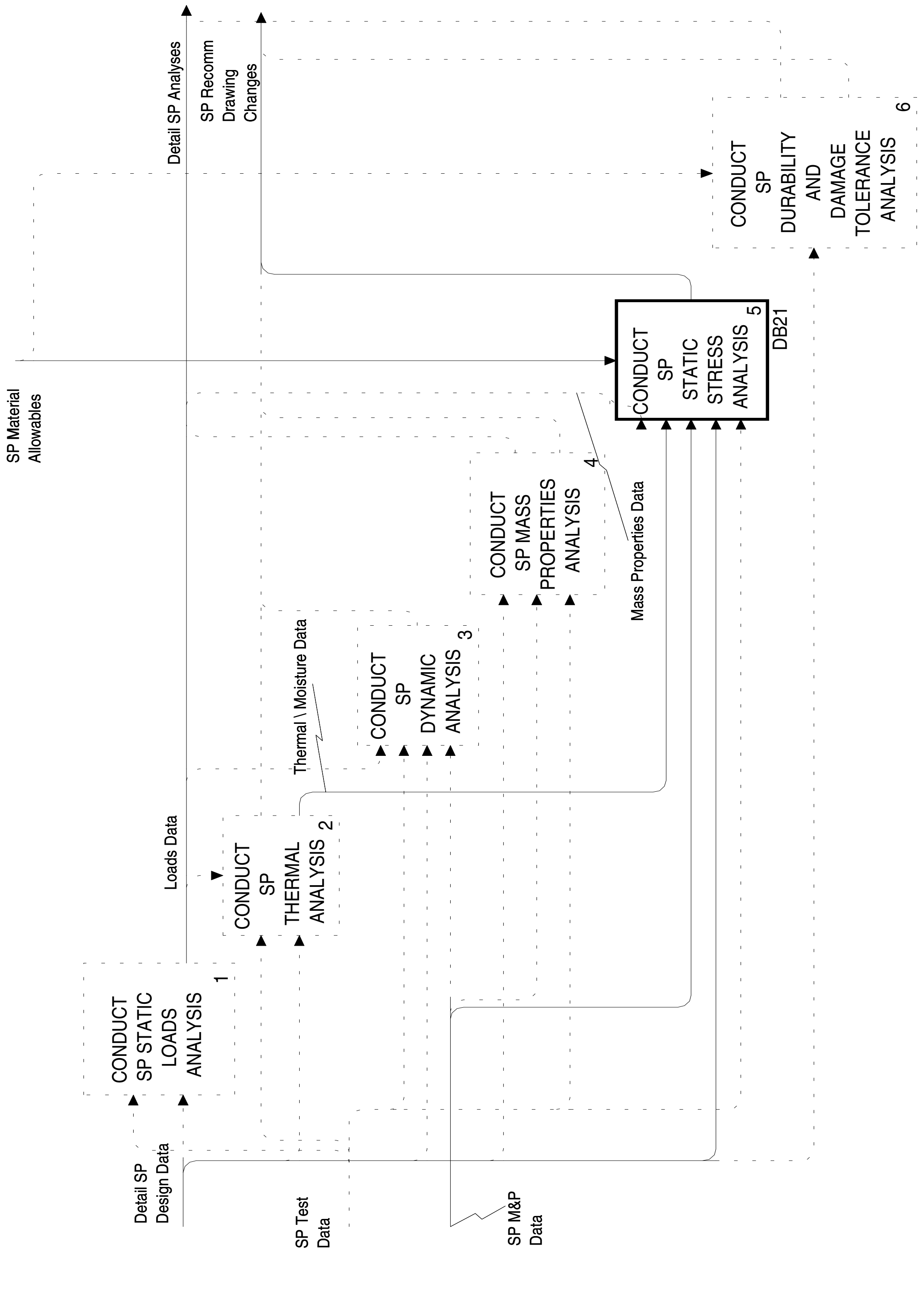

Figure F.24 - A2233 conduct detail SP analysis

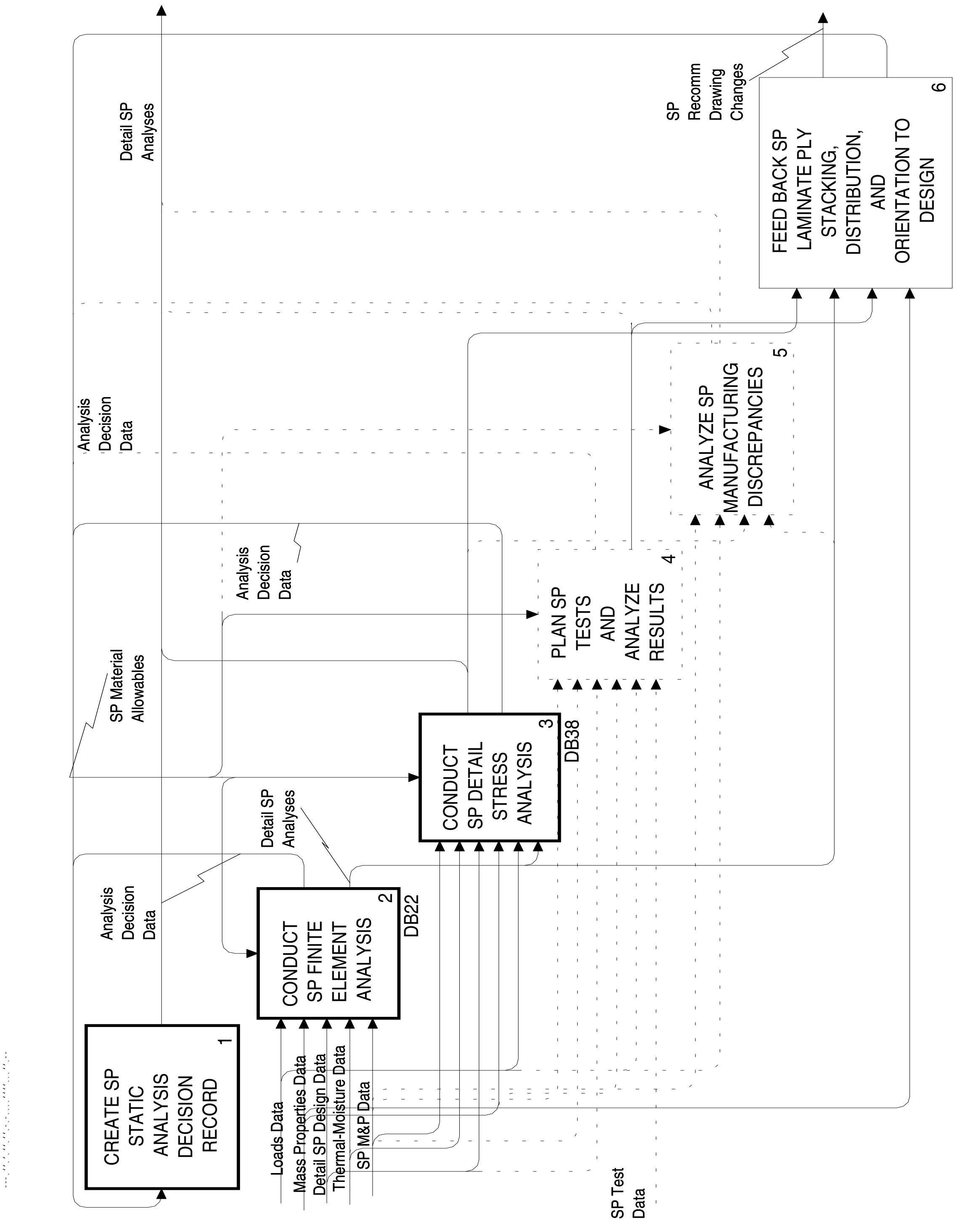

Figure F.25 - A22335 conduct SP static stress analysis

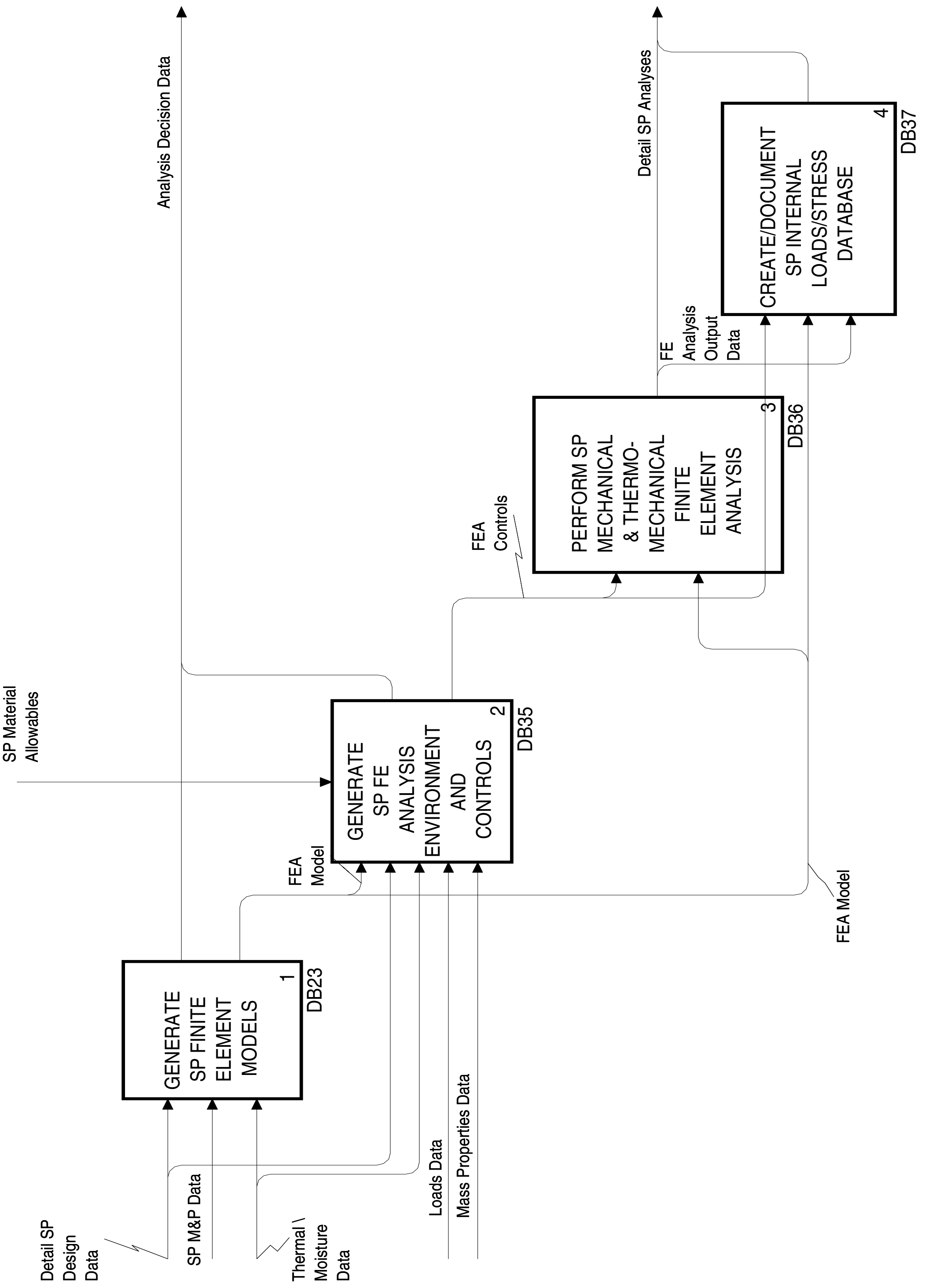

Figure F.26 - A223352 conduct SP finite element analysis

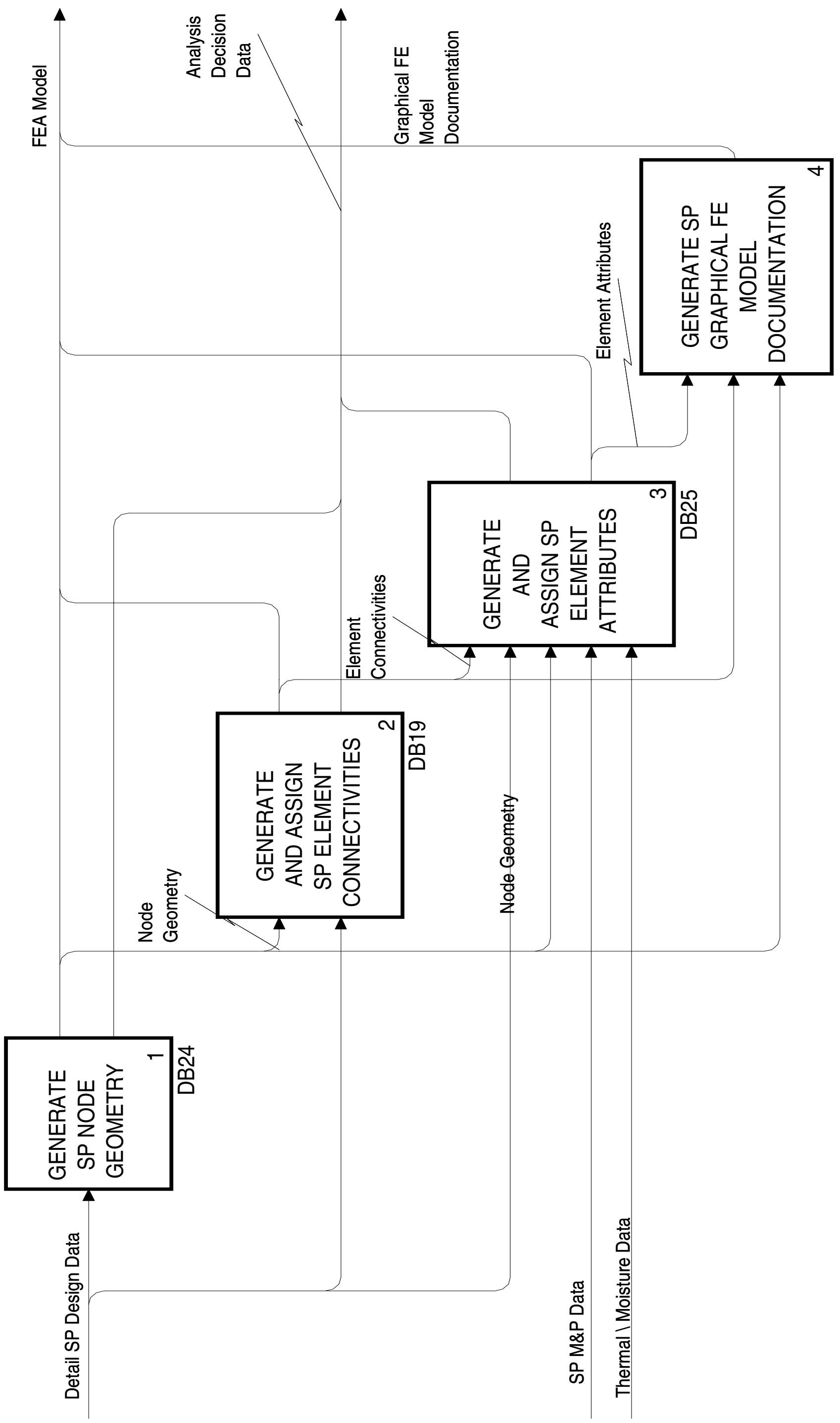

Figure F.27 - A2233521 generate SP finite element models

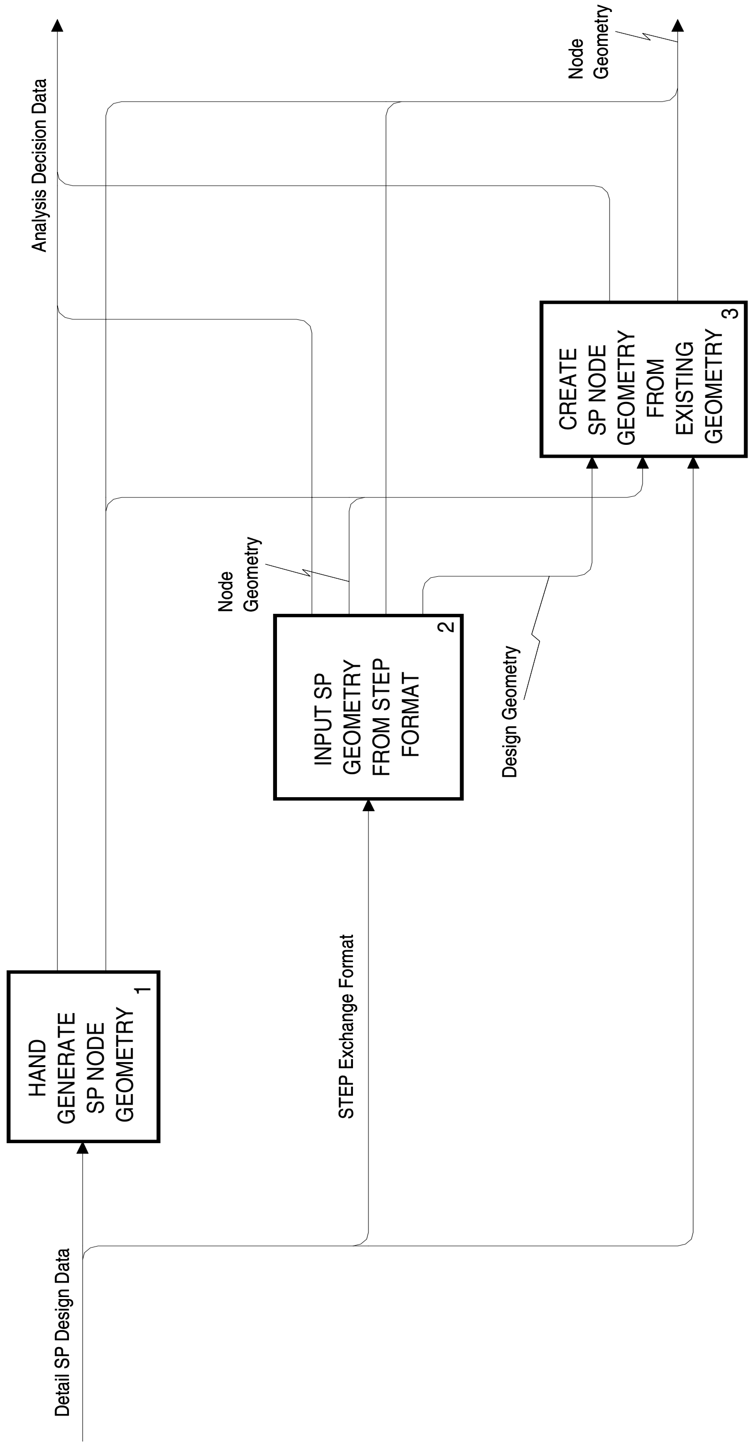

Figure F.28 - A22335211 generate SP node geometry

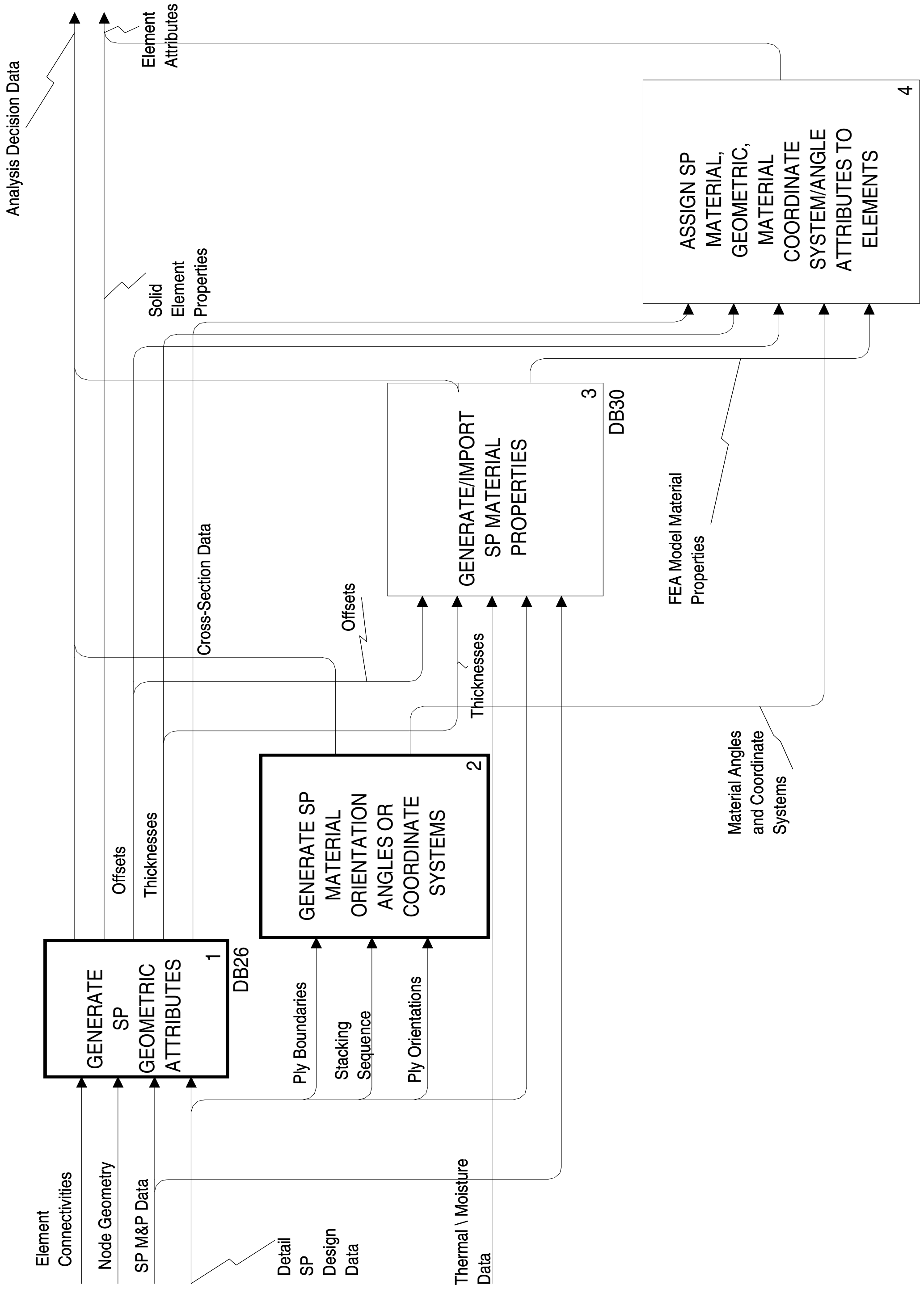

Figure F.29 - A22335213 generate and assign SP element attributes

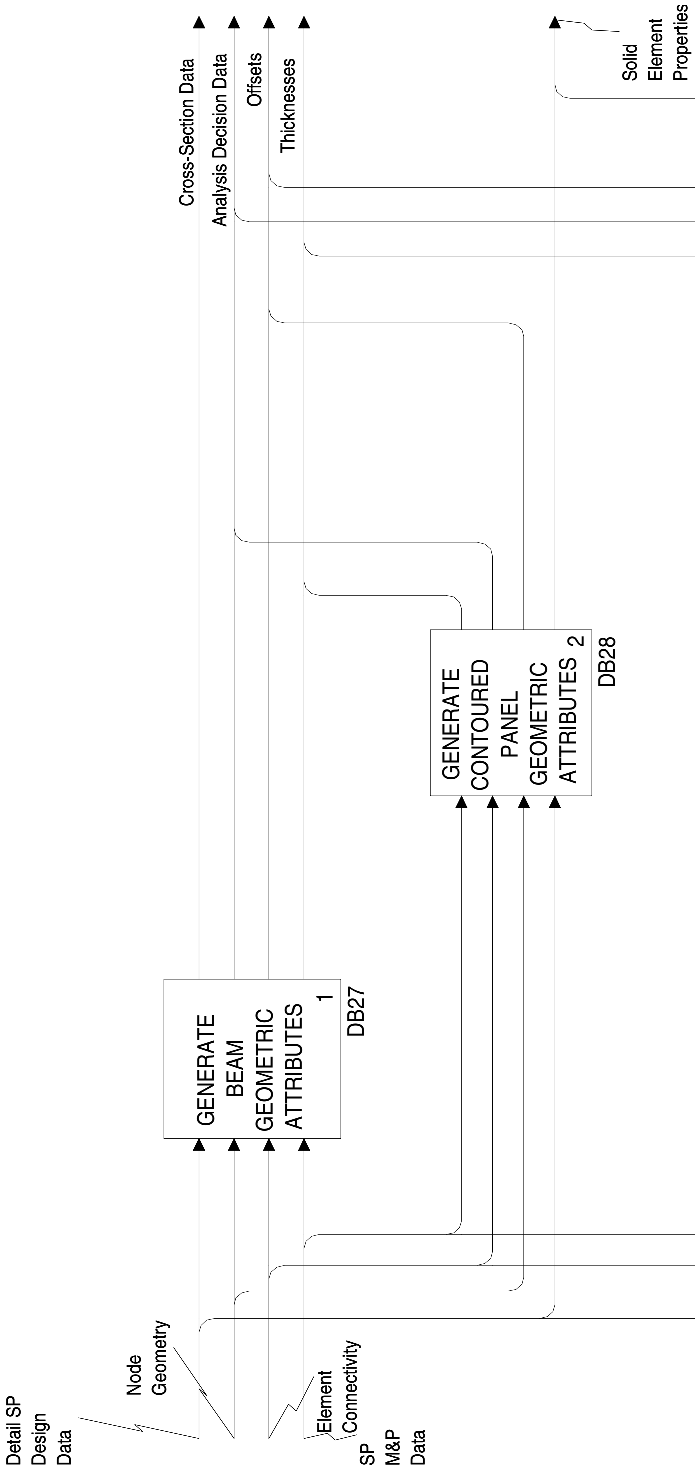

Figure F.30 - A2233352131 generate SP geometric attributes

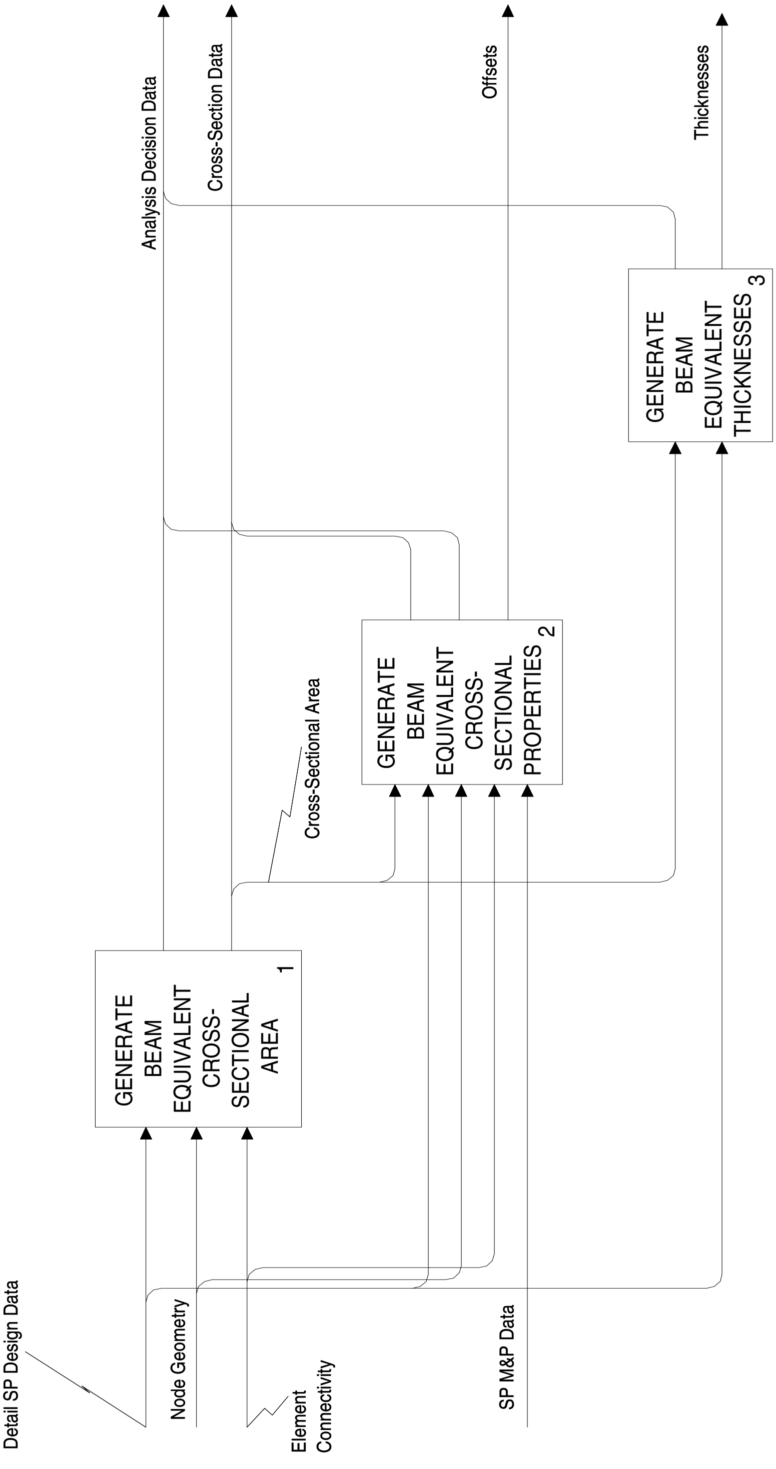

Figure F.31 - A2233521311 generate beam geometric attributes

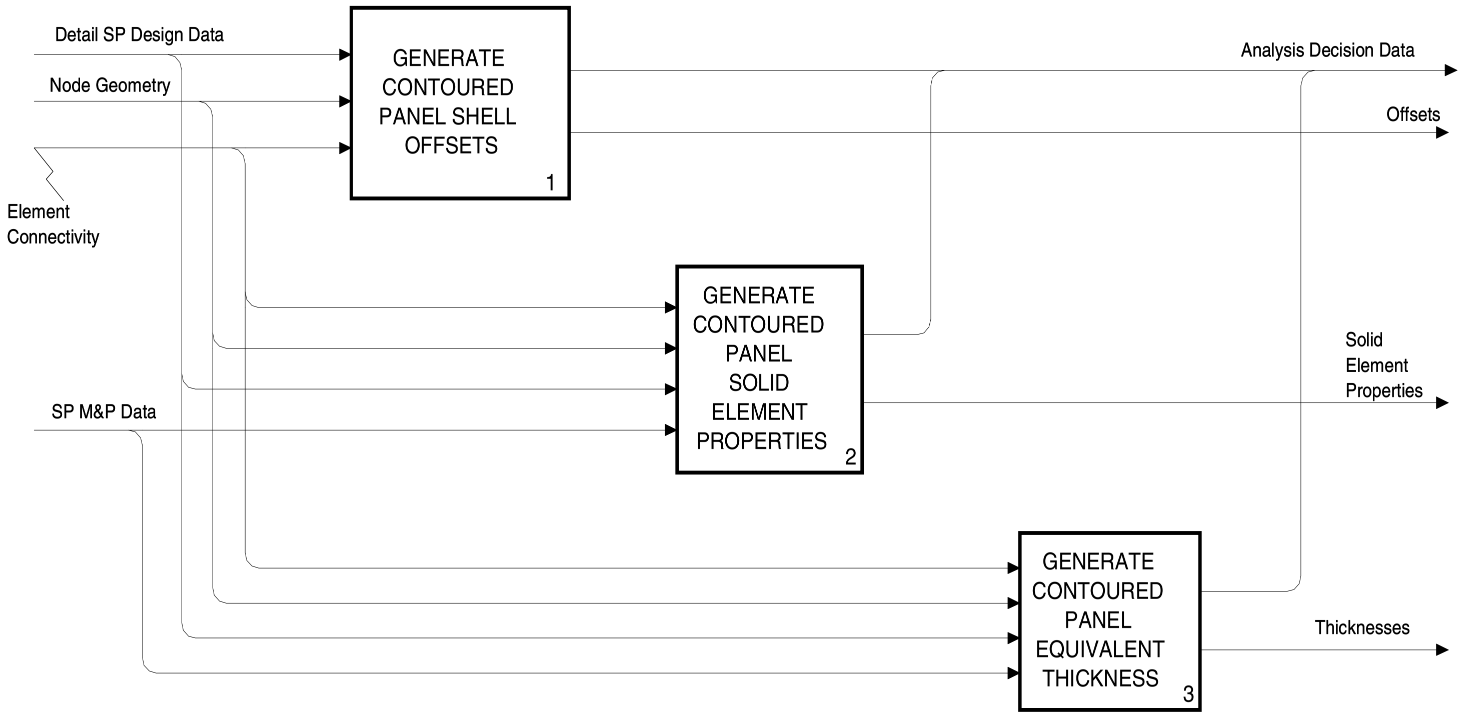

Figure F.32 - A2233521312 generate contoured panel geometric attributes

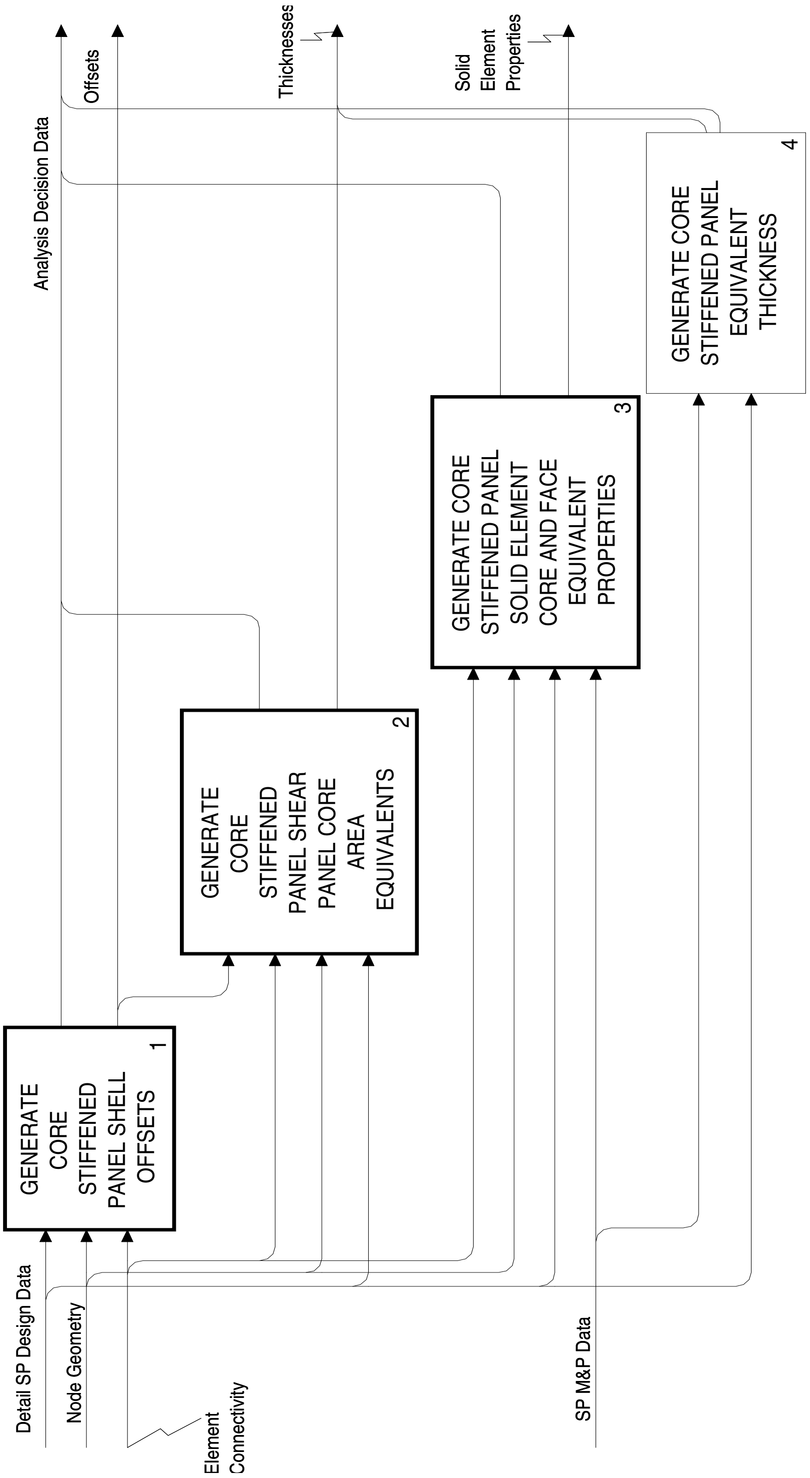

Figure F.33 - A22335521313 generate core stiffened panel geometric attributes

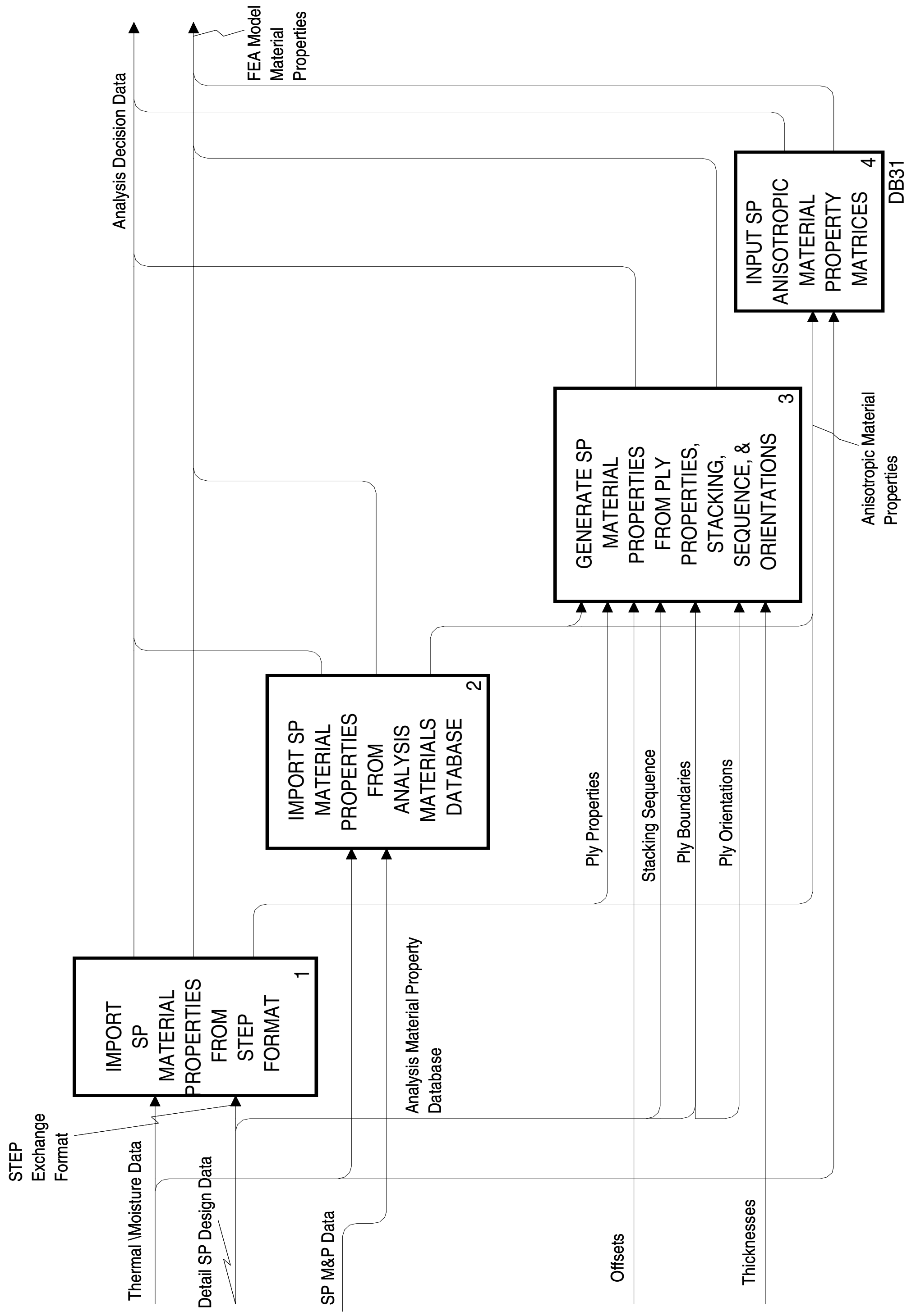

Figure F.34 - A223352133 generate/import SP material properties

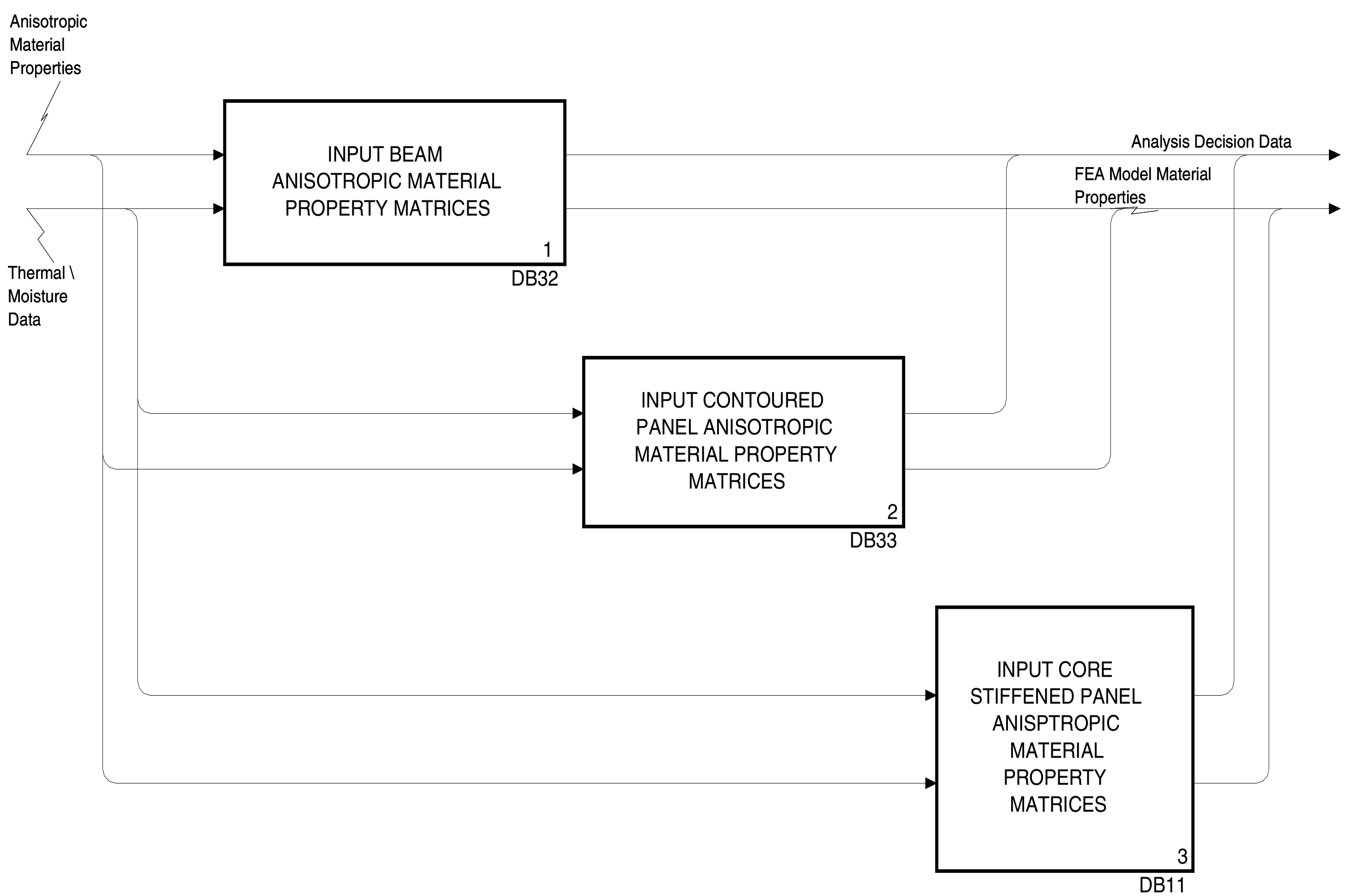

Figure F.35 - A2233521334 input SP anisotropic material property matrices

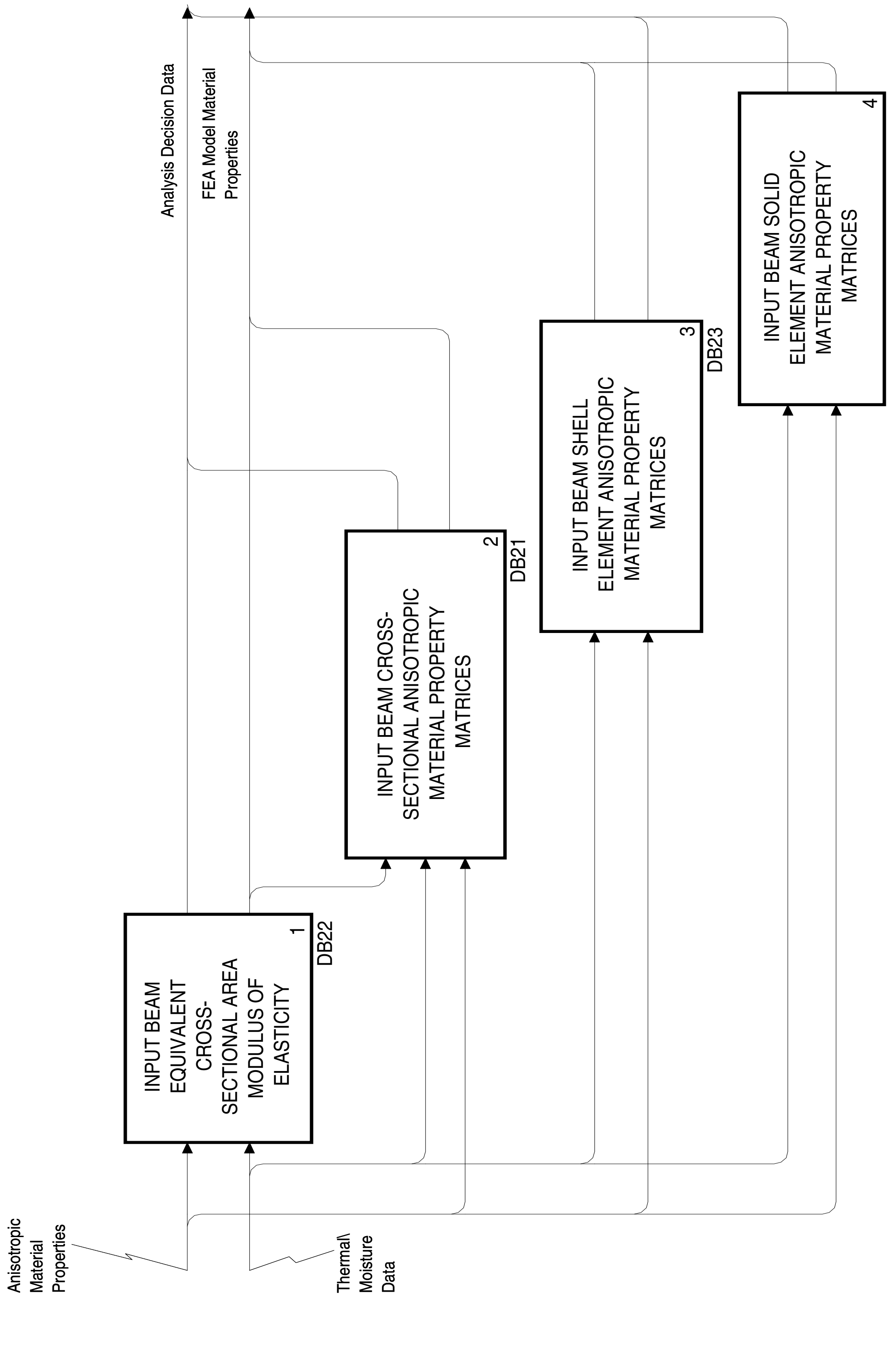

Figure F.36 - A22335213341 input beam anisotropic material property

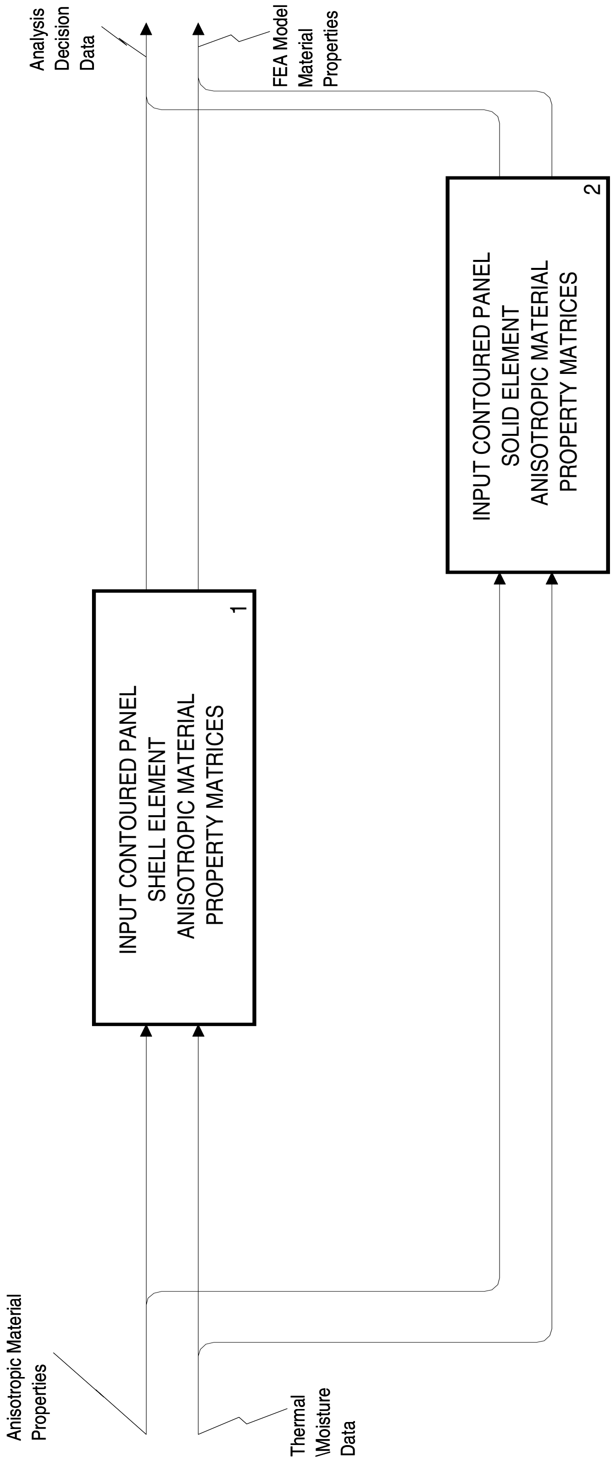

Figure F.37 - A22335213342 input contoured panels anisotropic material property

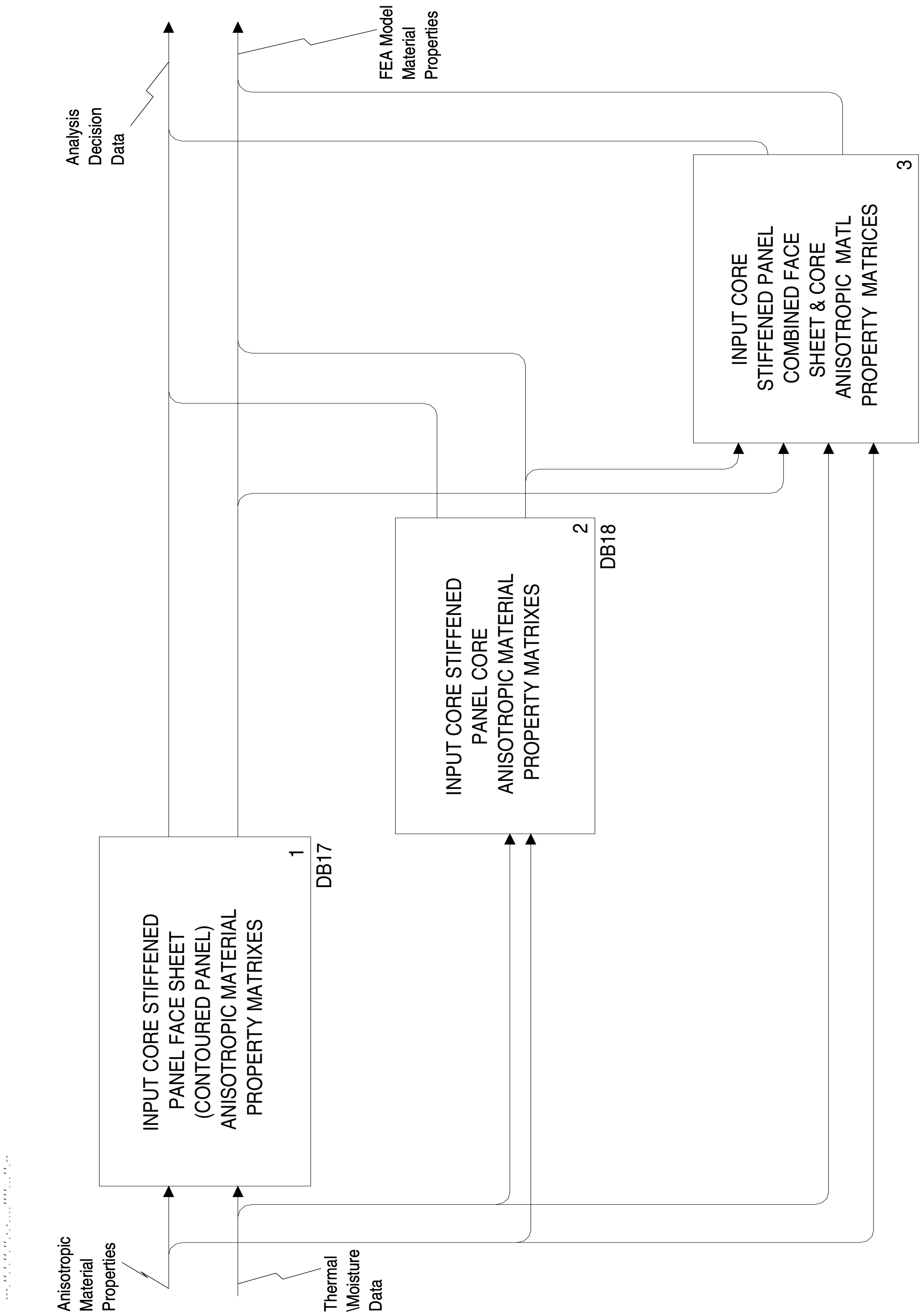

Figure F.38 - A22335123343 input core stiffened panel anisotropic material property

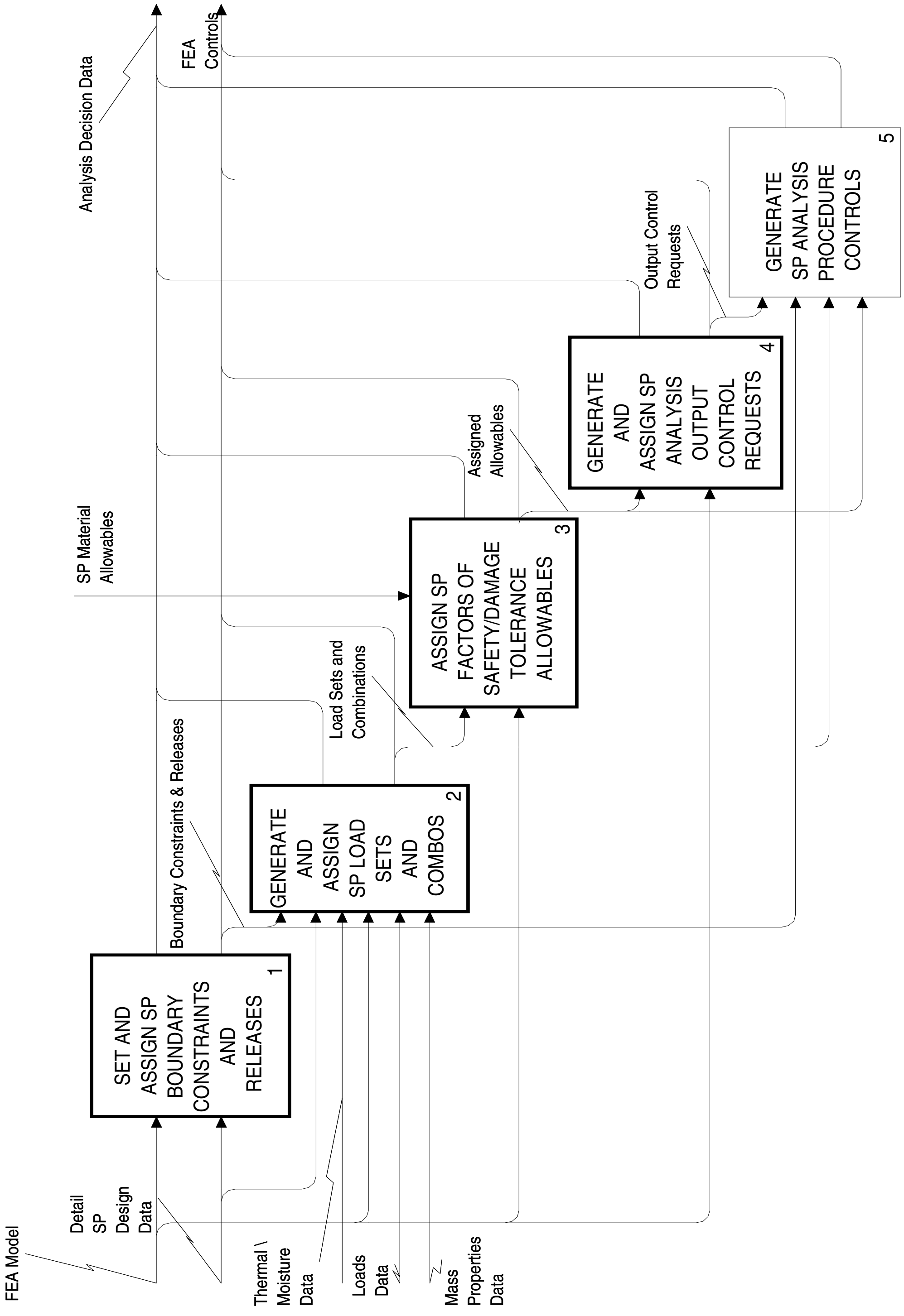

Figure F.39 - A22233522 generate SP FE analysis environment and controls

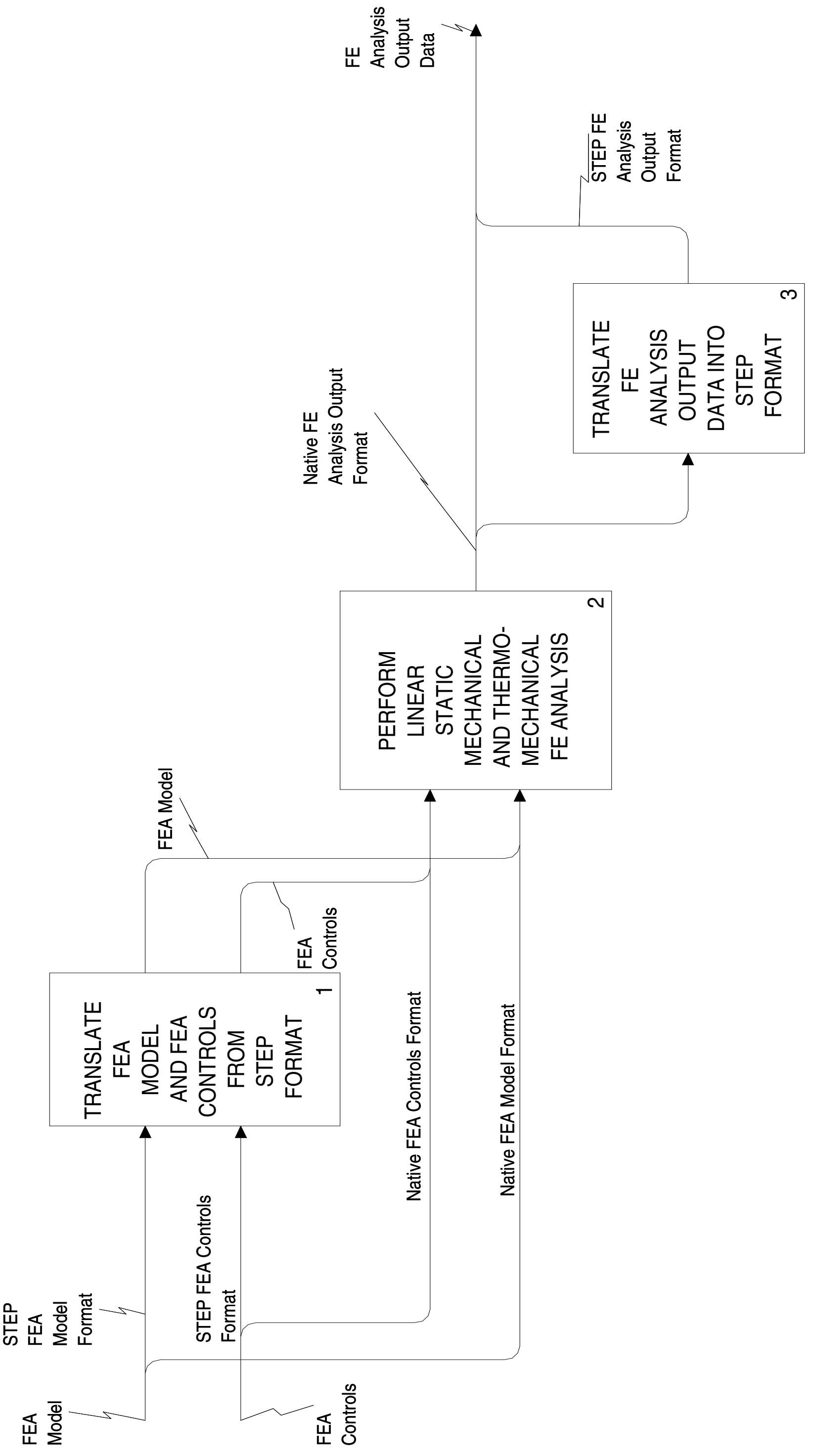

Figure F.40 - A2233523 perform SP mechanical and themo-mechanical analysis

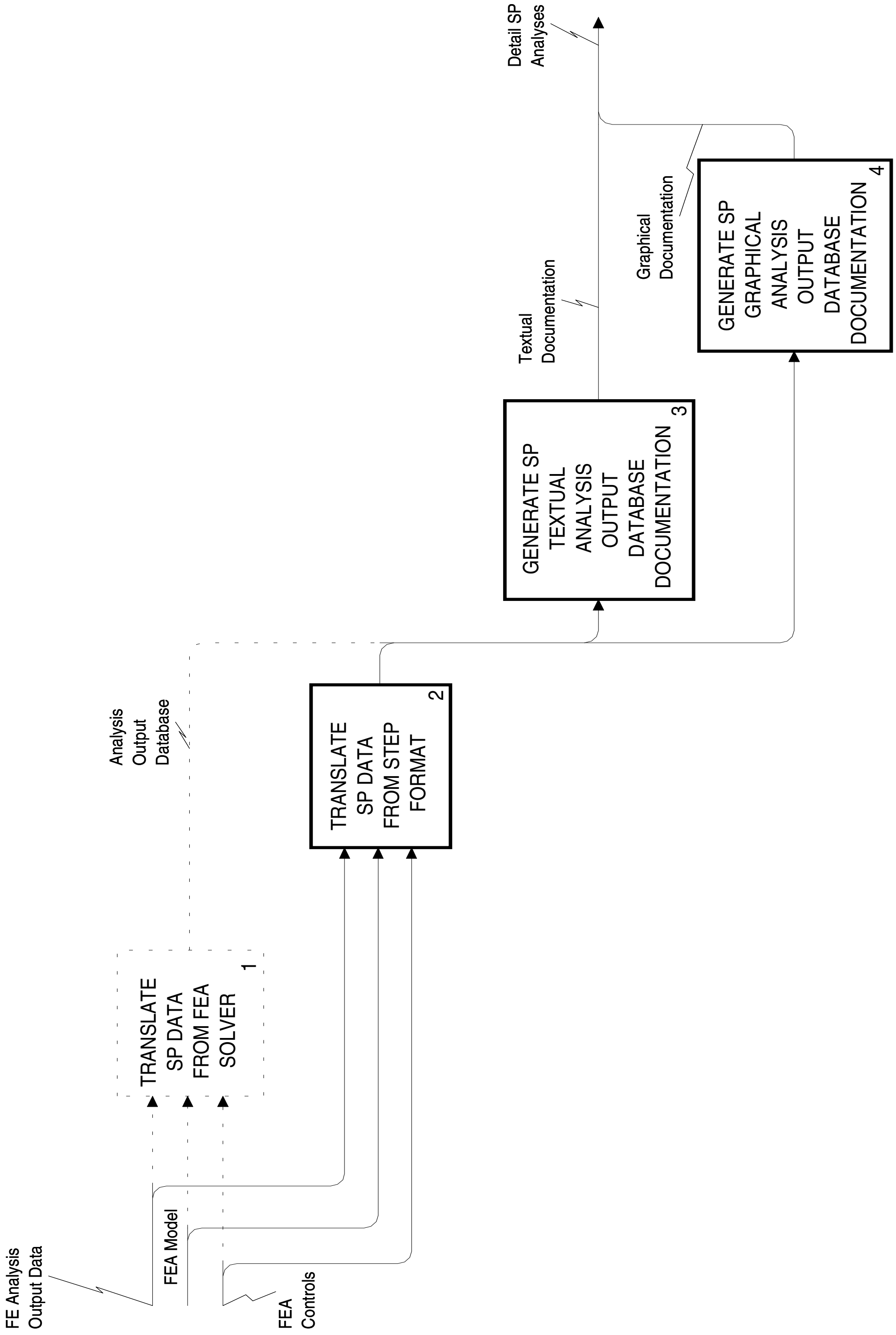

Figure F.41 - A2233524 create/document SP internal loads/stress data

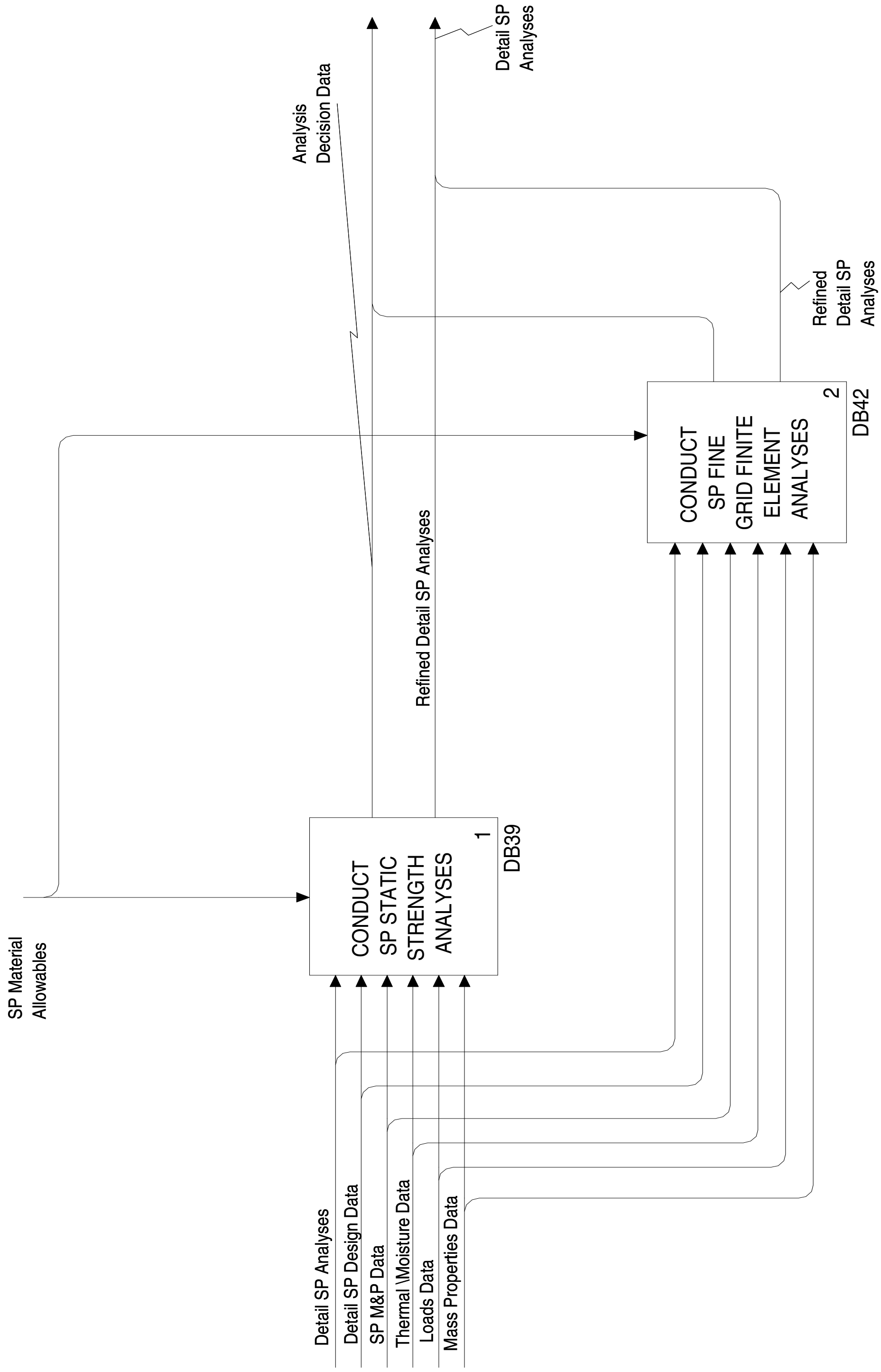

Figure F.42 - A223353 conduct SP detail stress analyses

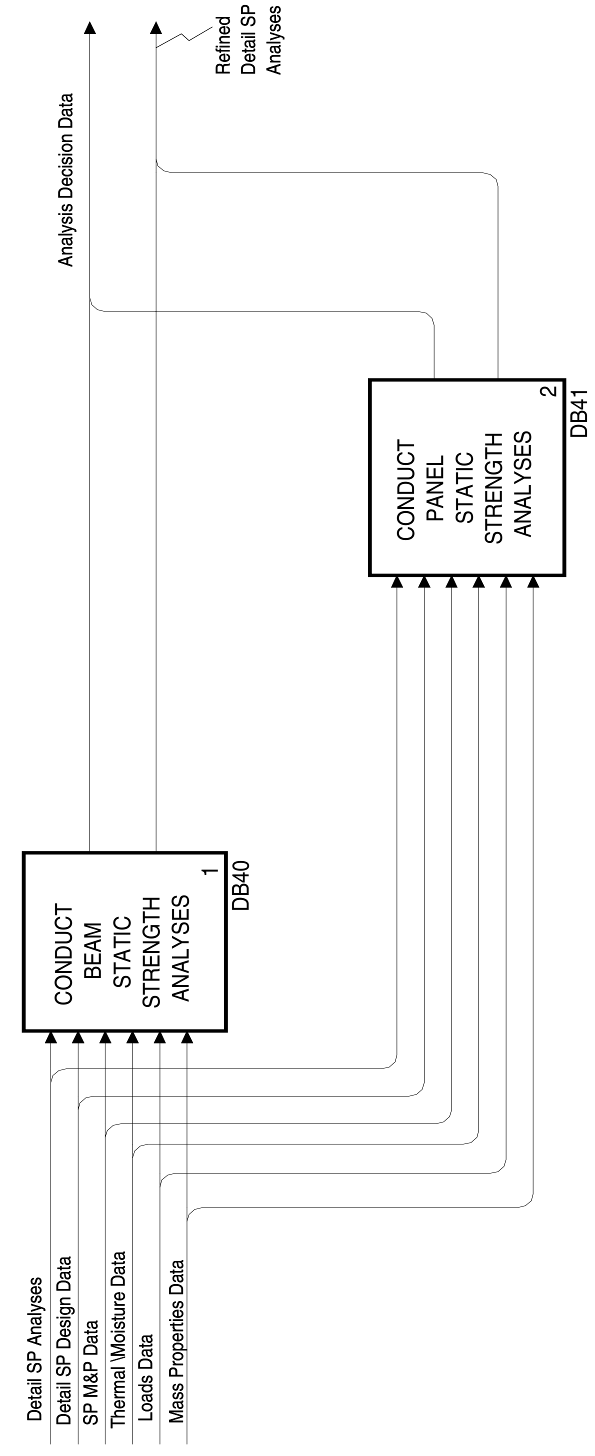

Figure F.43 - A223351 conduct SP static strength analyses

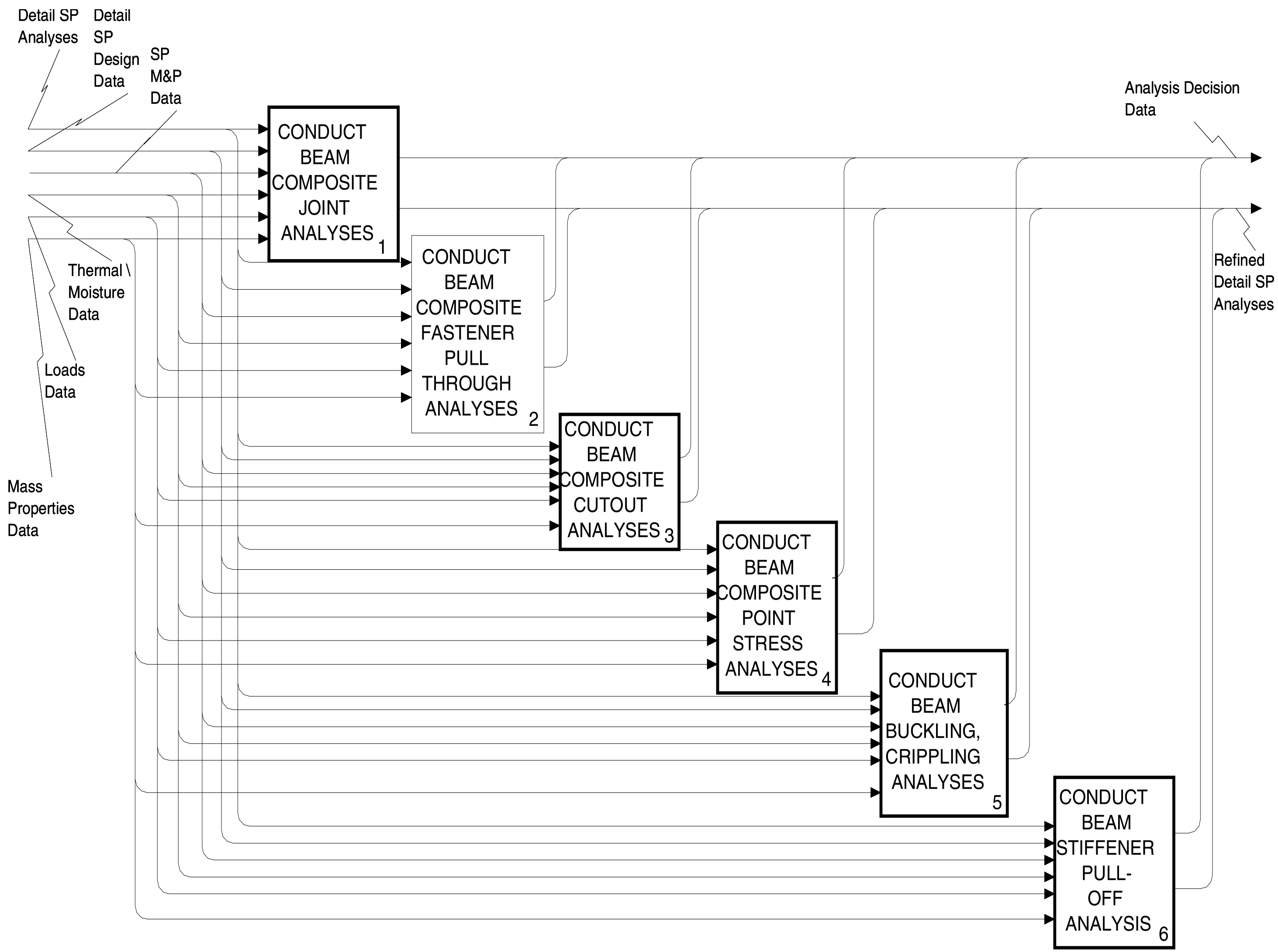

Figure F.44 - A22335311 conduct beam static strength analysis

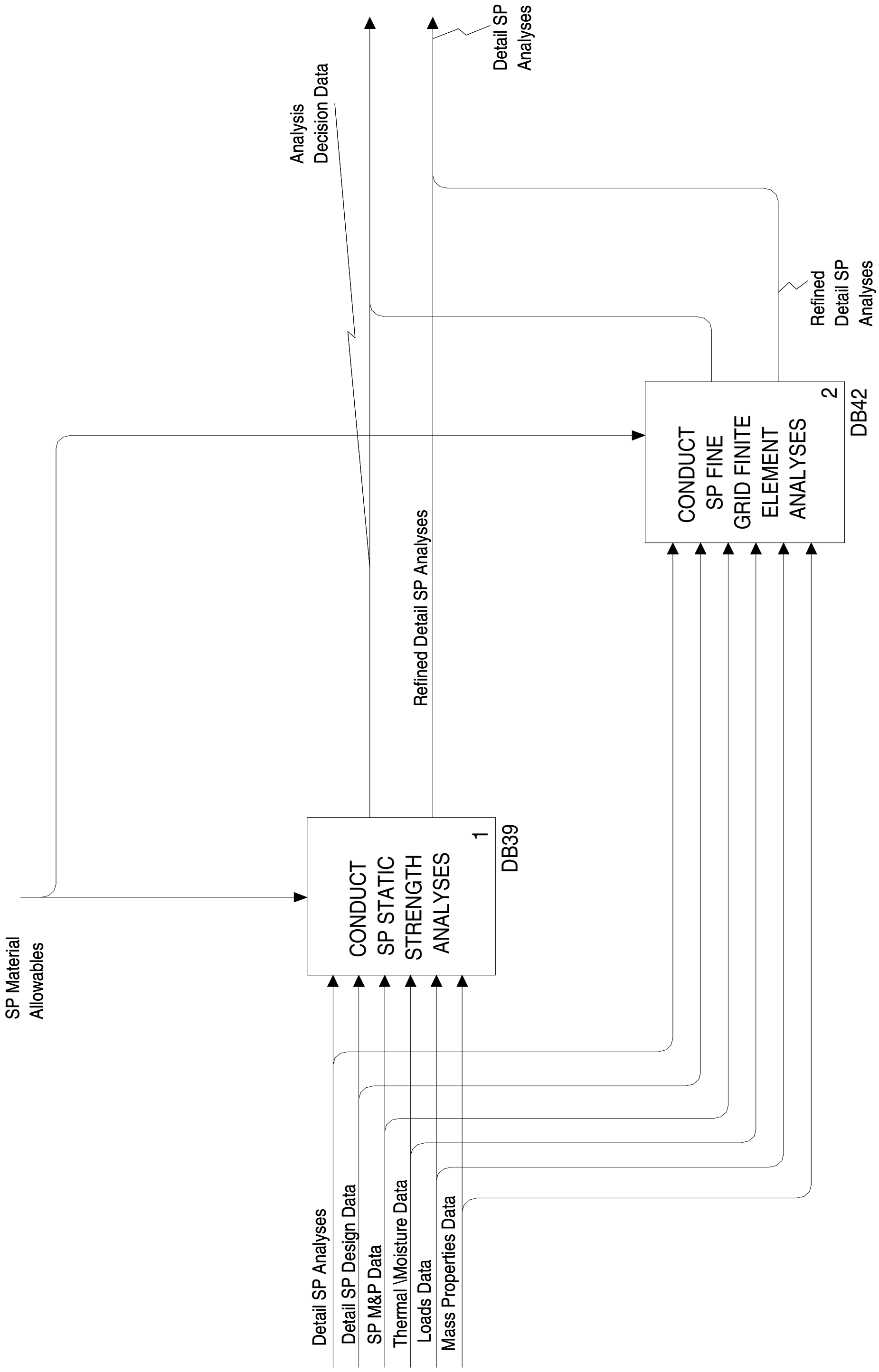

Figure F.45 - A22335312 conduct panel static strength analysis

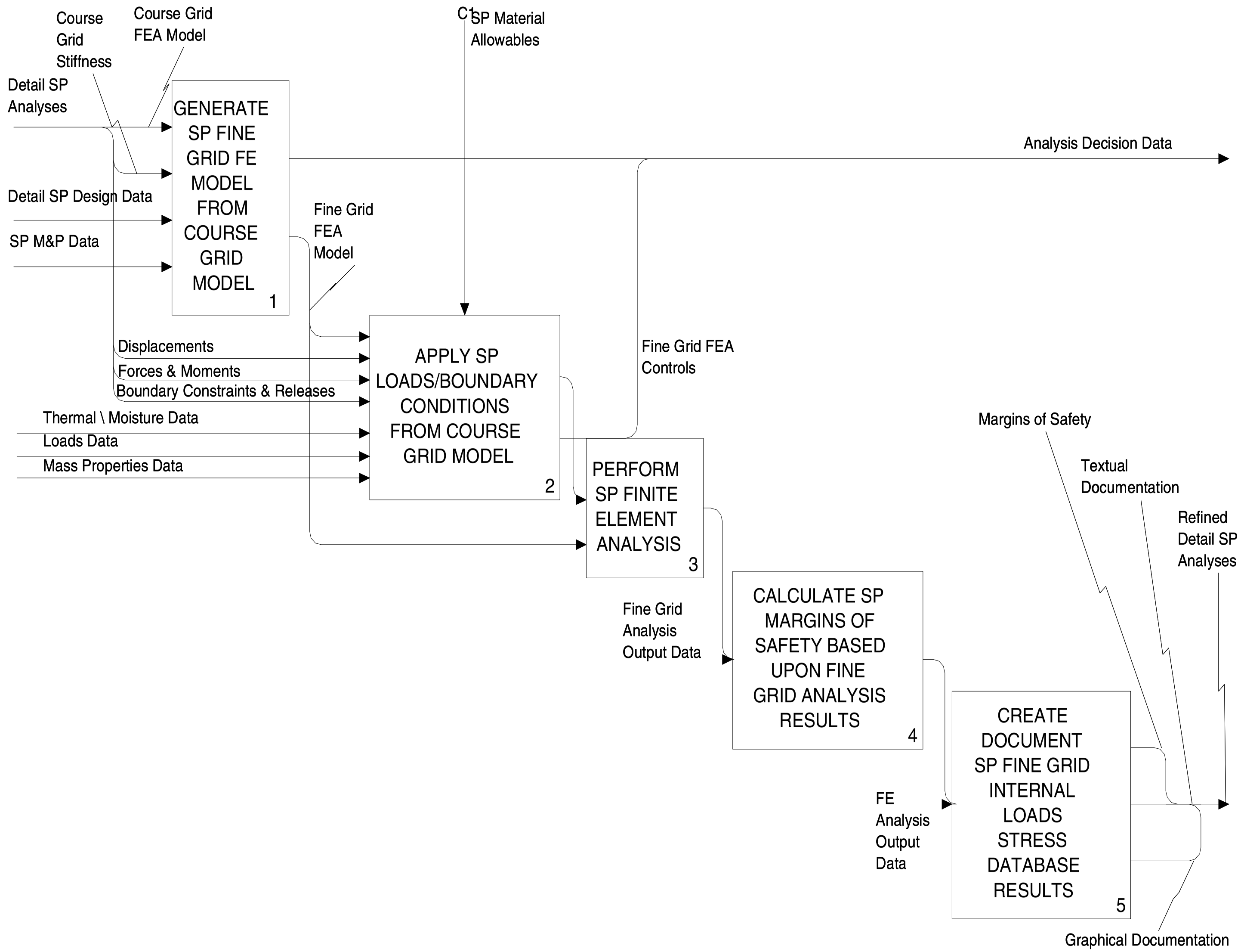

Figure F.46 - A2233532 conduct SP fine grid finite element analysis

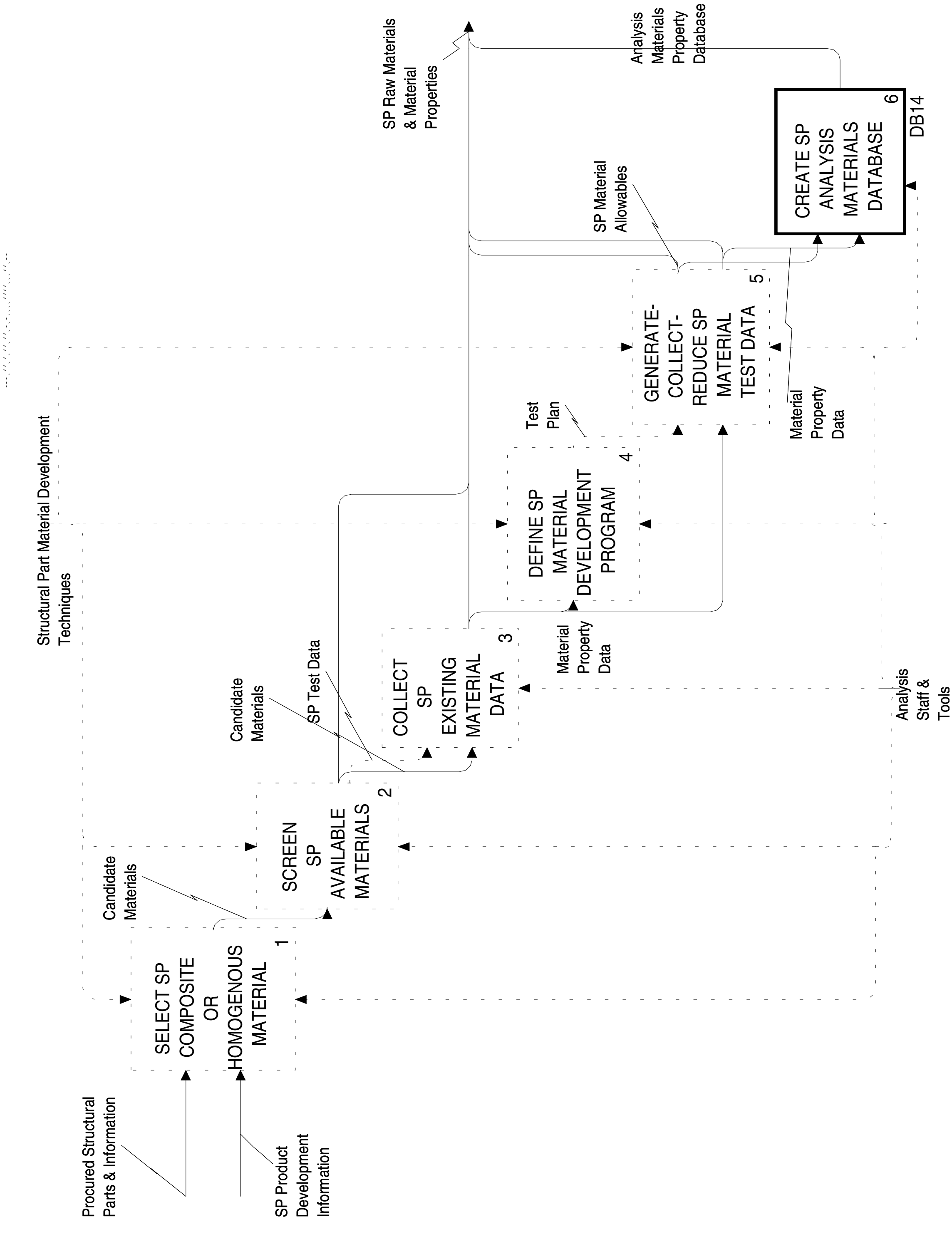

Figure F.47 - A4 develop and provide SP materials

Annex G
(informative)

Application reference model

This annex provides the application reference model for this part of ISO 10303 and is given in Figures G.1 through G.15. 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.

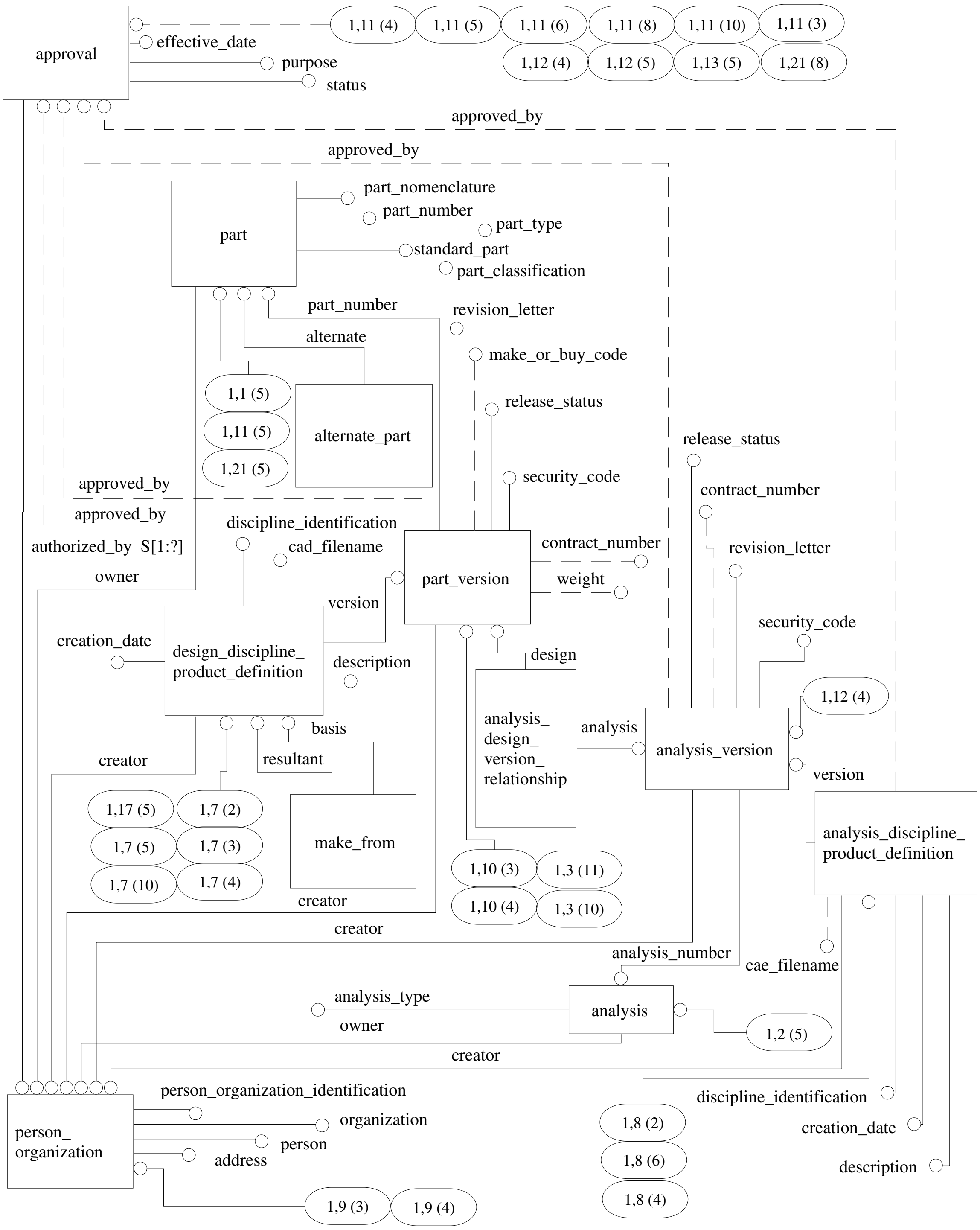

Figure G.1 - ARM EXPRESS-G diagram 1 of 15

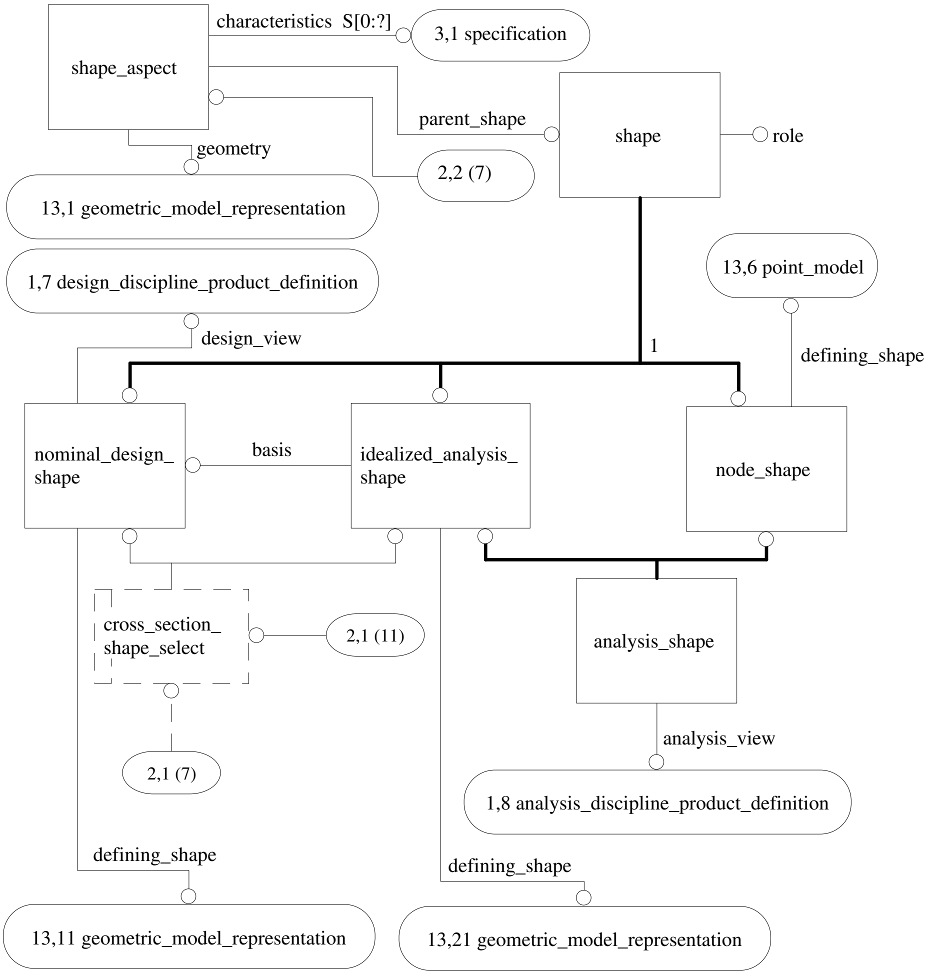

Figure G.2 - ARM EXPRESS-G diagram 2 of 15

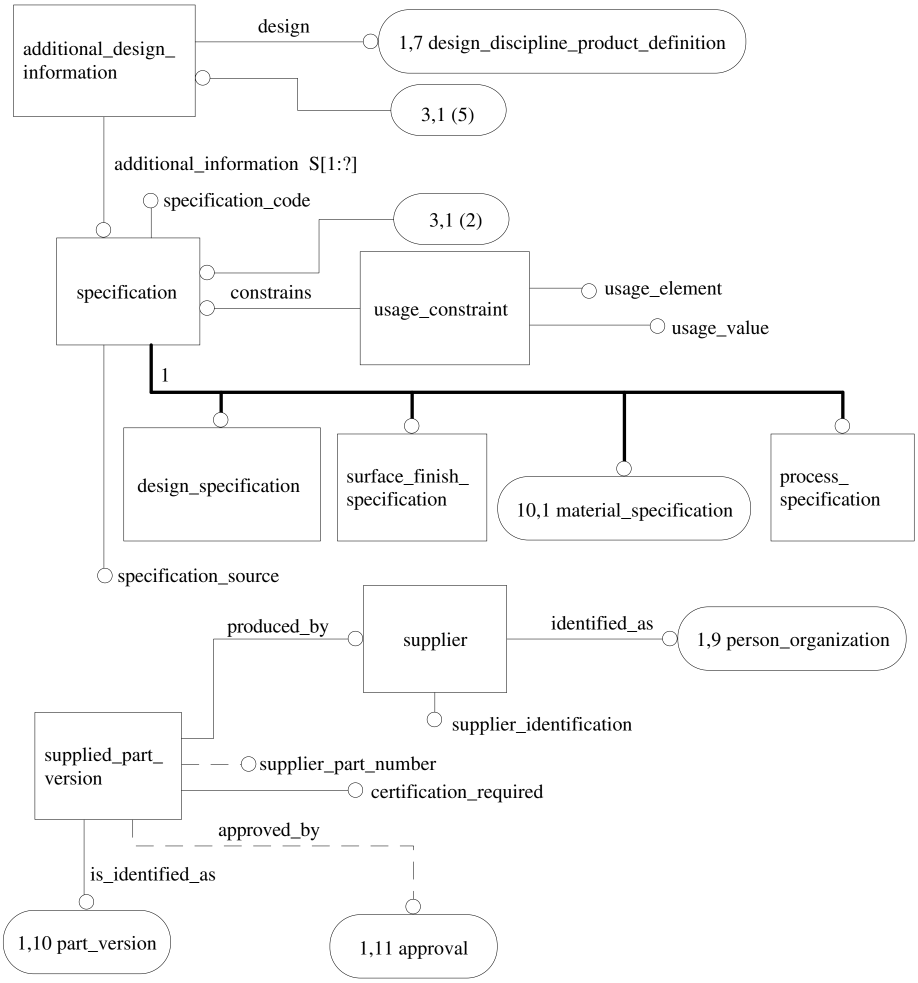

Figure G.3 - ARM EXPRESS-G diagram 3 of 15

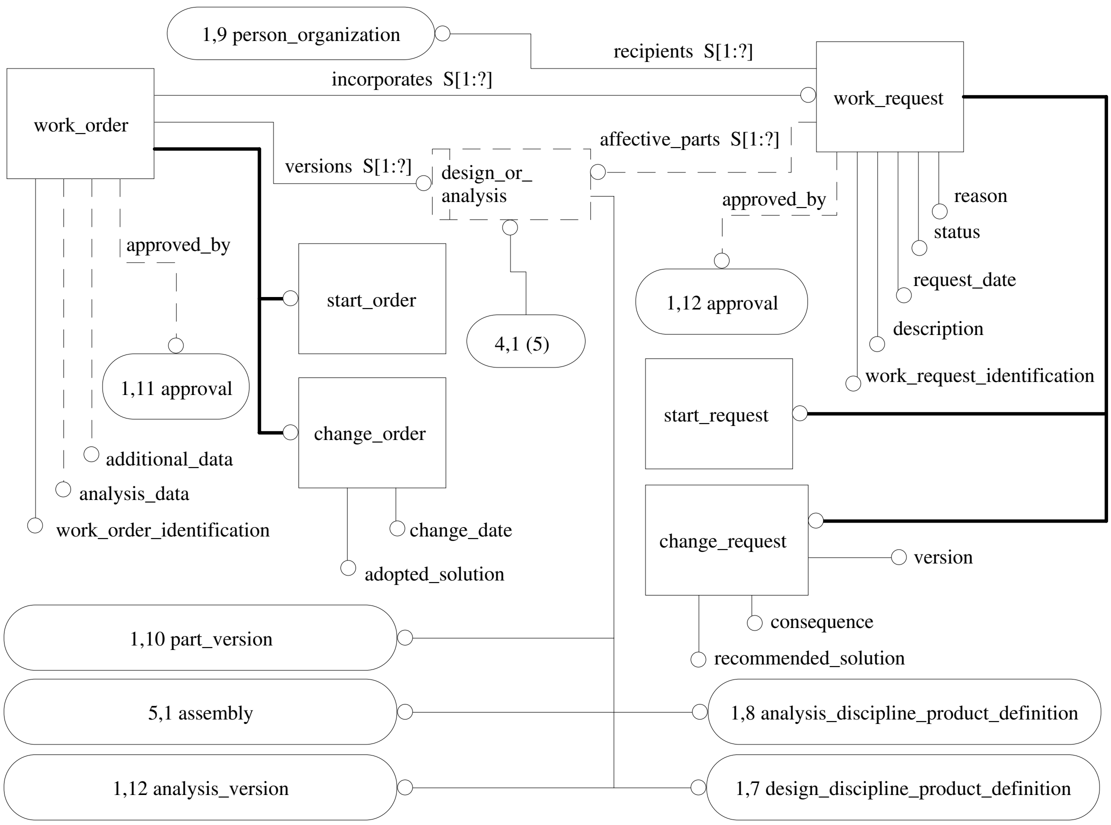

Figure G.4 - ARM EXPRESS- G diagram 4 of 15

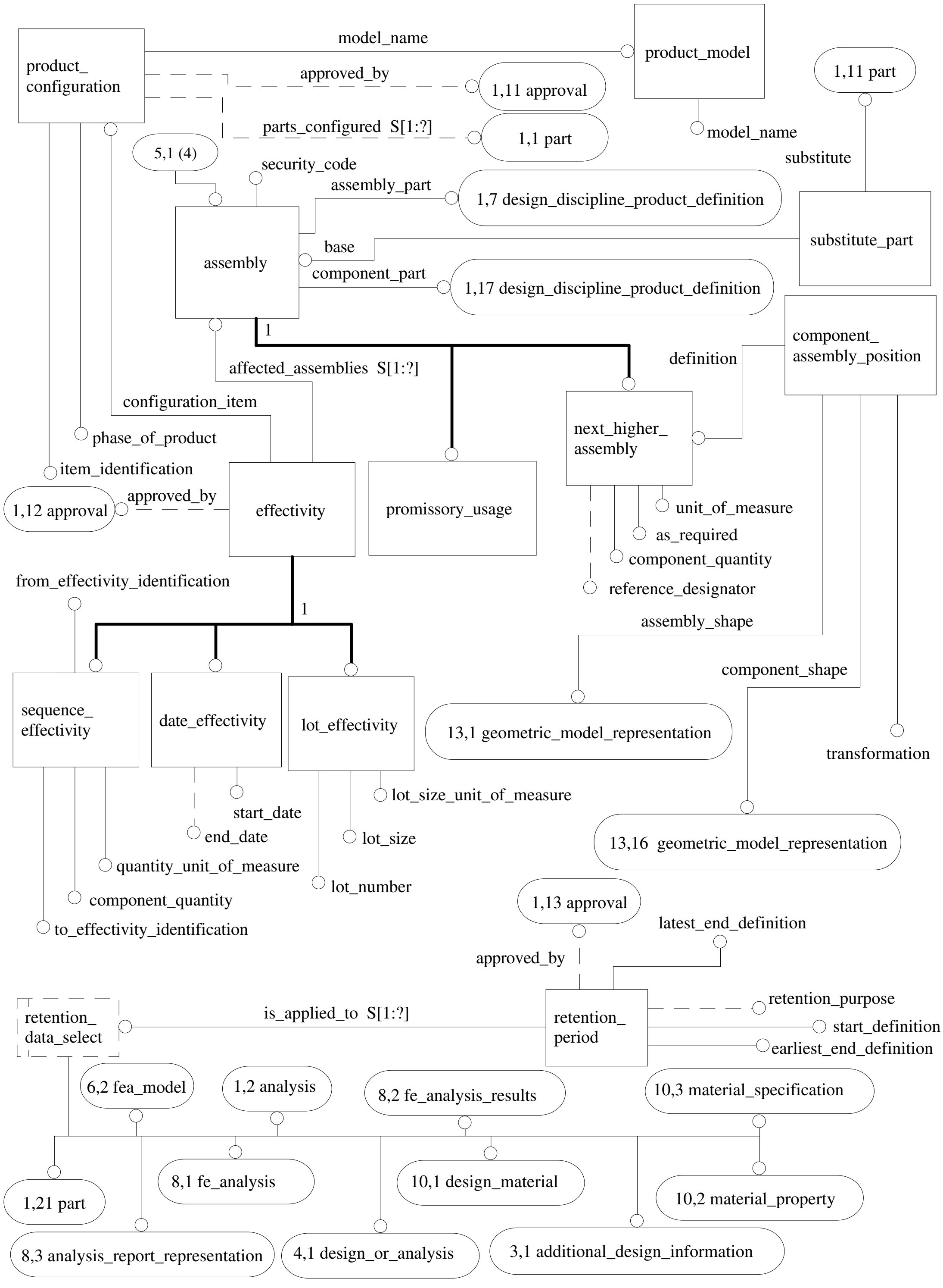

Figure G.5 - ARM EXPRESS-G diagram 5 of 15

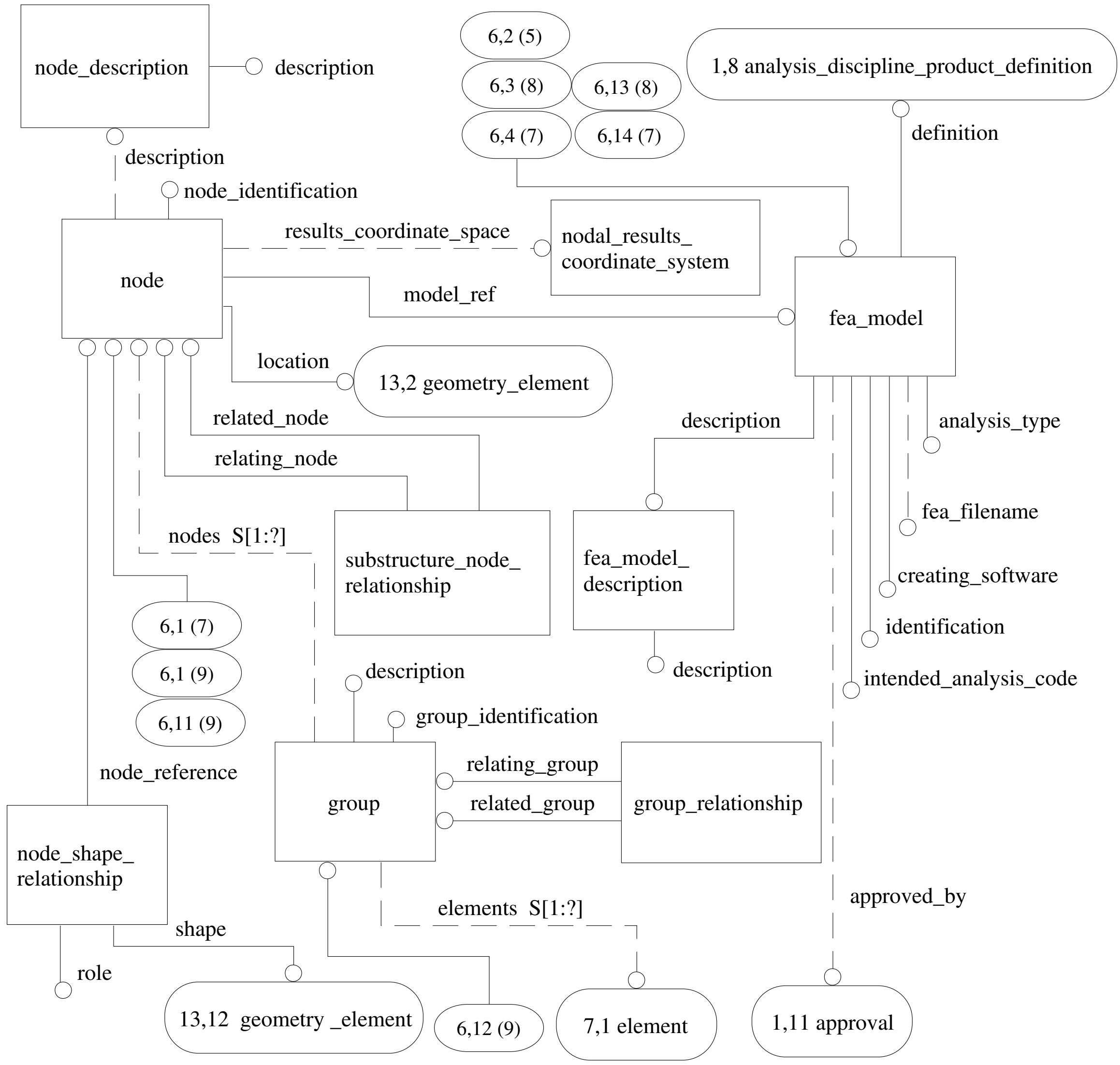

Figure G.6 - ARM EXPRESS-G diagram 6 of 15

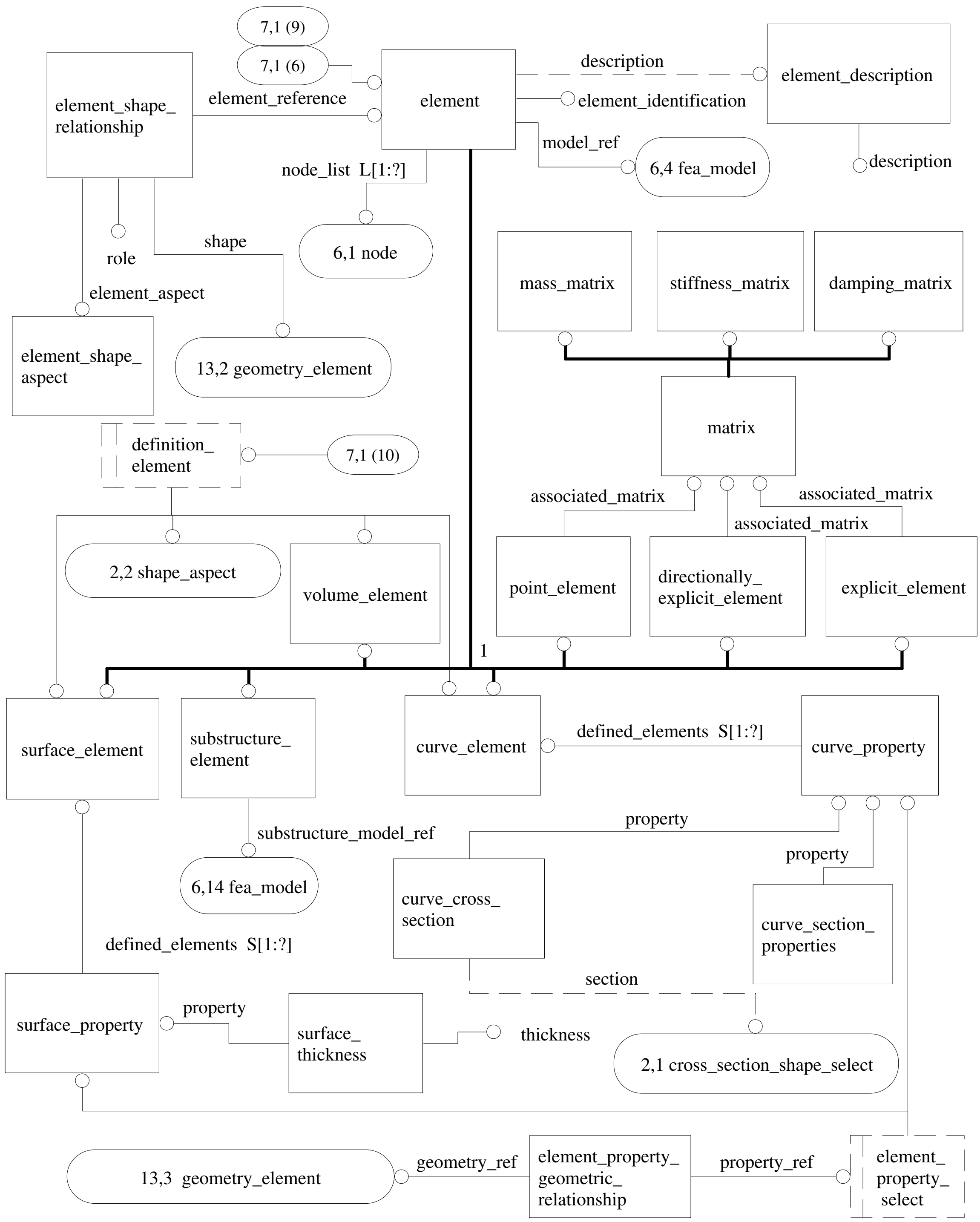

Figure G.7 - ARM EXPRESS-G diagram 7 of 15

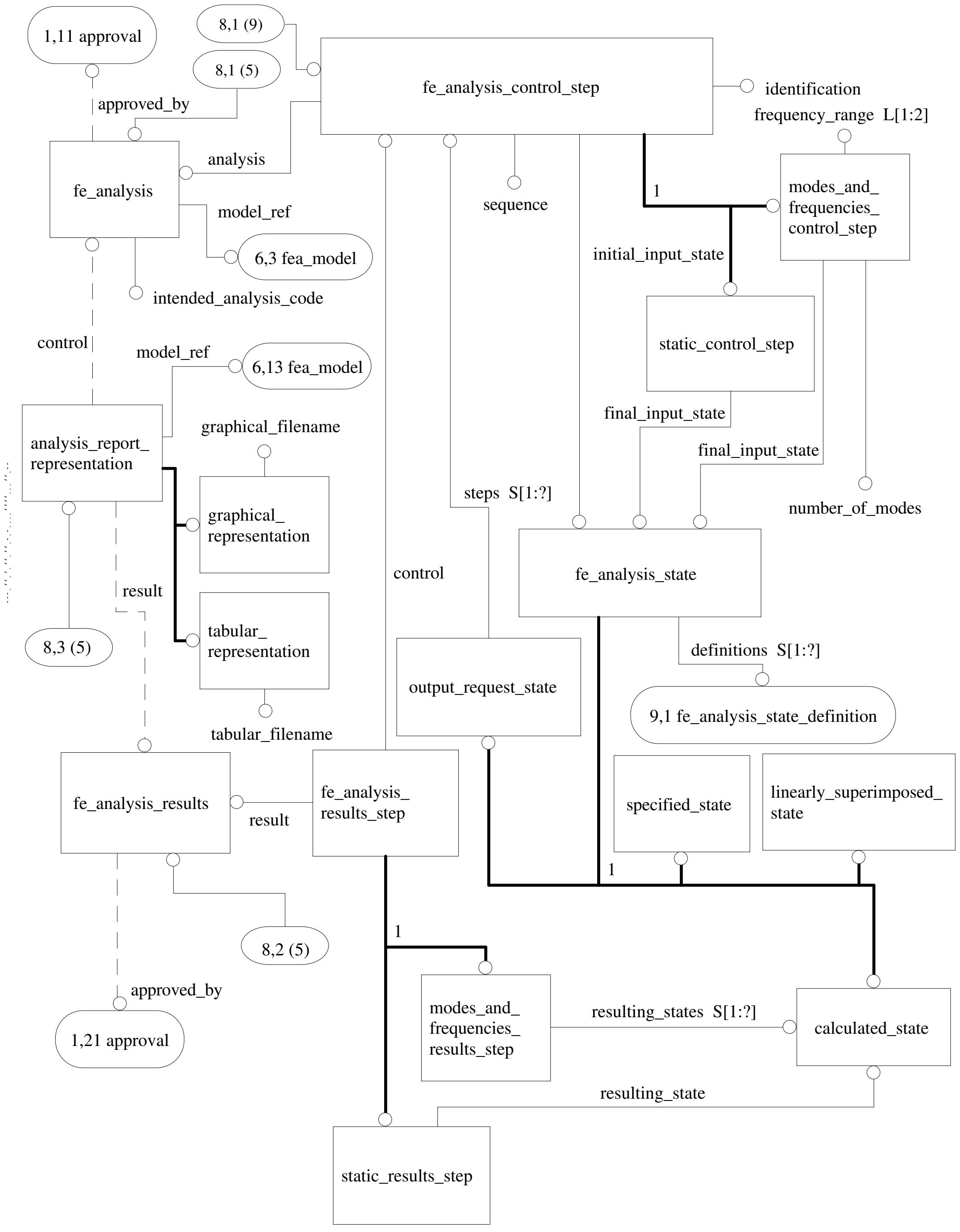

Figure G.8 - ARM EXPRESS-G diagram 8 of 15

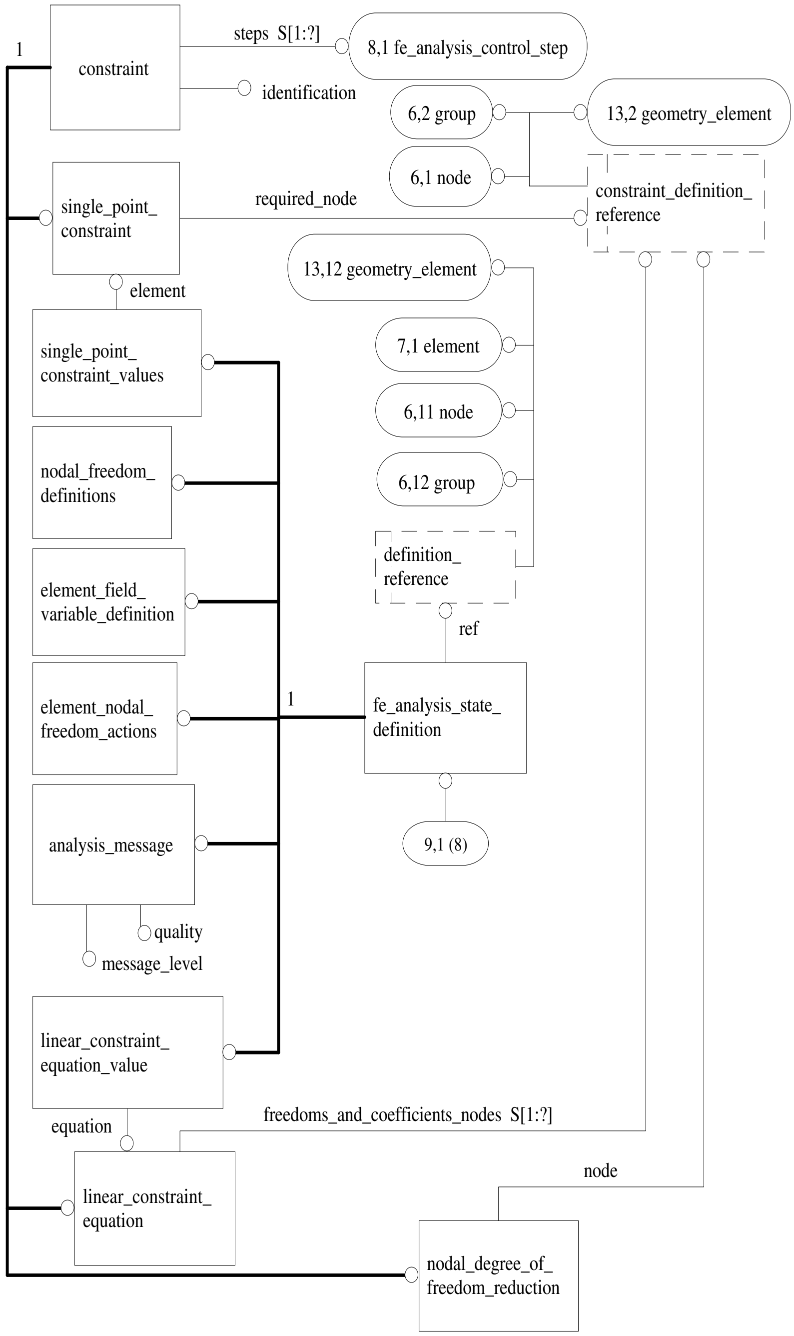

Figure G.9 - ARM EXPRESS-G diagram 9 of 15

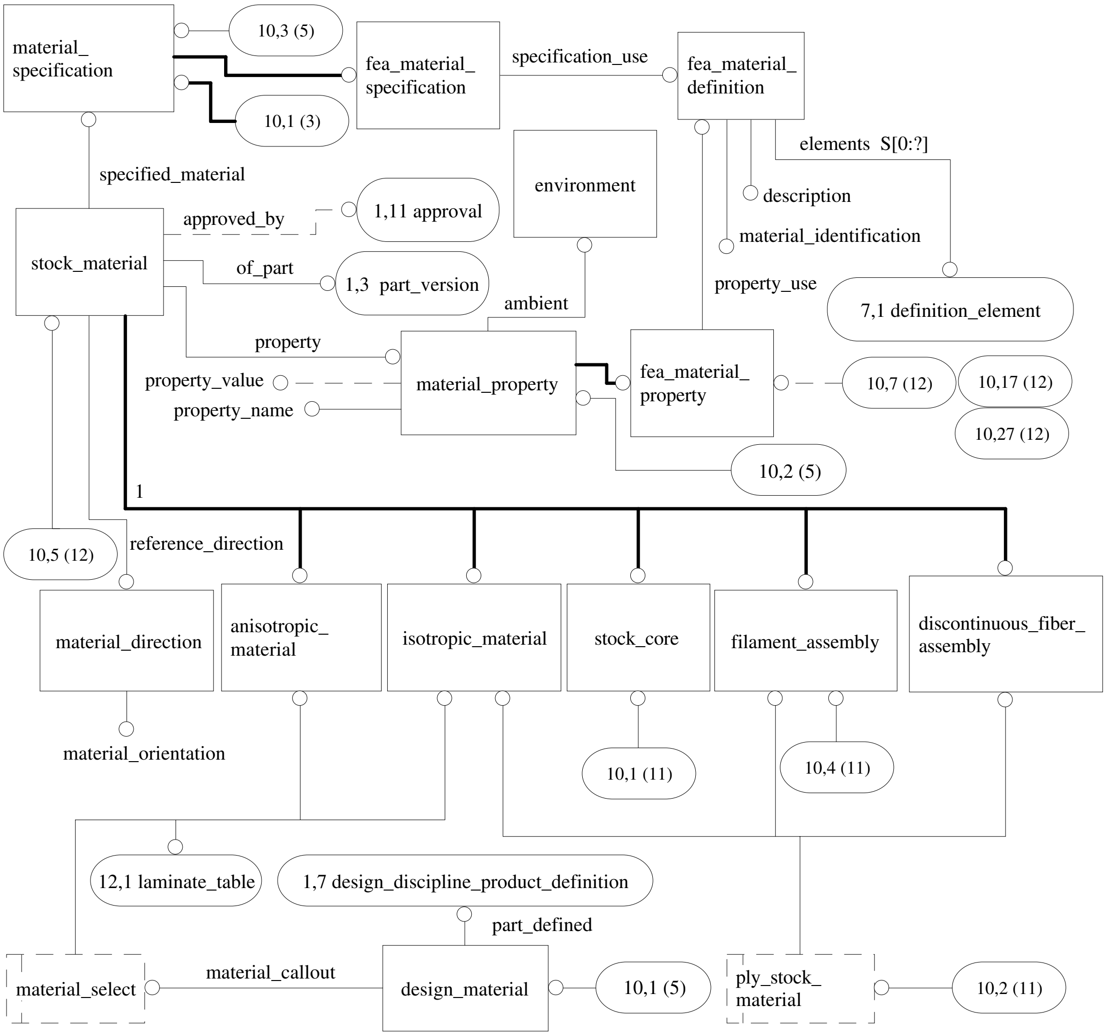

Figure G.10 - ARM EXPRESS-G diagram 10 of 15

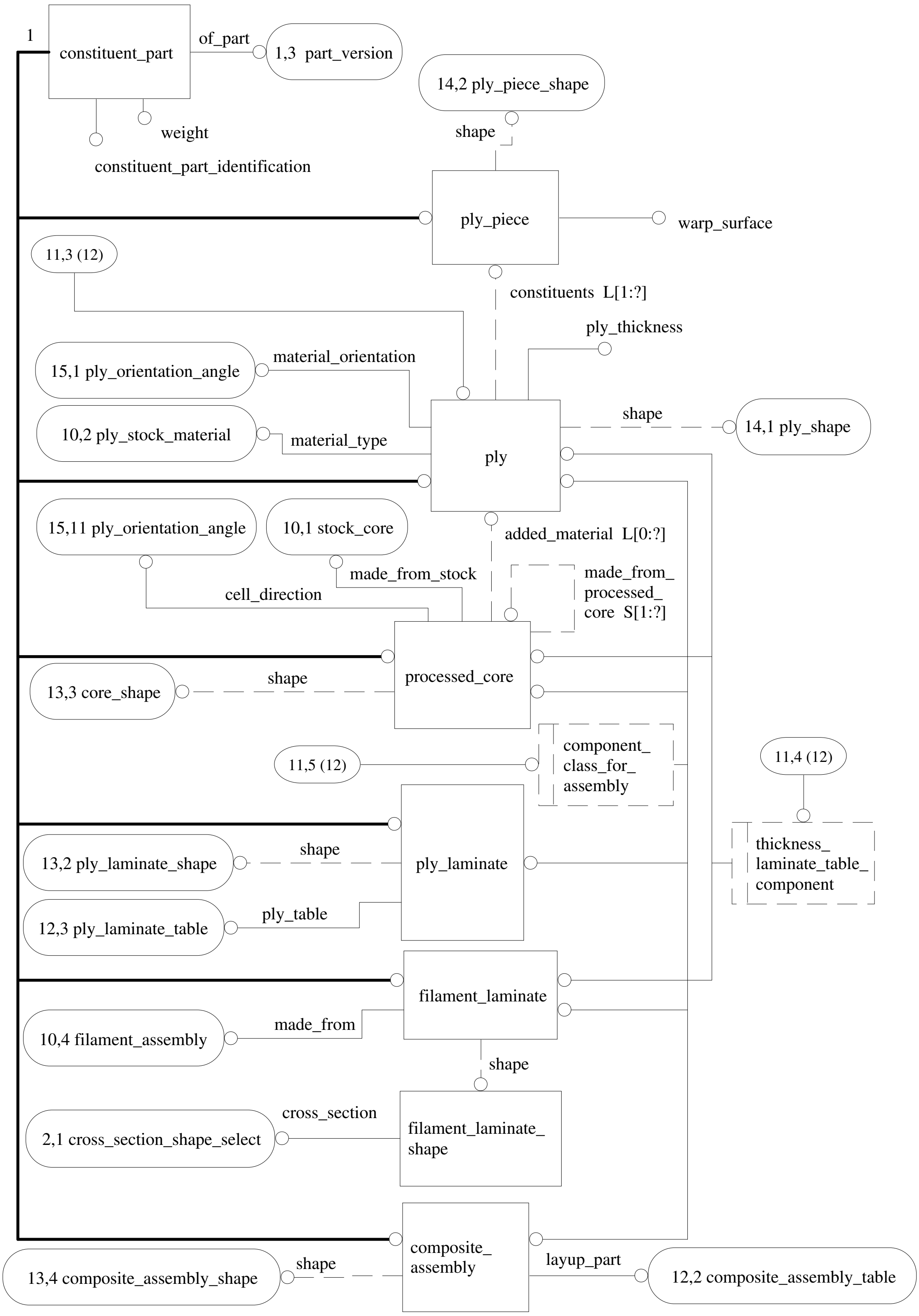

Figure G.11 - ARM EXPRESS-G diagram 11 of 15

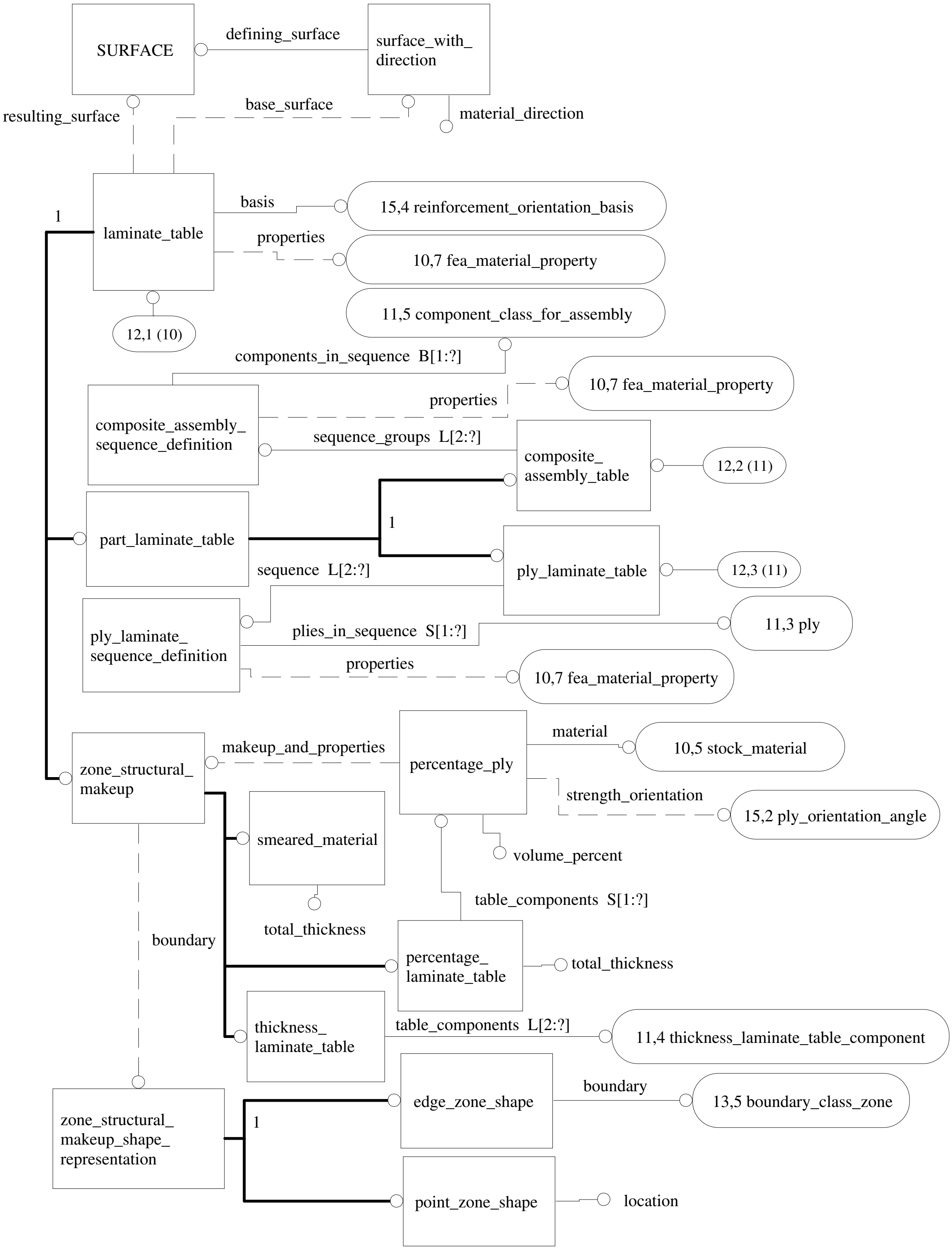

Figure G.12 - ARM EXPRESS-G diagram 12 of 15

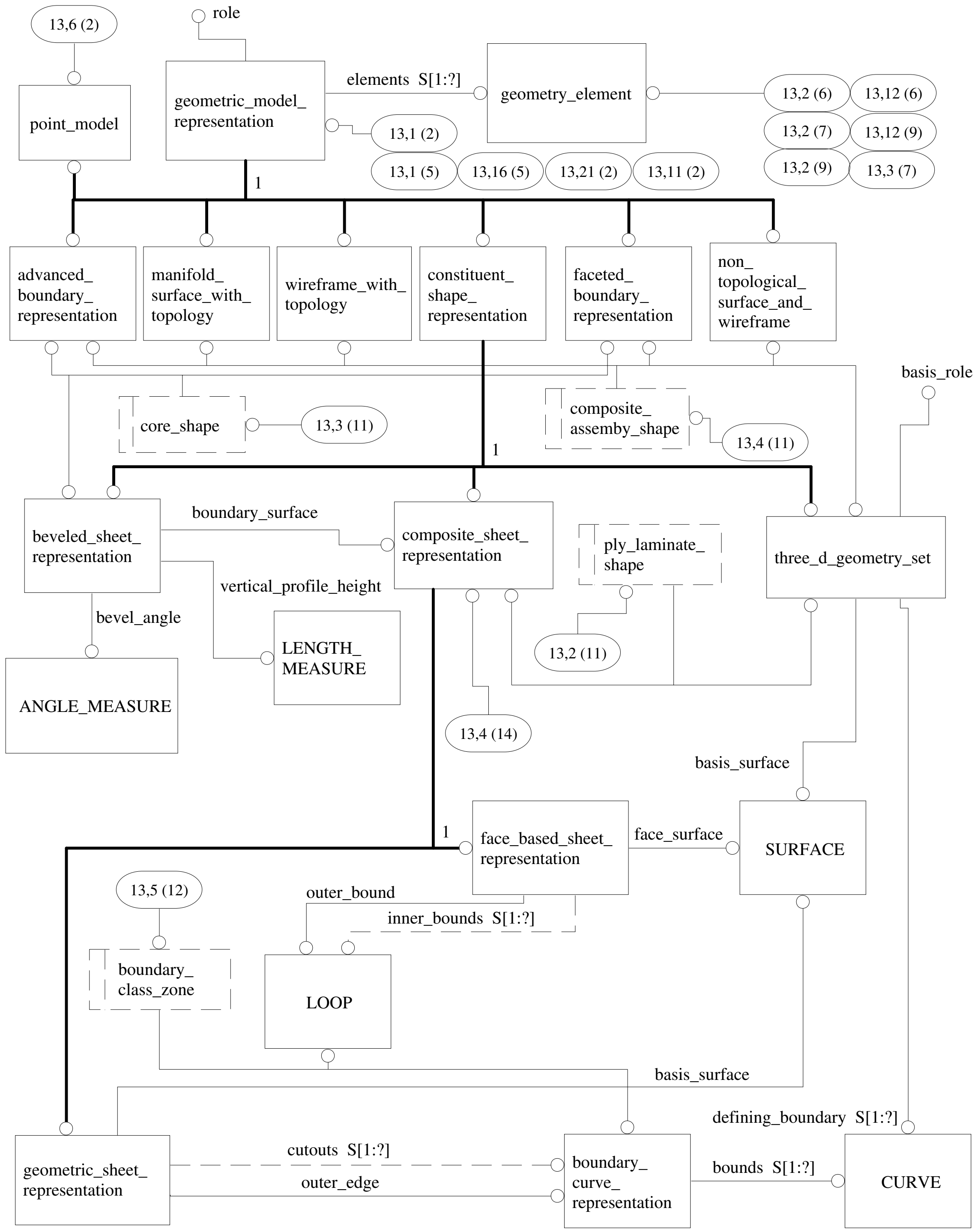

Figure G.13 - ARM EXPRESS-G diagram 13 of 15

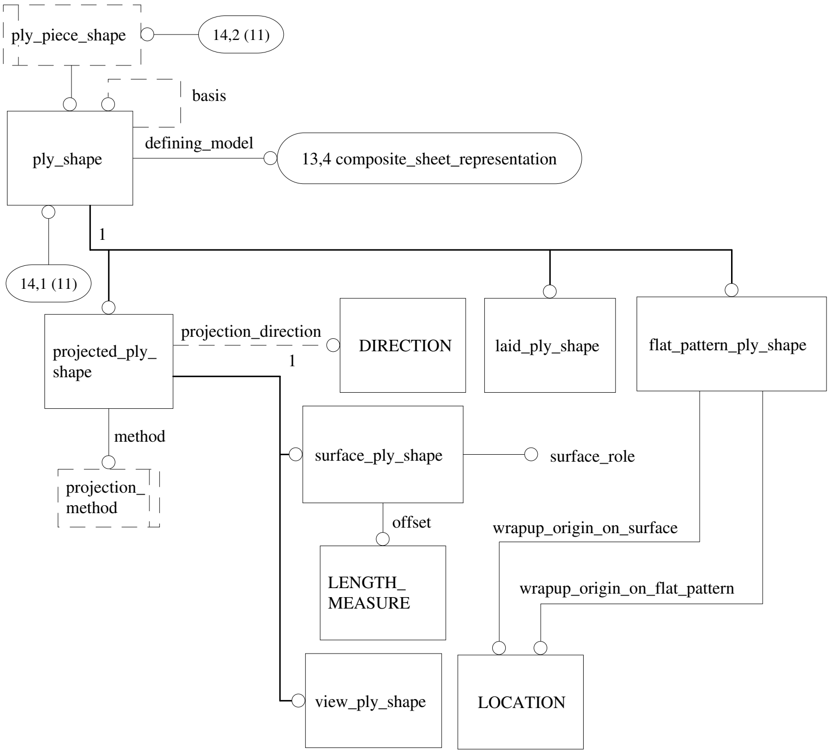

Figure G.14 - ARM EXPRESS-G diagram 14 of 15

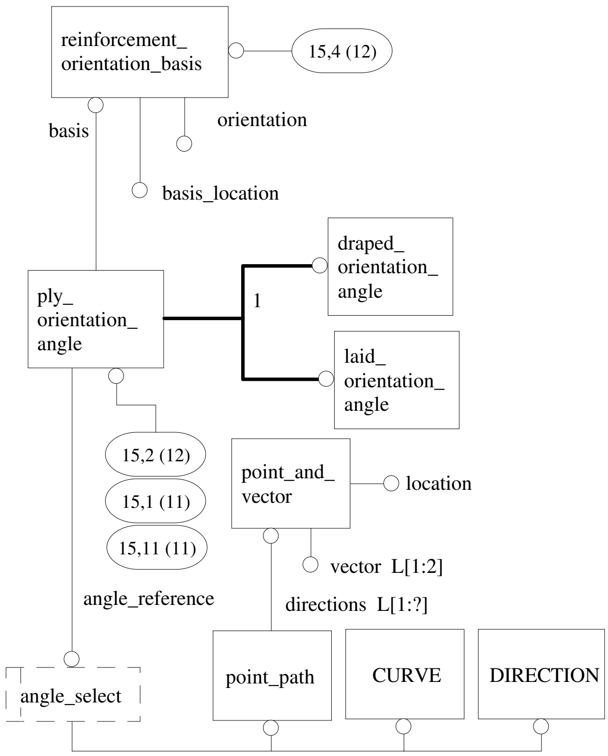

Figure G.15 - ARM EXPRESS-G diagram 15 of 15

Annex H
(informative)

AIM EXPRESS-G

Figures H.1 through H.94 correspond to the AIM EXPRESS expanded listing given in annex A. The diagrams use the EXPRESS-G graphical notation for the EXPRESS language. EXPRESS-G is defined in annex D of ISO 10303-11.

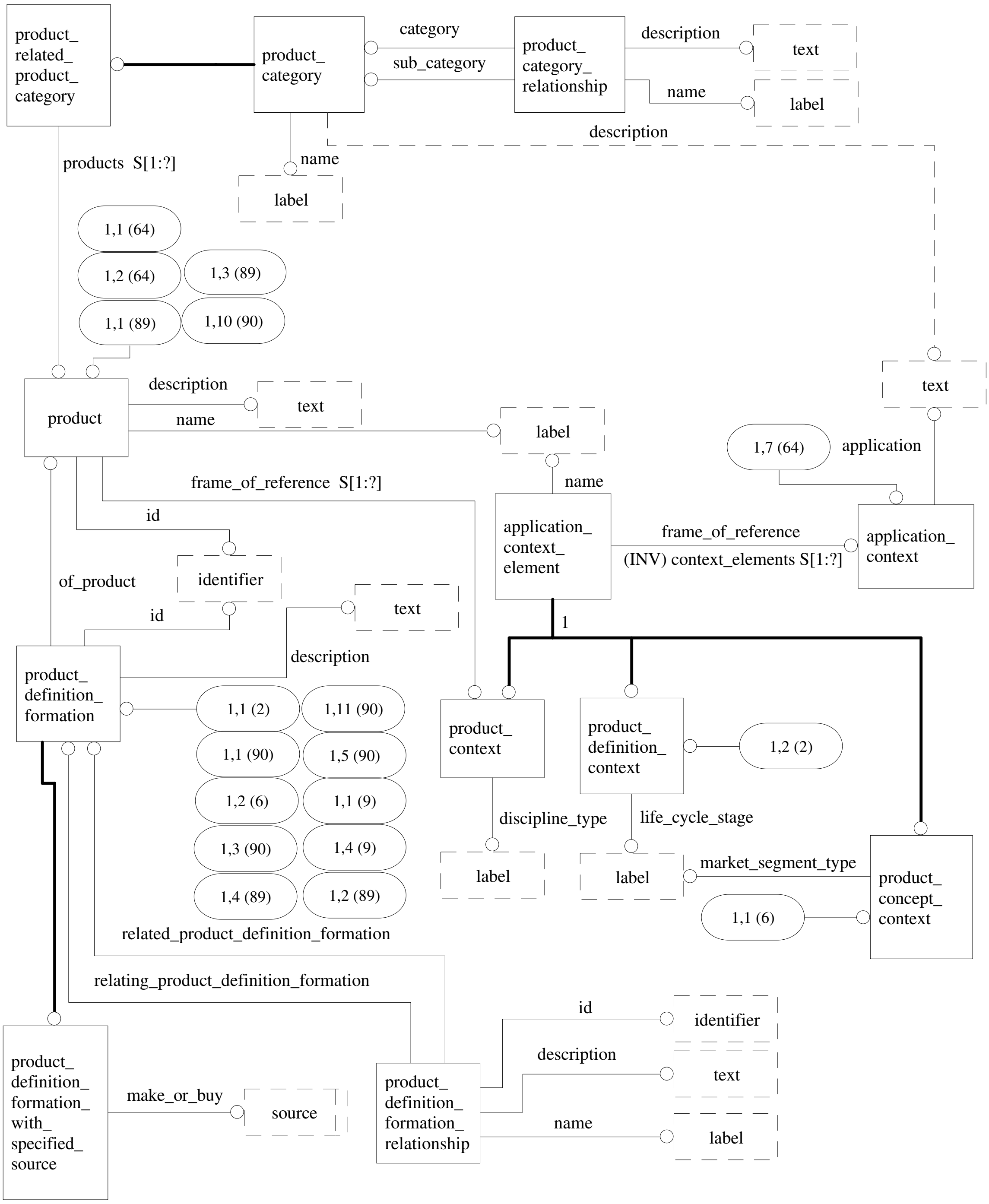

Figure H.1 - AIM EXPRESS-G diagram 1 of 94

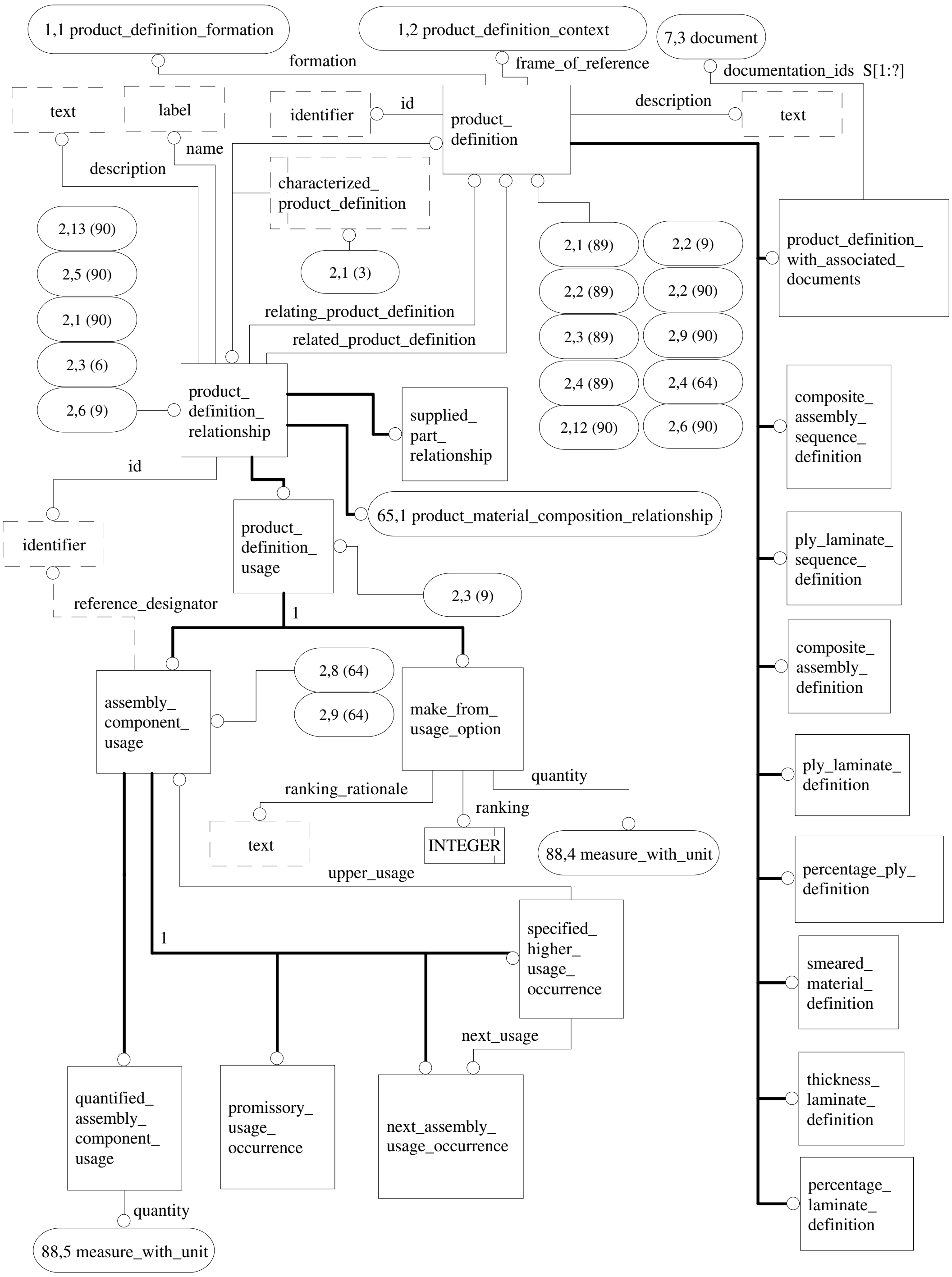

Figure H.2 - AIM EXPRESS-G diagram 2 of 94

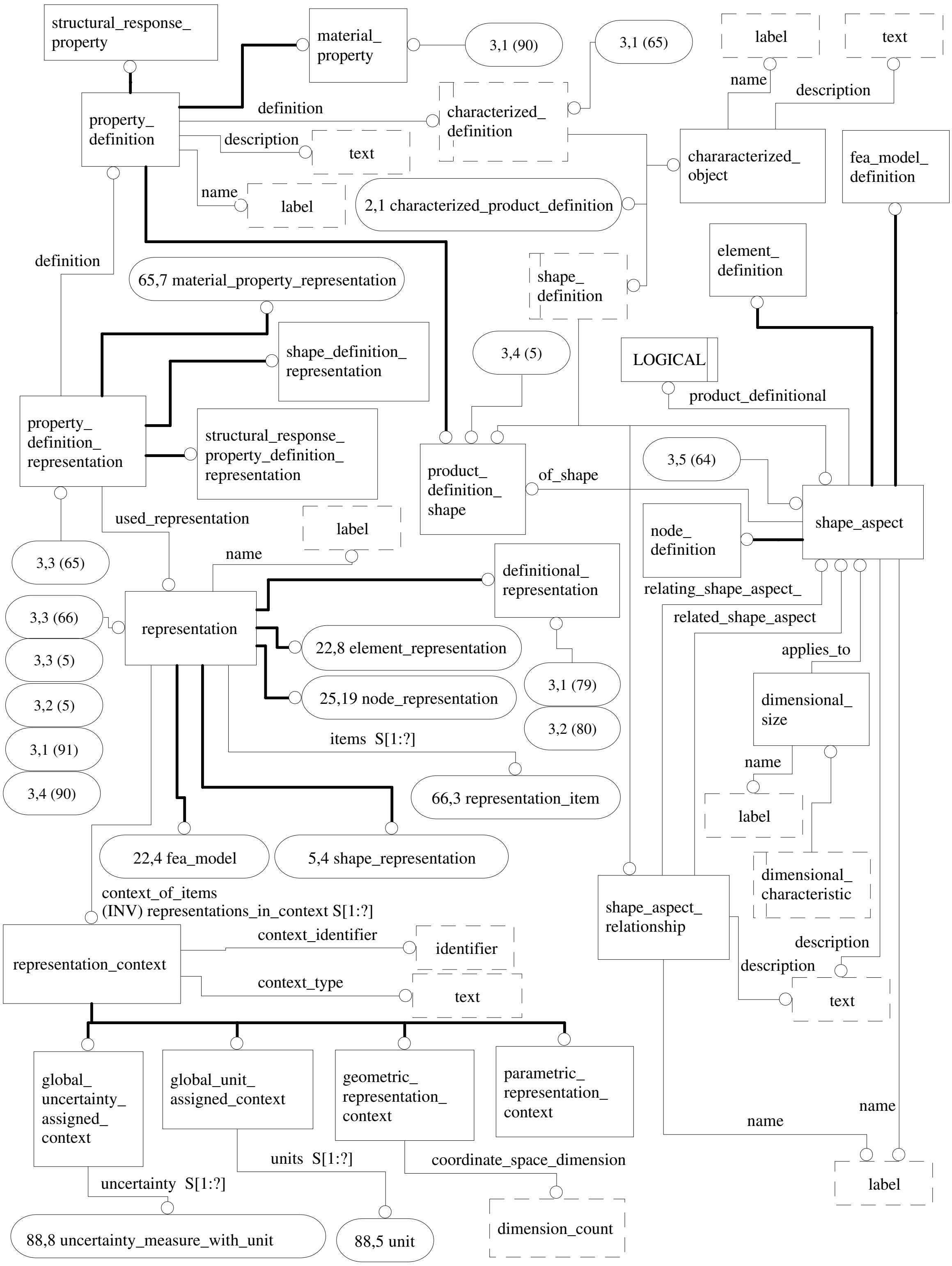

Figure H.3 - AIM EXPRESS-G diagram 3 of 94

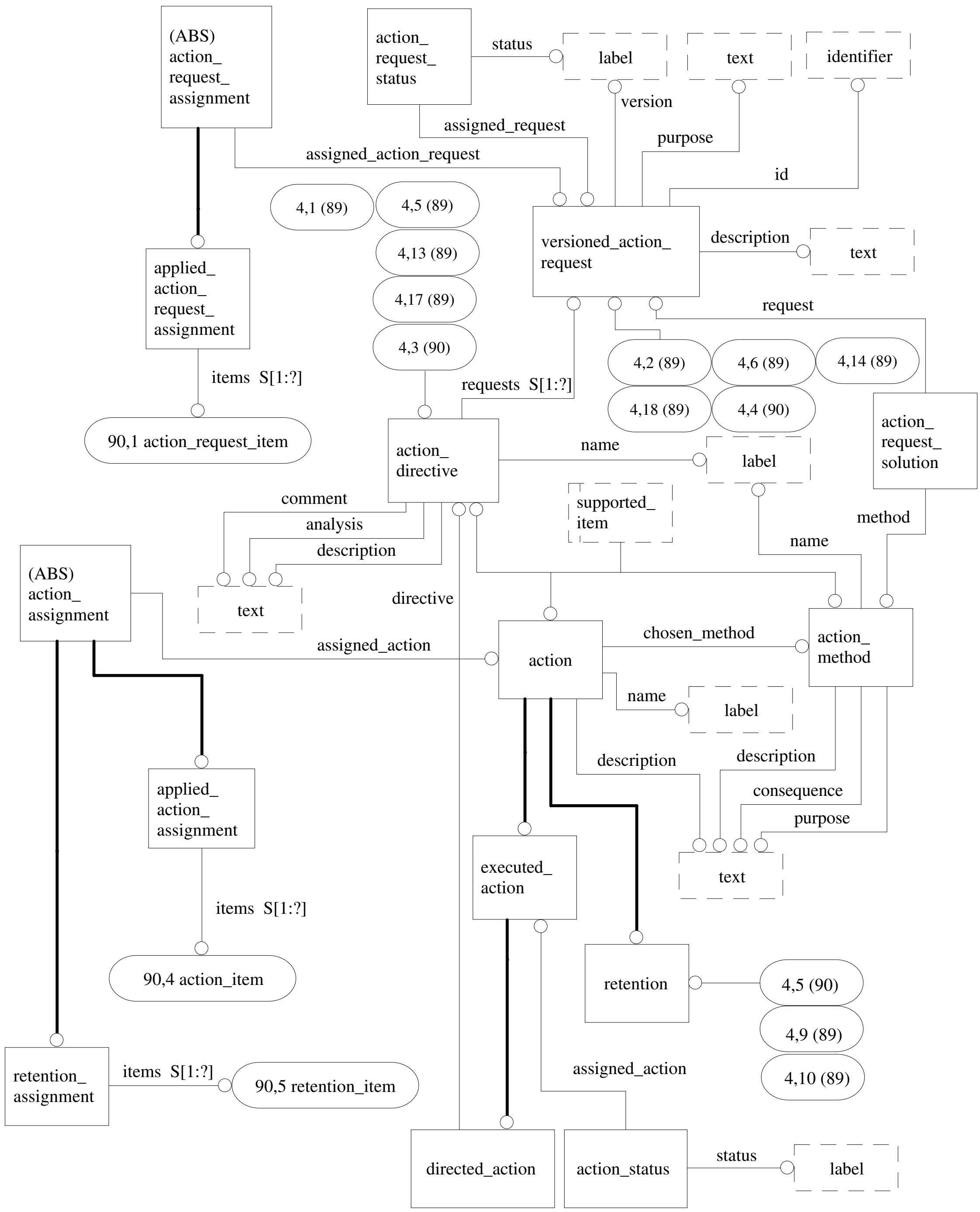

Figure H.4 - AIM EXPRESS-G diagram 4 of 94

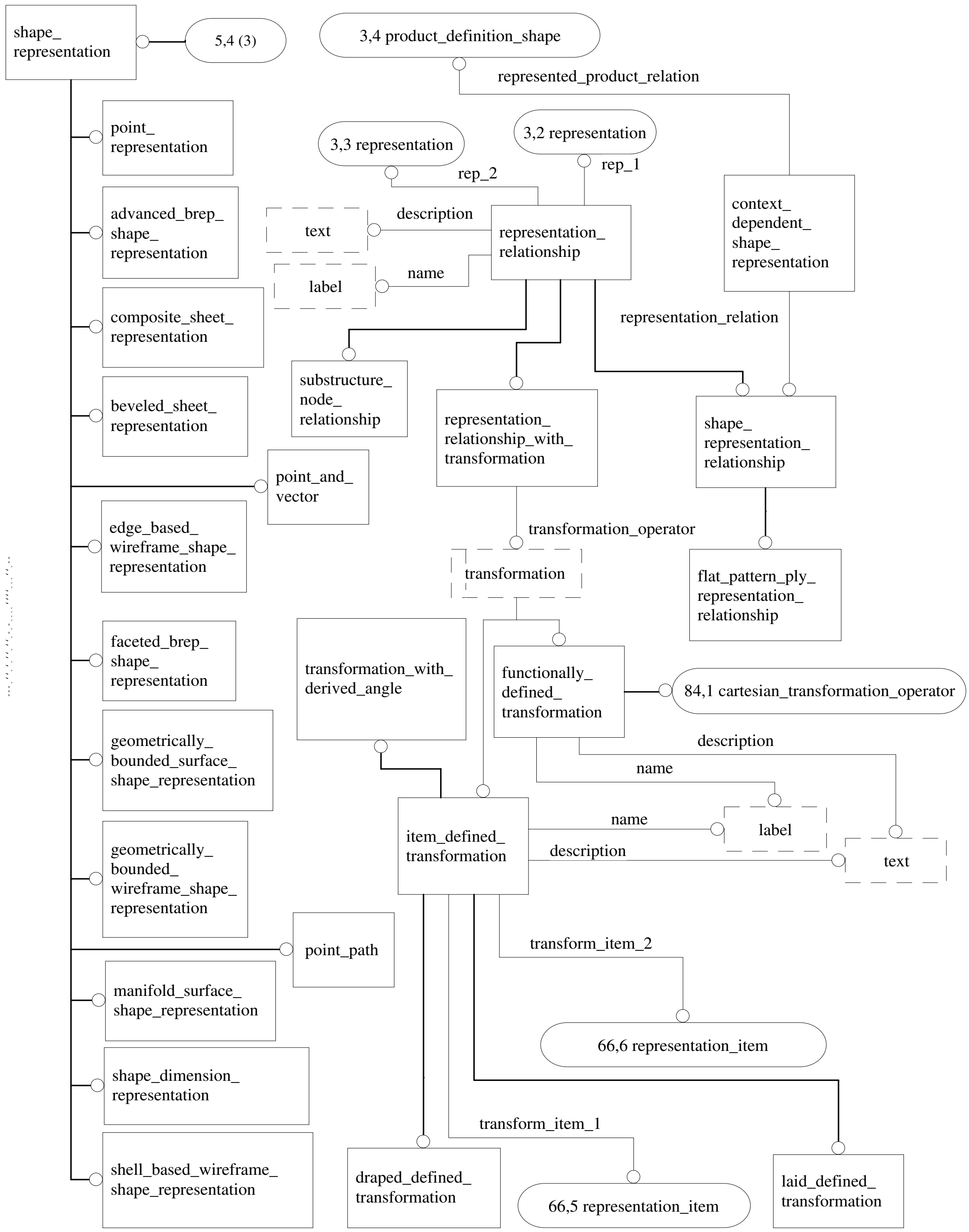

Figure H.5 - AIM EXPRESS-G diagram 5 of 94

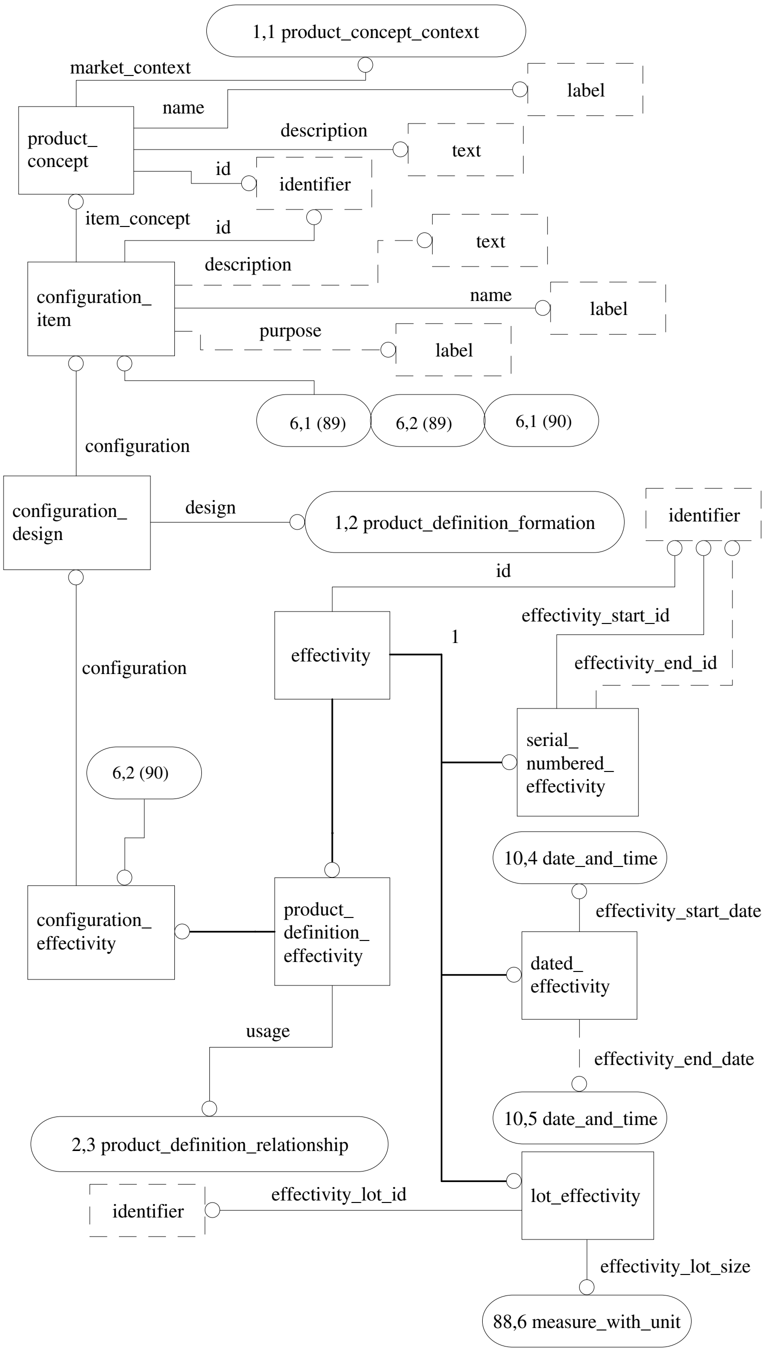

Figure H.6 - AIM EXPRESS-G diagram 6 of 94

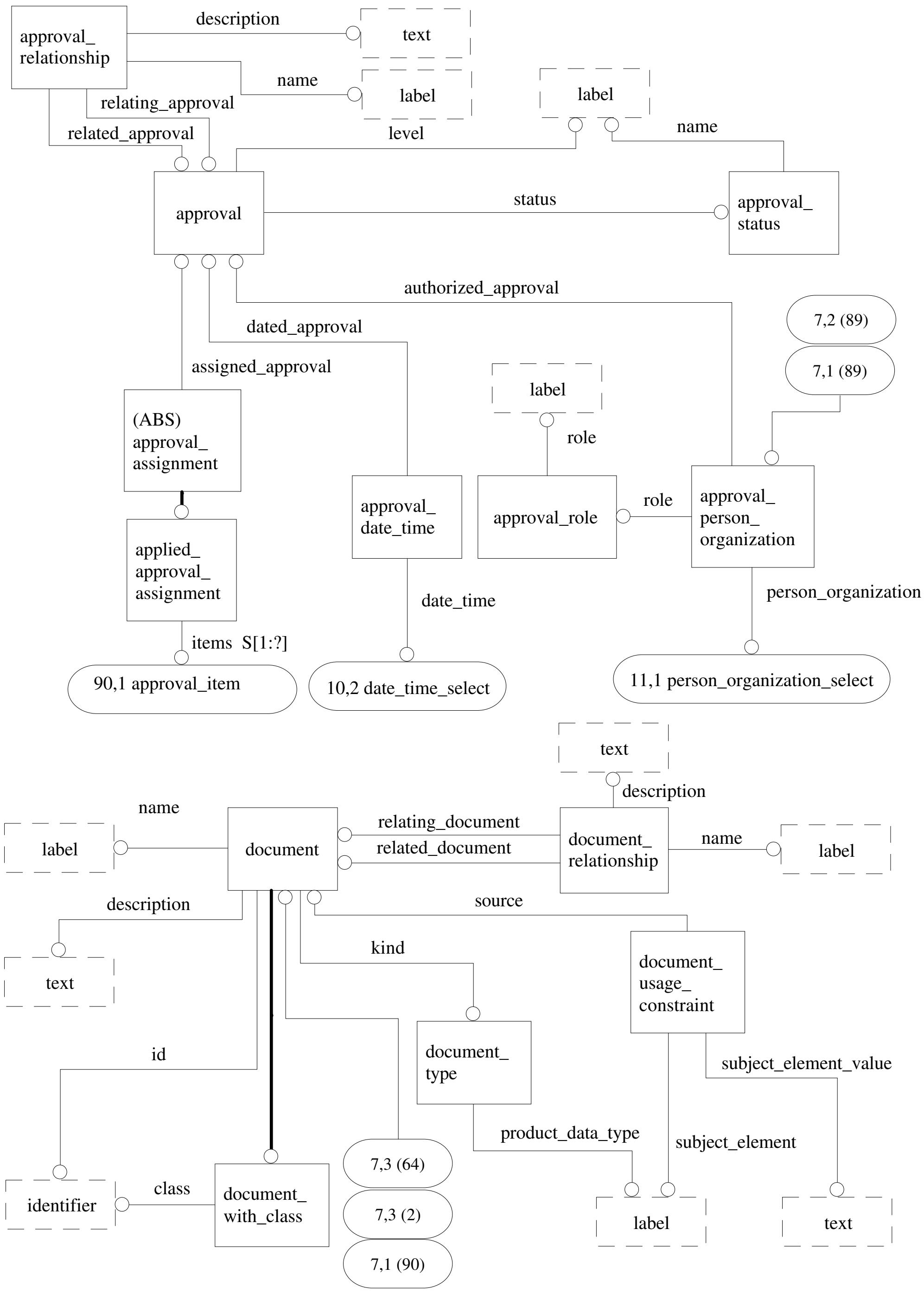

Figure H.7 - AIM EXPRESS-G diagram 7 of 94

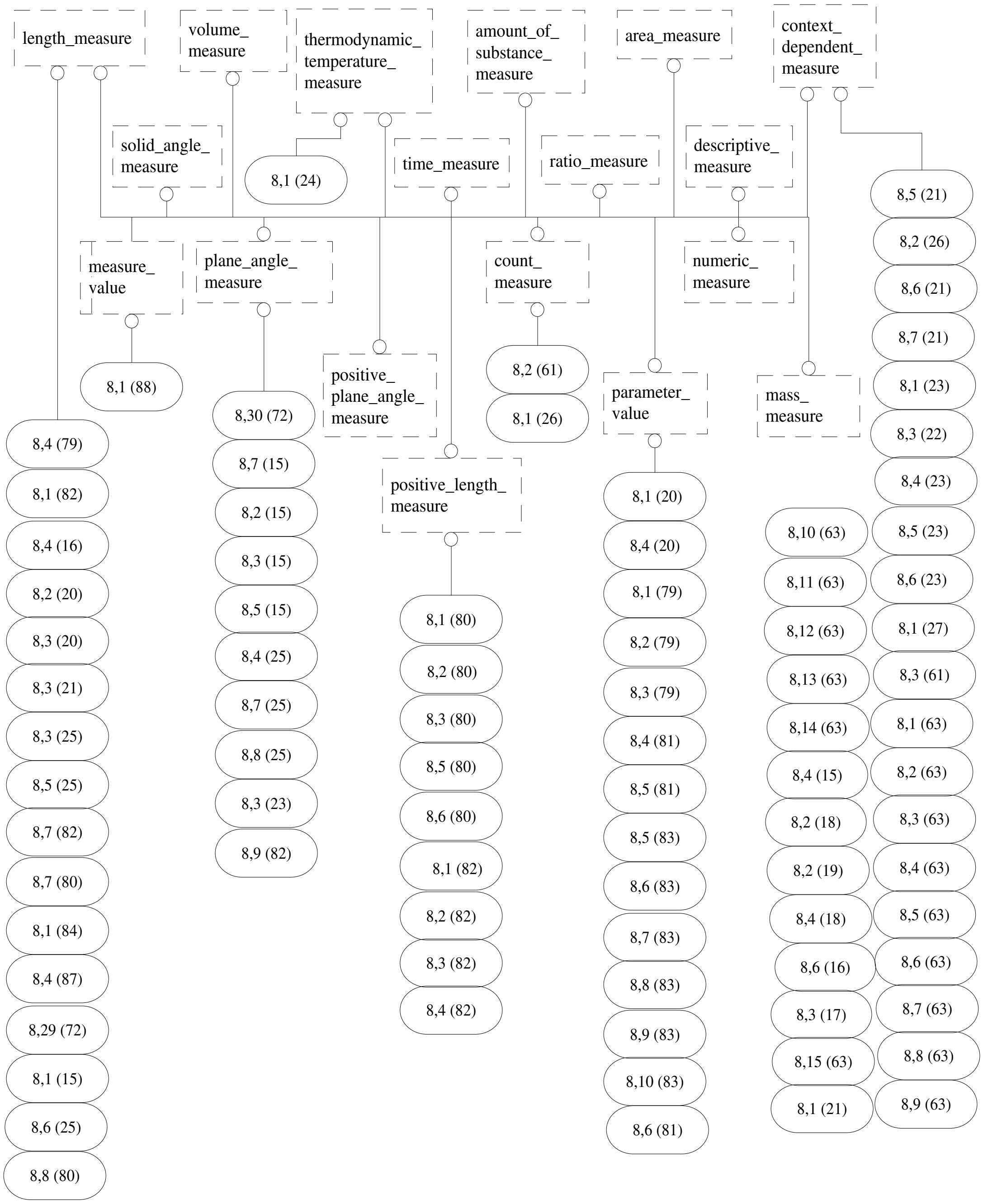

Figure H.8 - AIM EXPRESS-G diagram 8 of 94

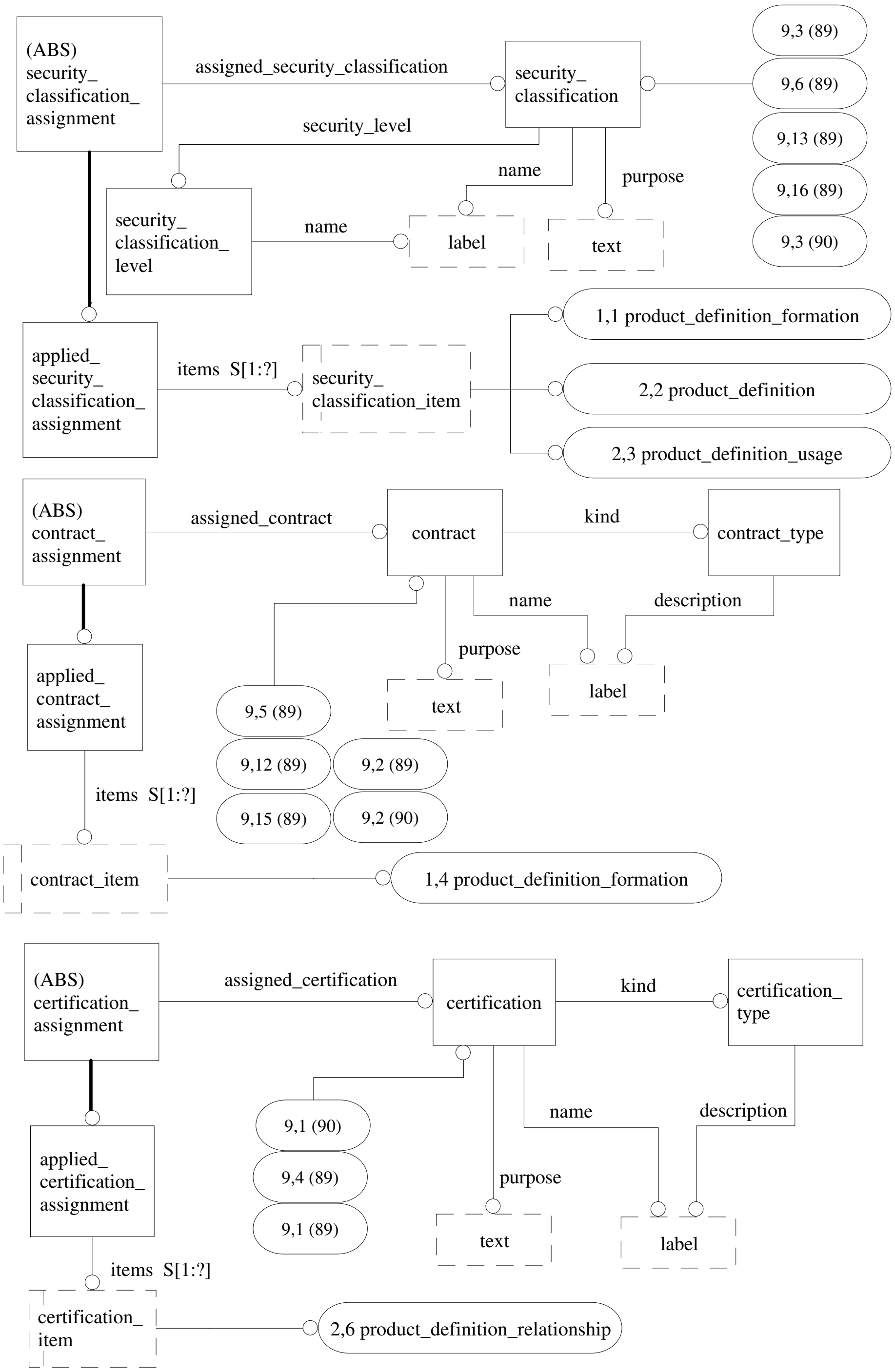

Figure H.9 - AIM EXPRESS-G diagram 9 of 94

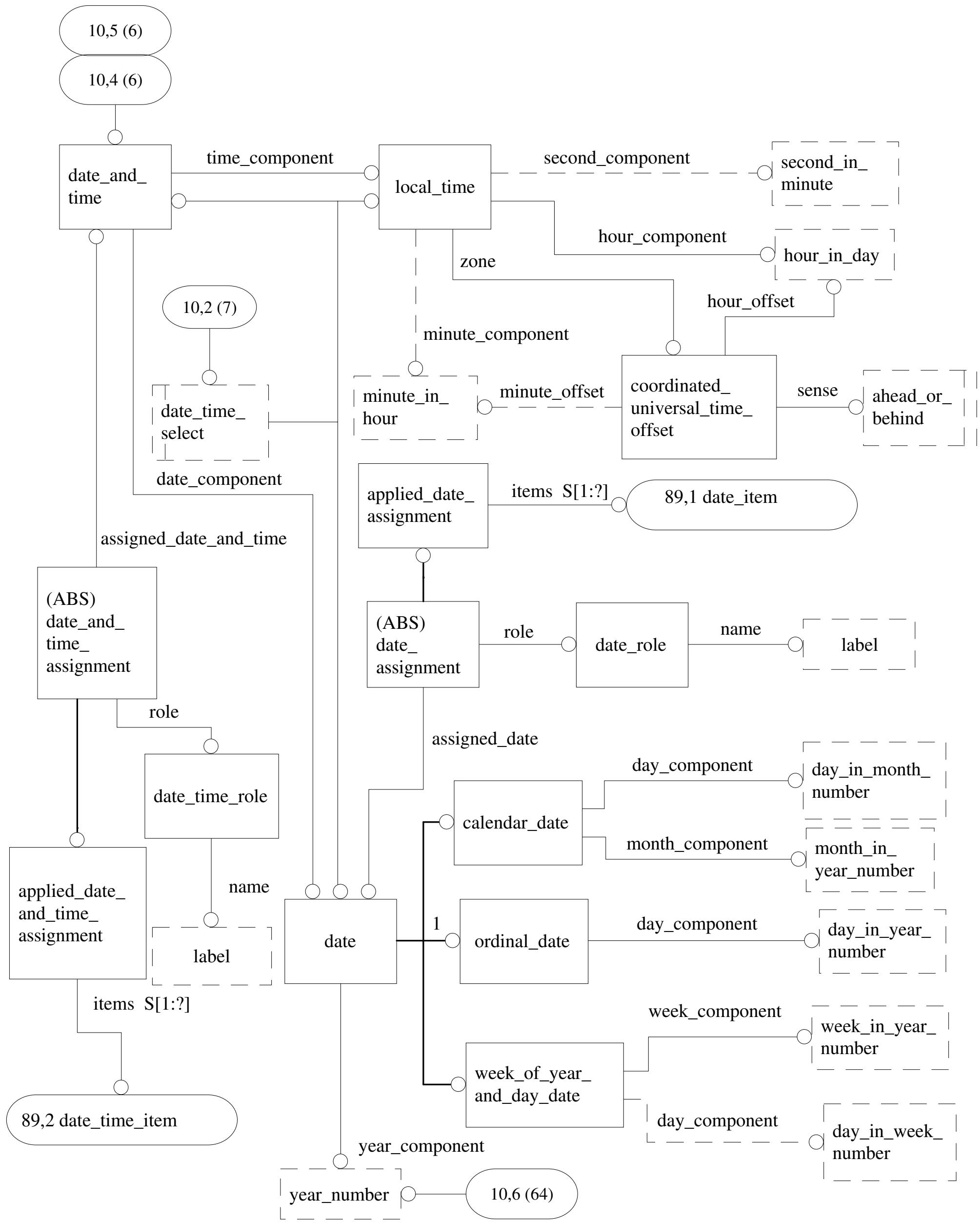

Figure H.10 - AIM EXPRESS-G diagram 10 of 94

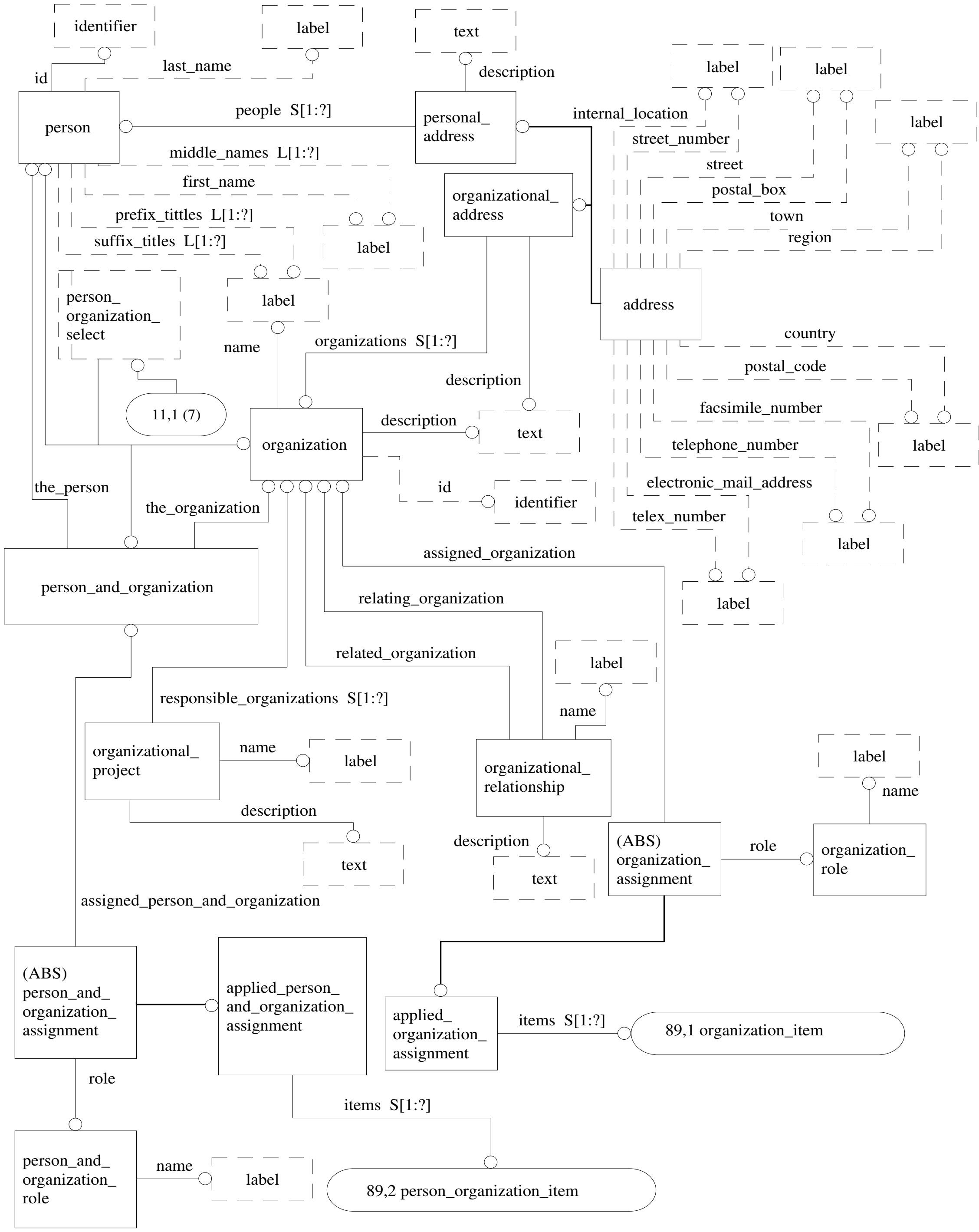

Figure H.11 - AIM EXPRESS-G diagram 11 of 94

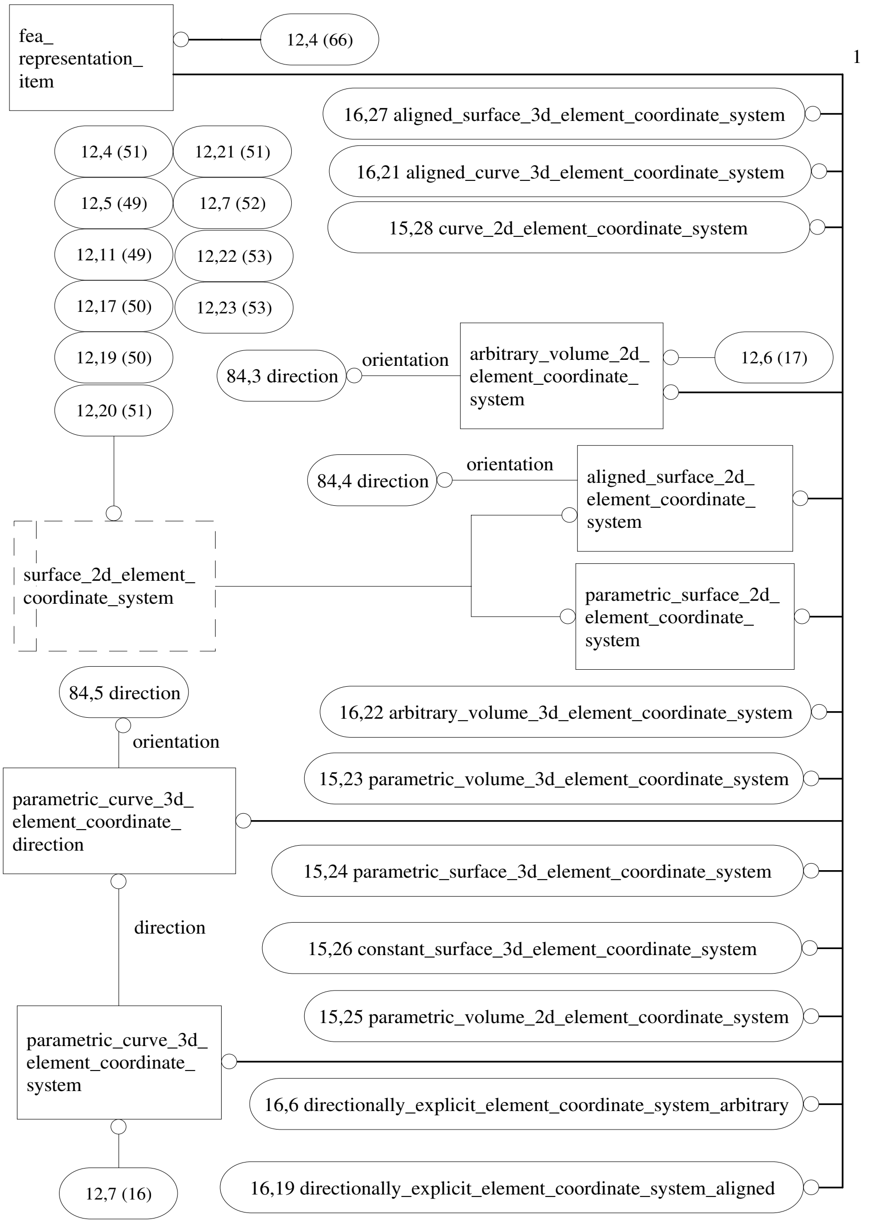

Figure H.12 - AIM EXPRESS-G diagram 12 of 94

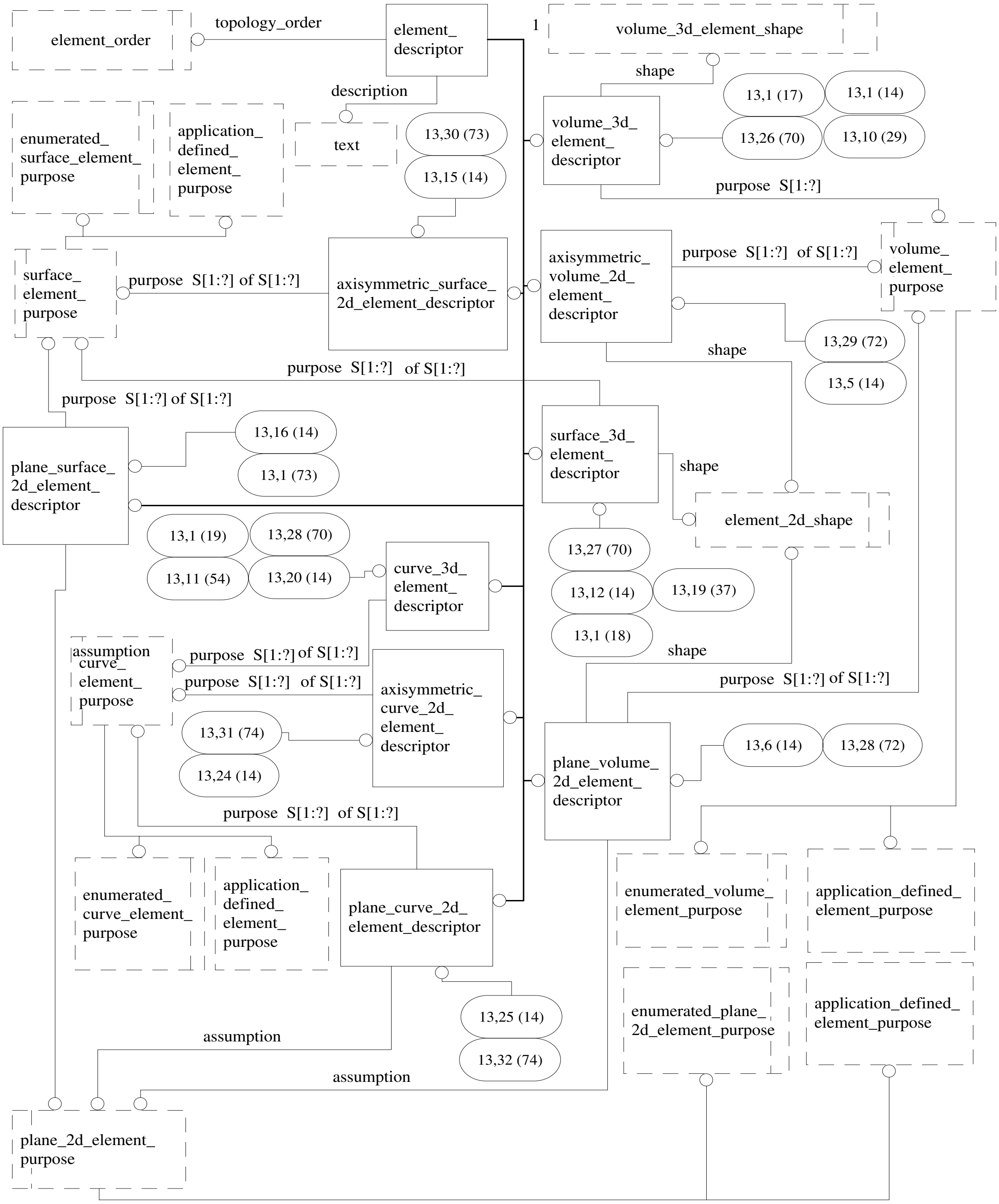

Figure H.13 - AIM EXPRESS-G diagram 13 of 94

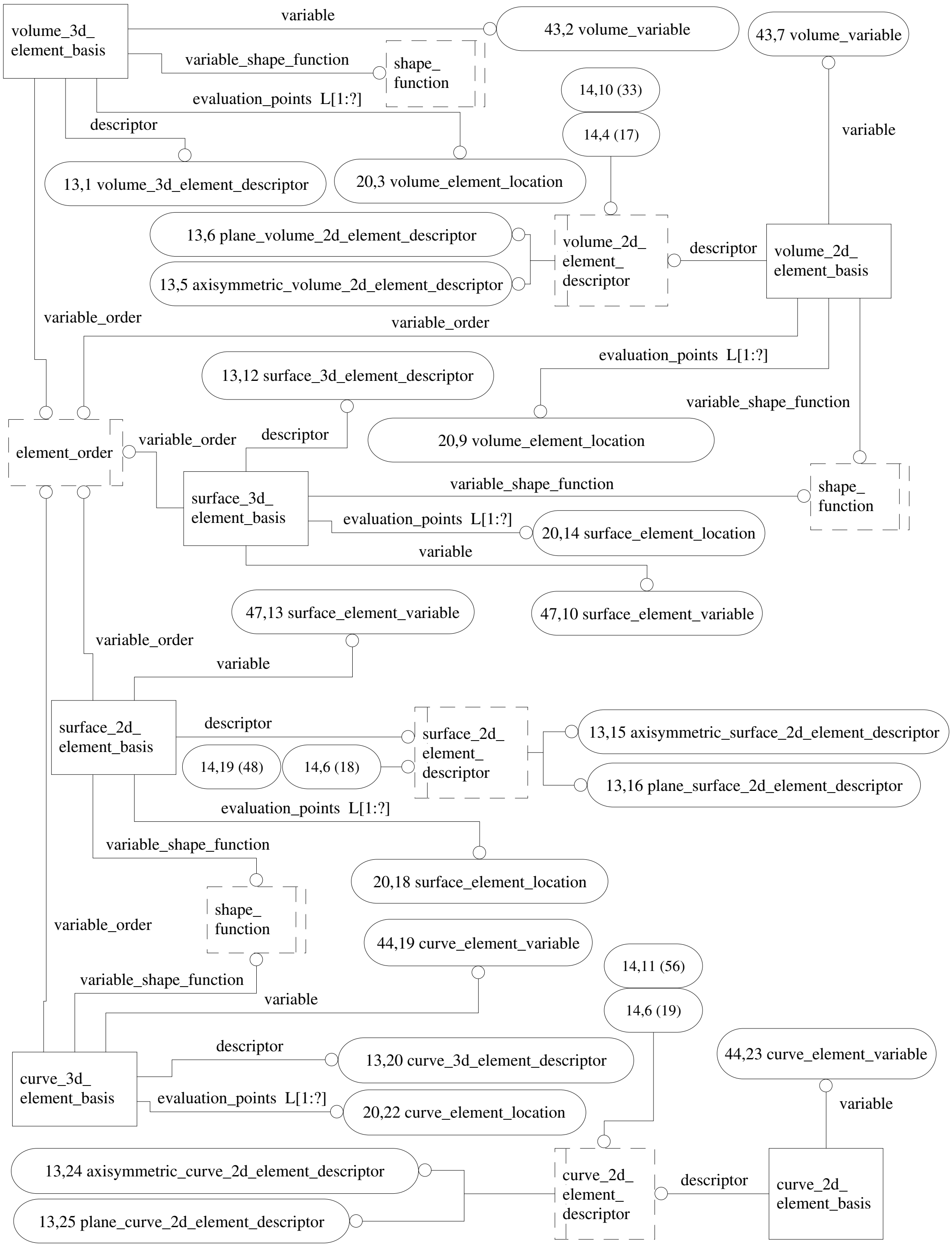

Figure H.14 - AIM EXPRESS-G diagram 14 of 94

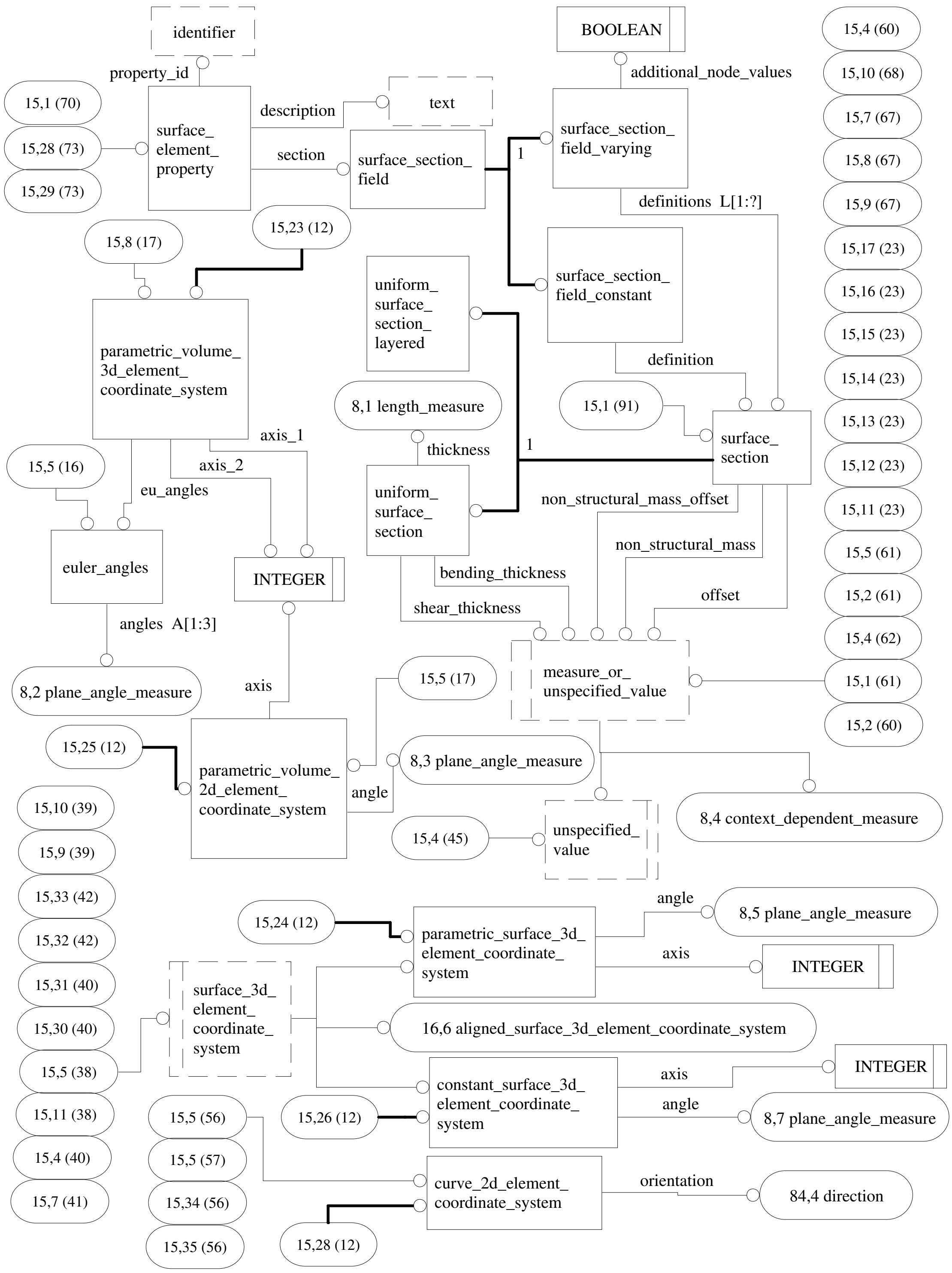

Figure H.15 - AIM EXPRESS-G diagram 15 of 94

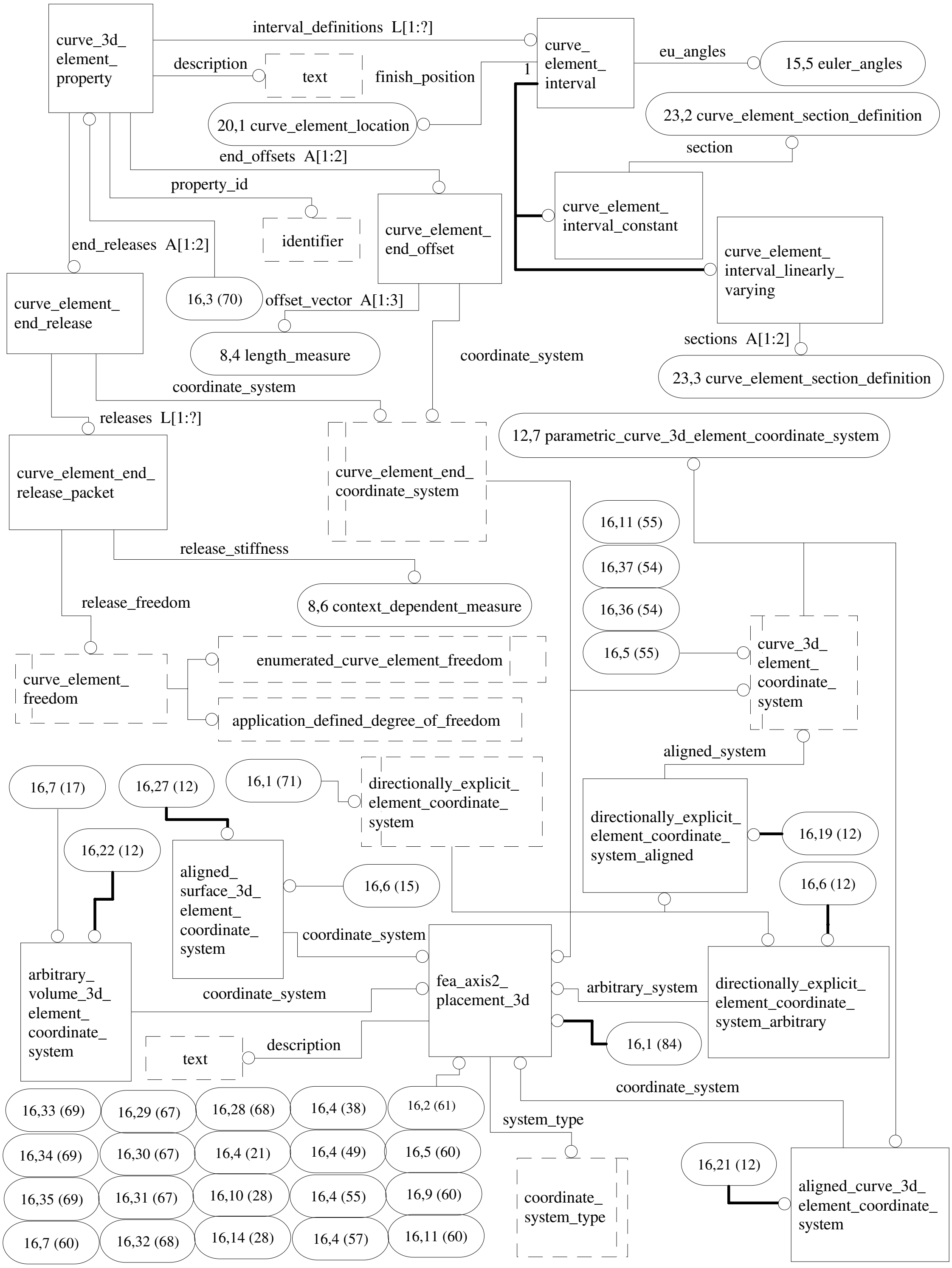

Figure H.16 - AIM EXPRESS-G diagram 16 of 94

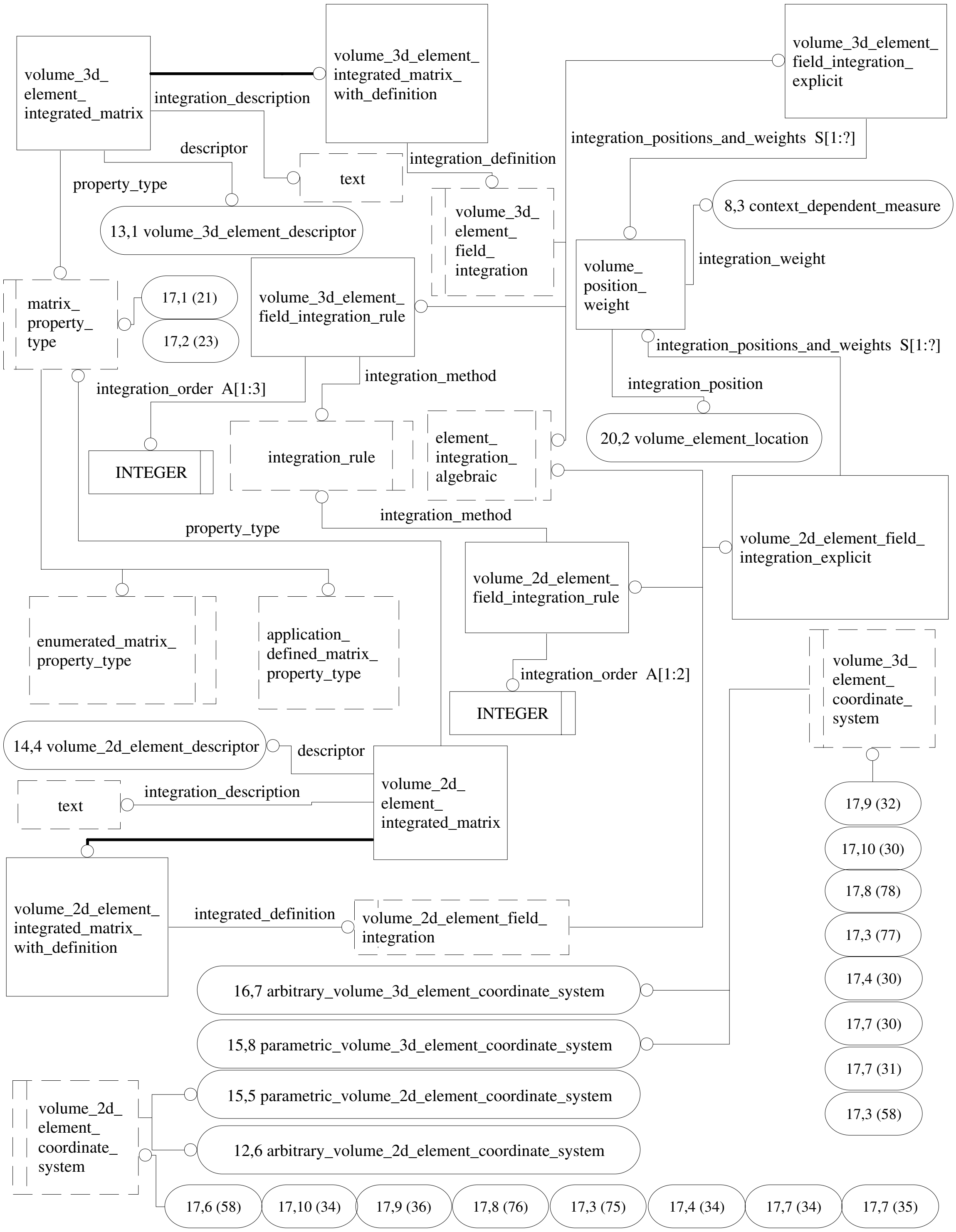

Figure H.17 - AIM EXPRESS-G diagram 17 of 94

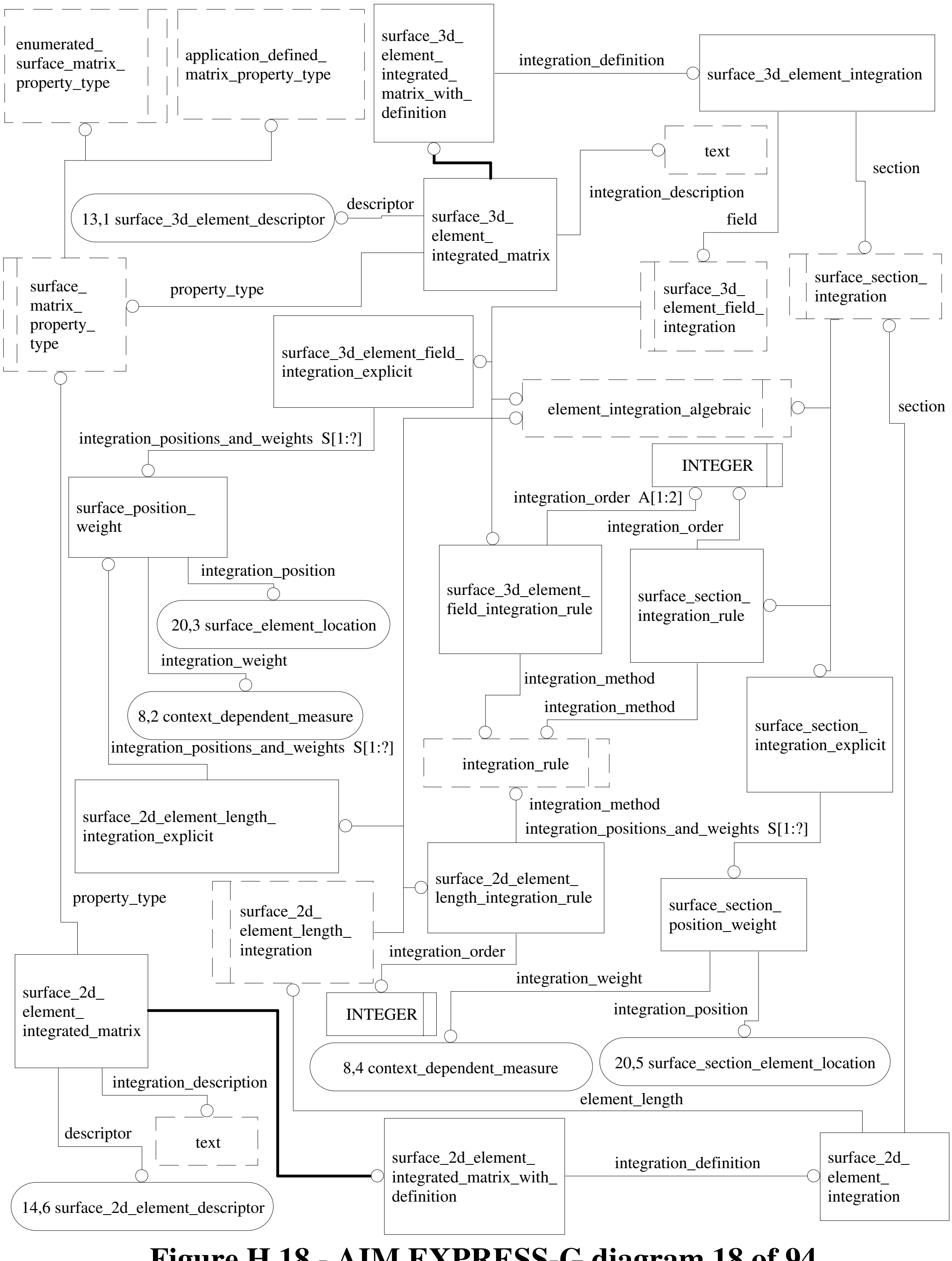

Figure H.18 - AIM EXPRESS-G diagram 18 of 94

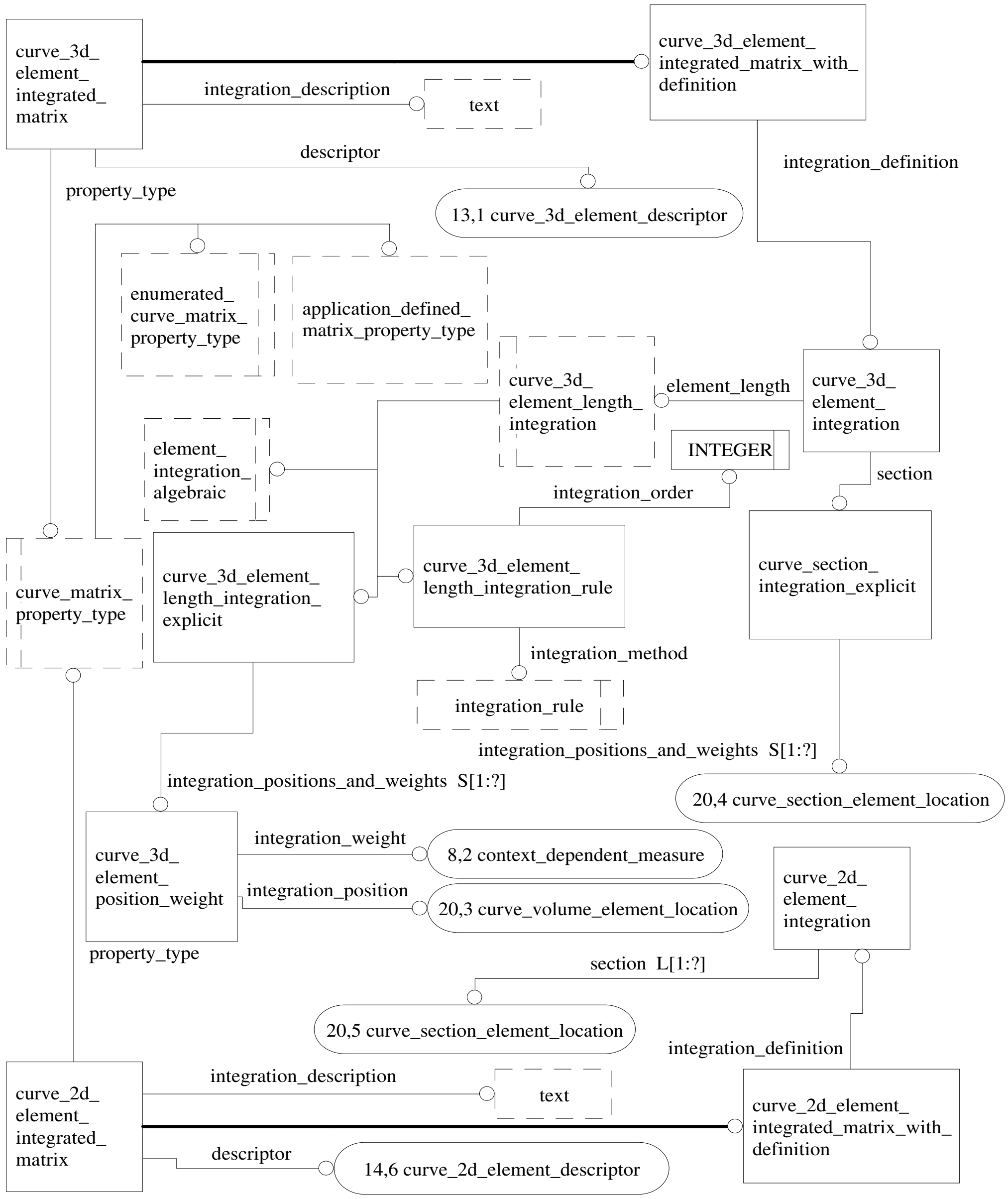

Figure H.19 - AIM EXPRESS-G diagram 19 of 94

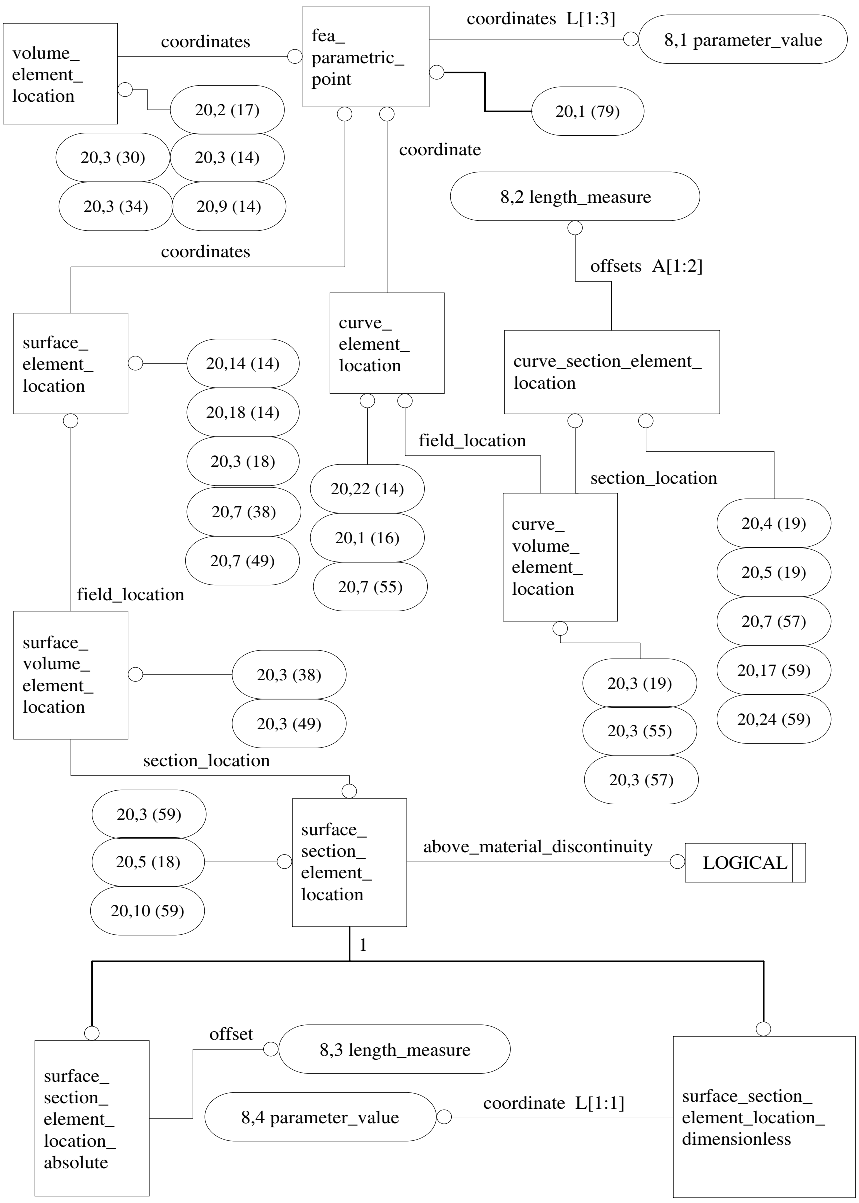

Figure H.20 - AIM EXPRESS-G diagram 20 of 94

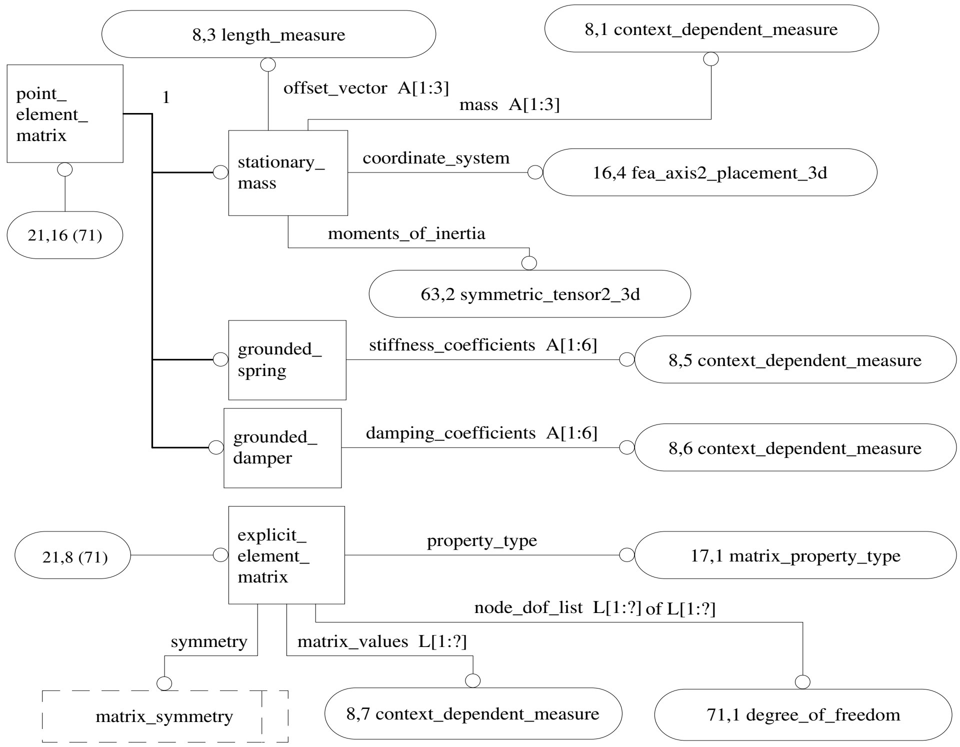

Figure H.21 - AIM EXPRESS-G diagram 21 of 94

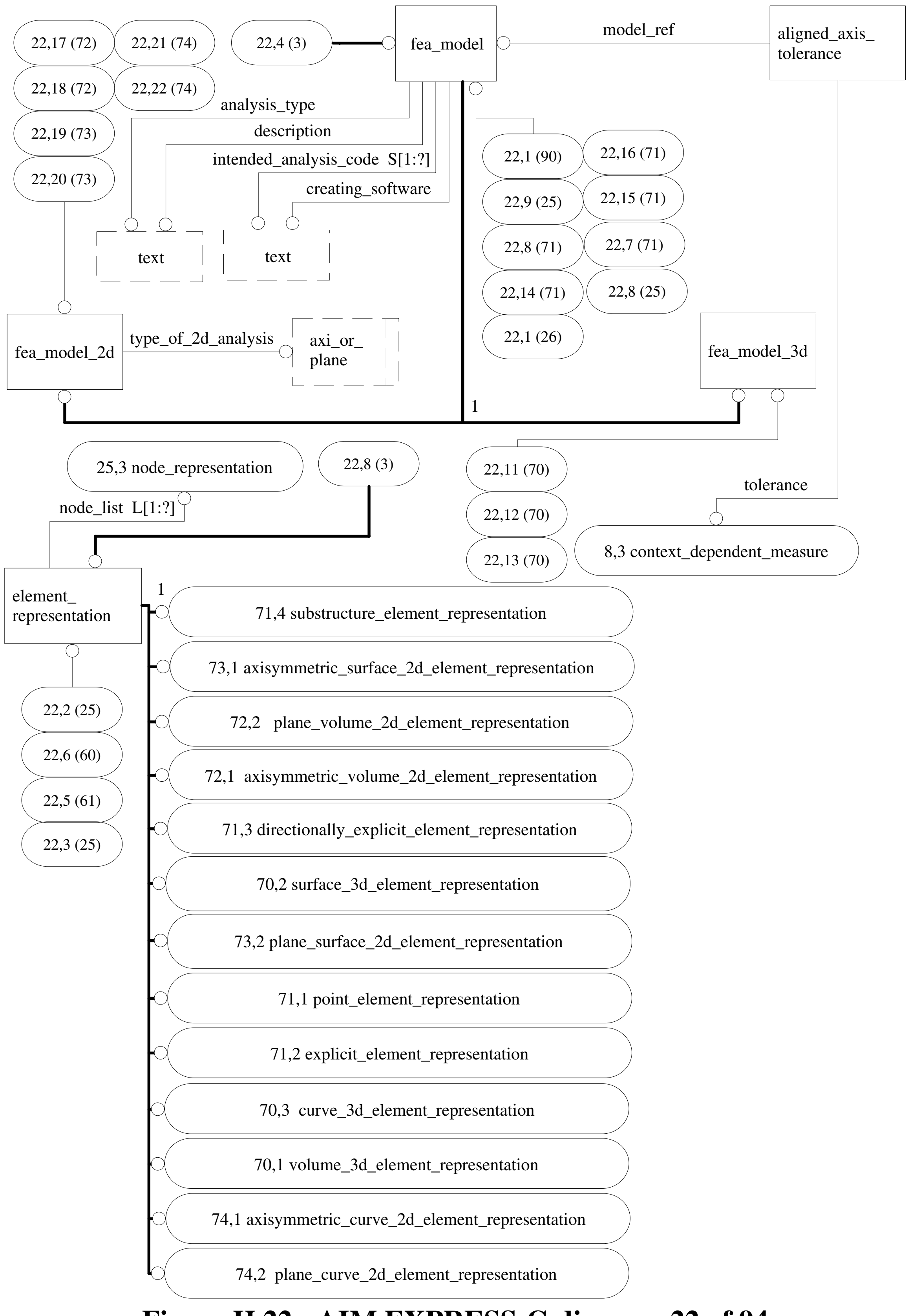

Figure H.22 - AIM EXPRESS-G diagram 22 of 94

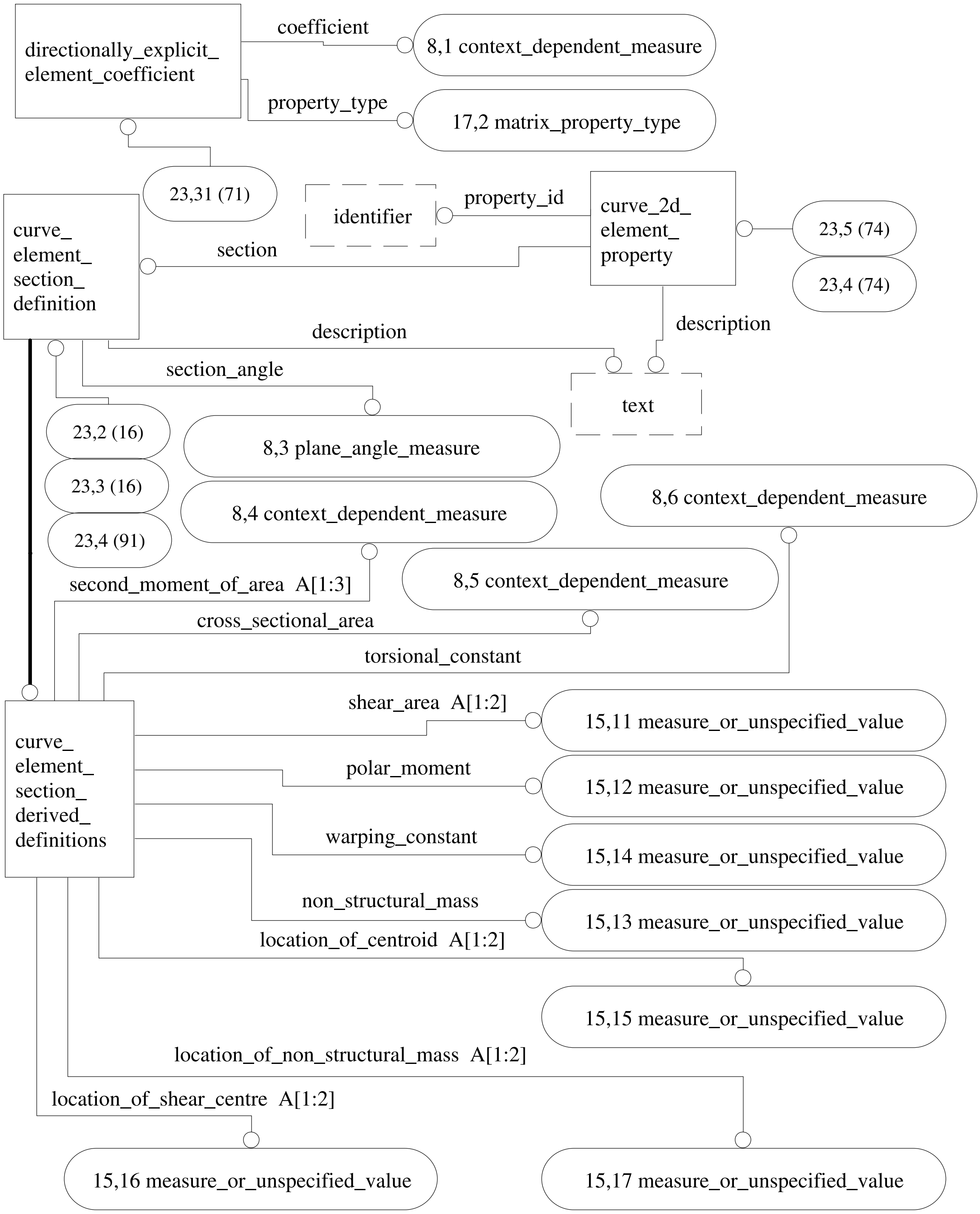

Figure H.23 - AIM EXPRESS-G diagram 23 of 94

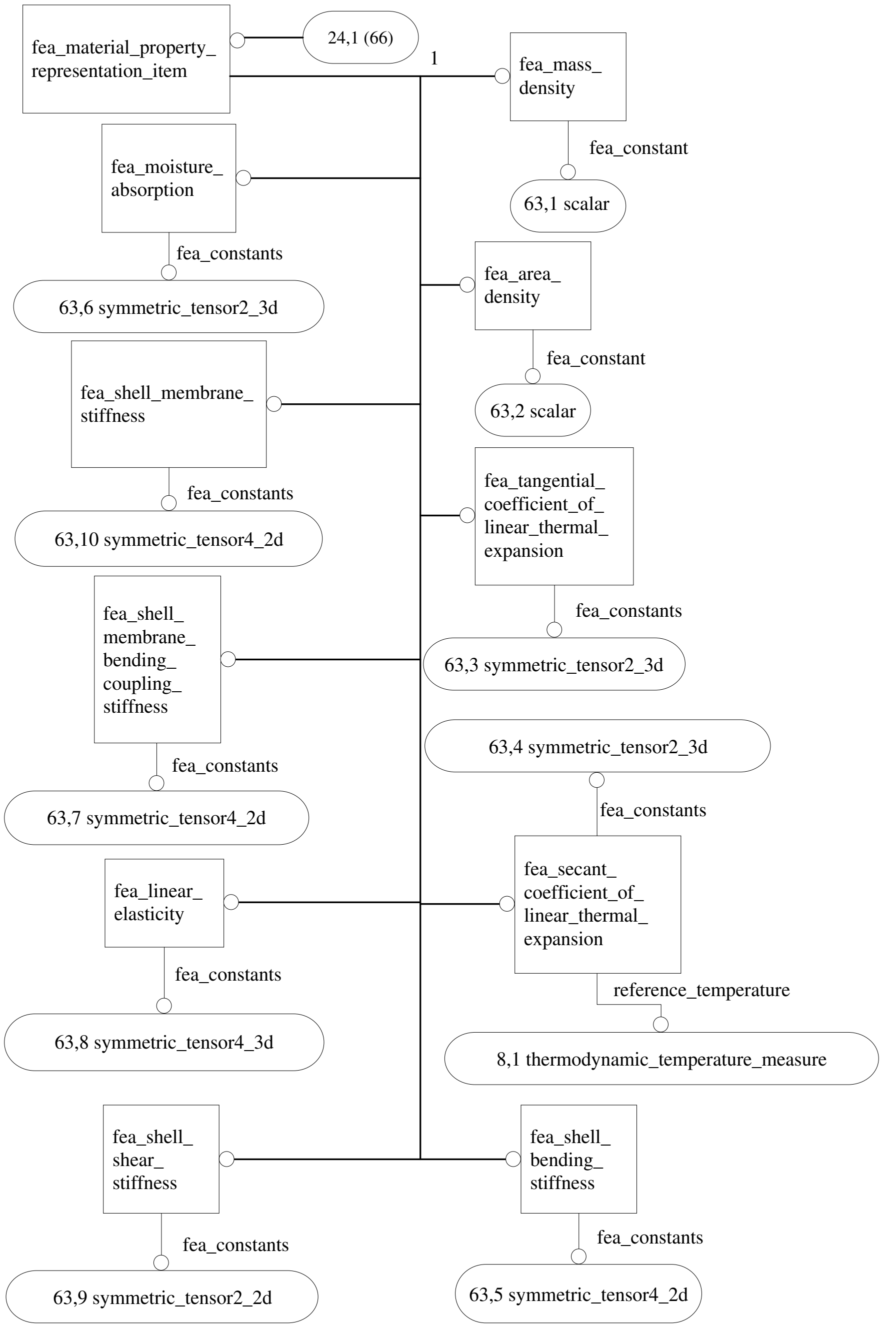

Figure H.24 - AIM EXPRESS-G diagram 24 of 94

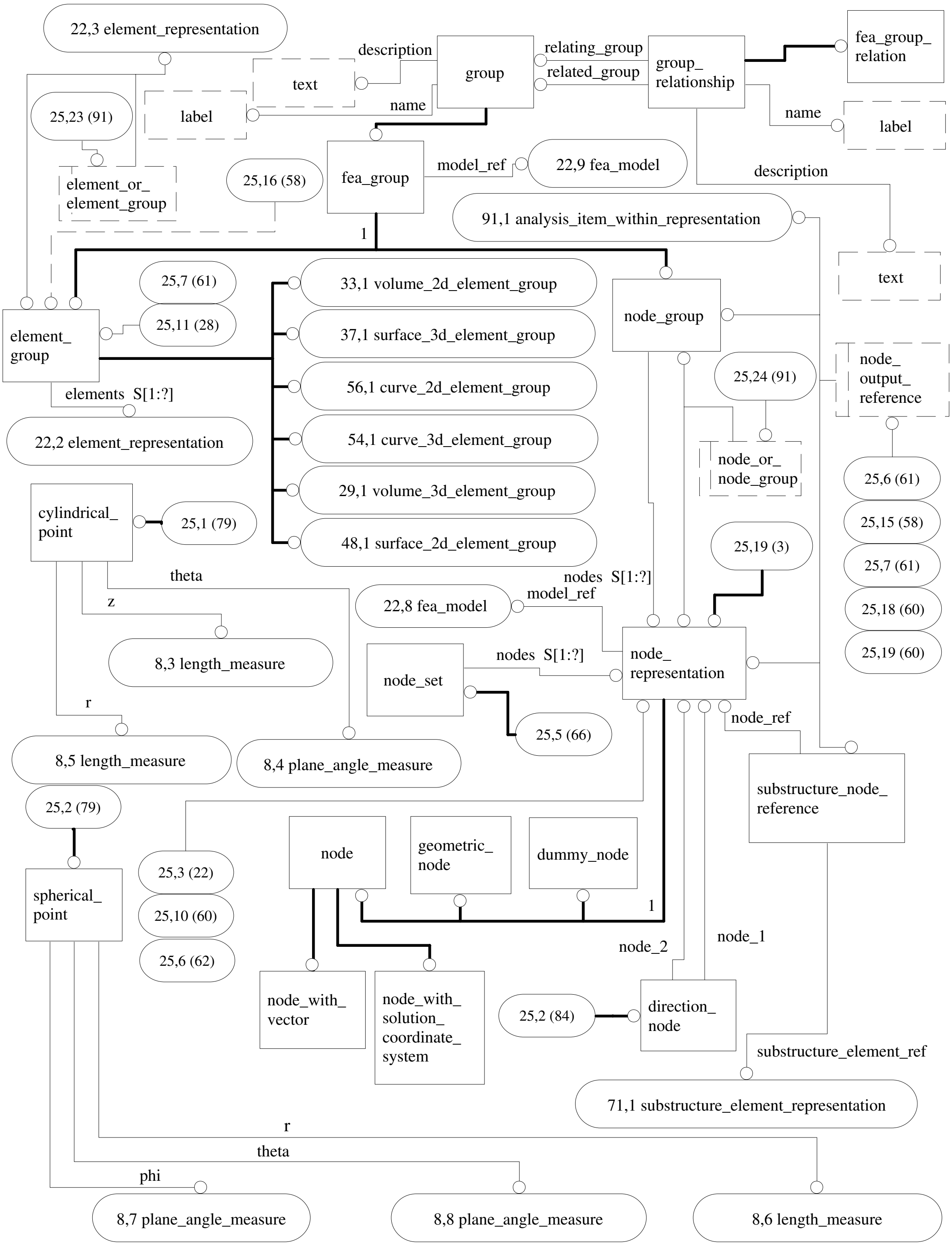

Figure H.25 - AIM EXPRESS-G diagram 25 of 94

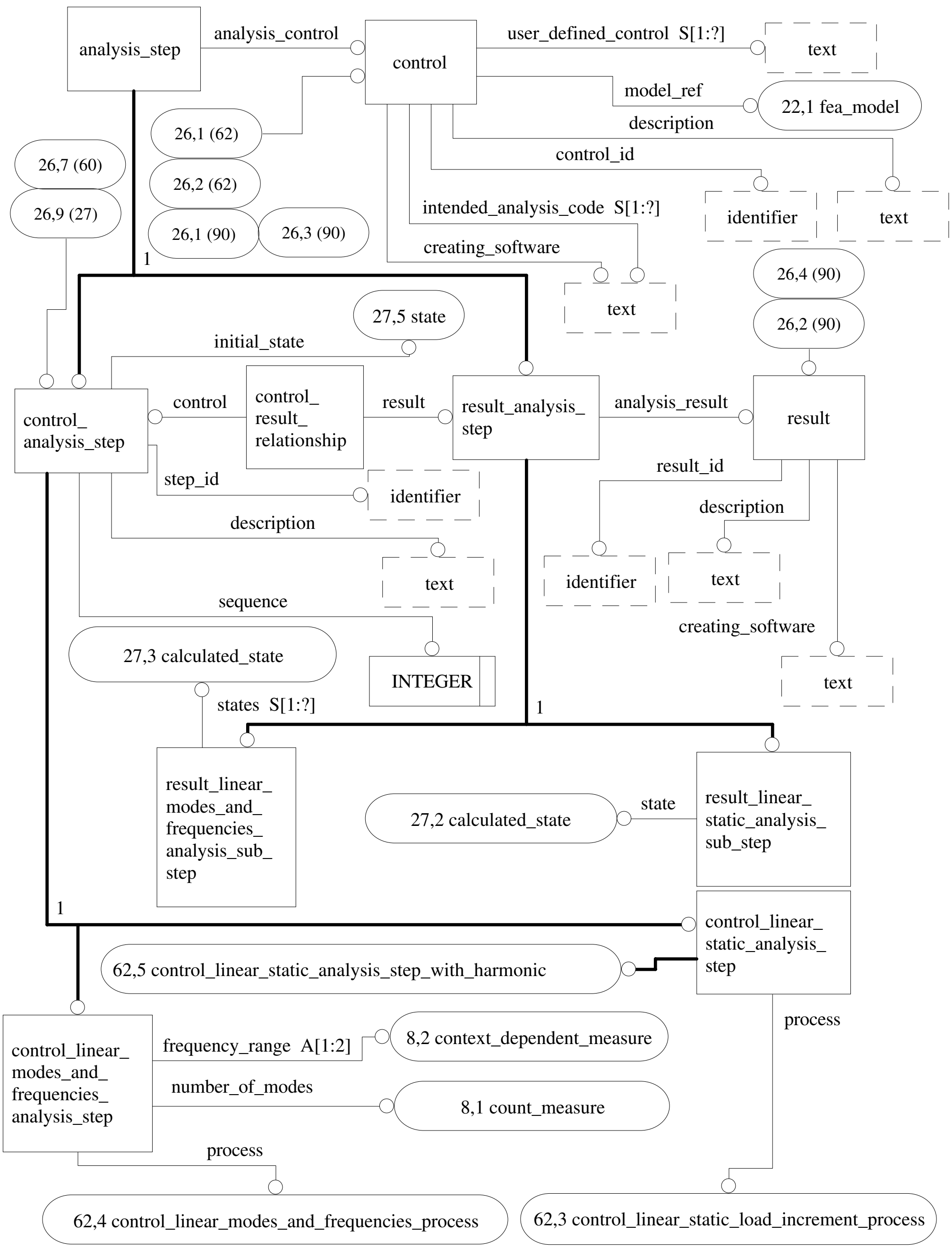

Figure H.26 - AIM EXPRESS-G diagram 26 of 94

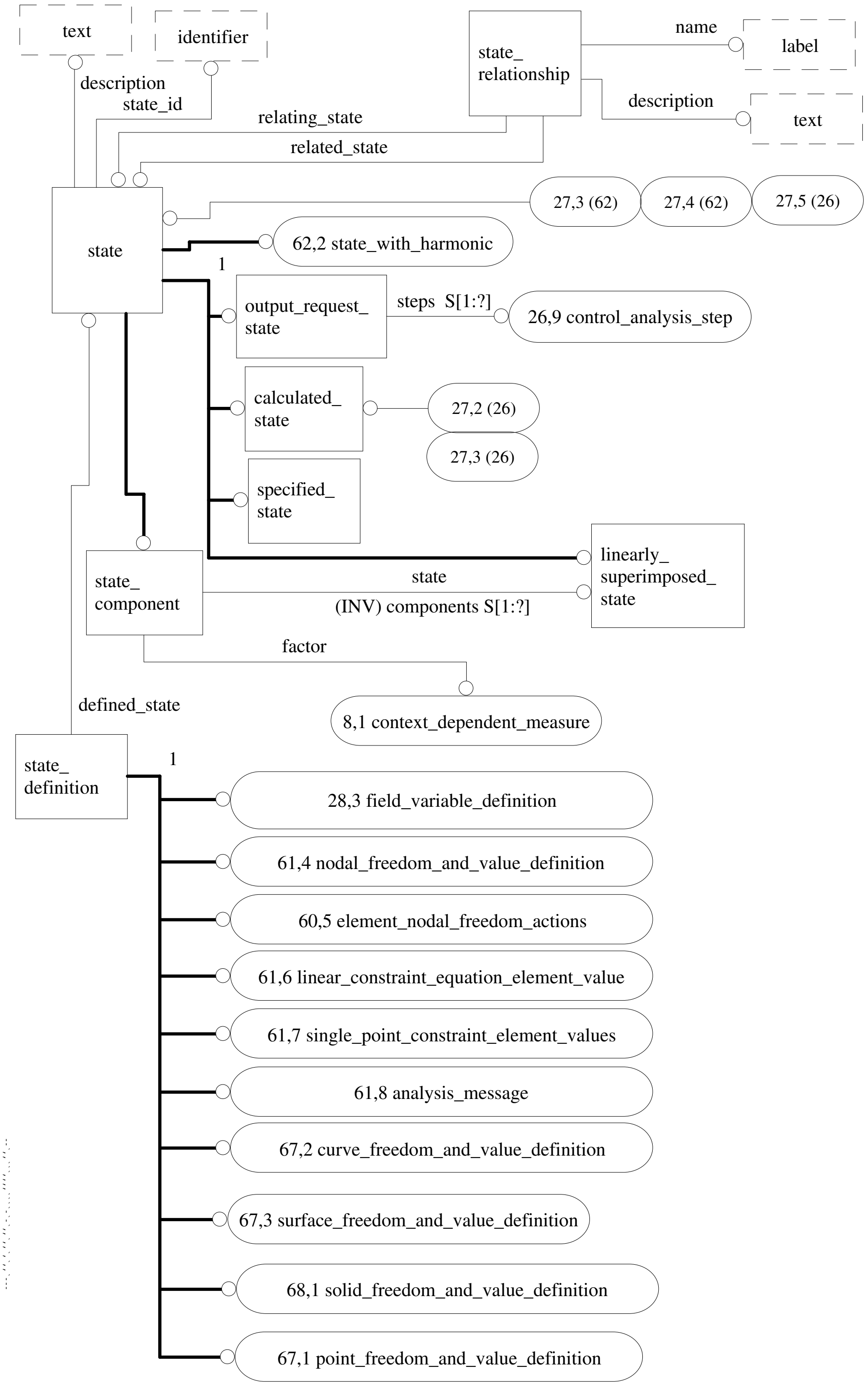

Figure H.27 - AIM EXPRESS-G diagram 27 of 94

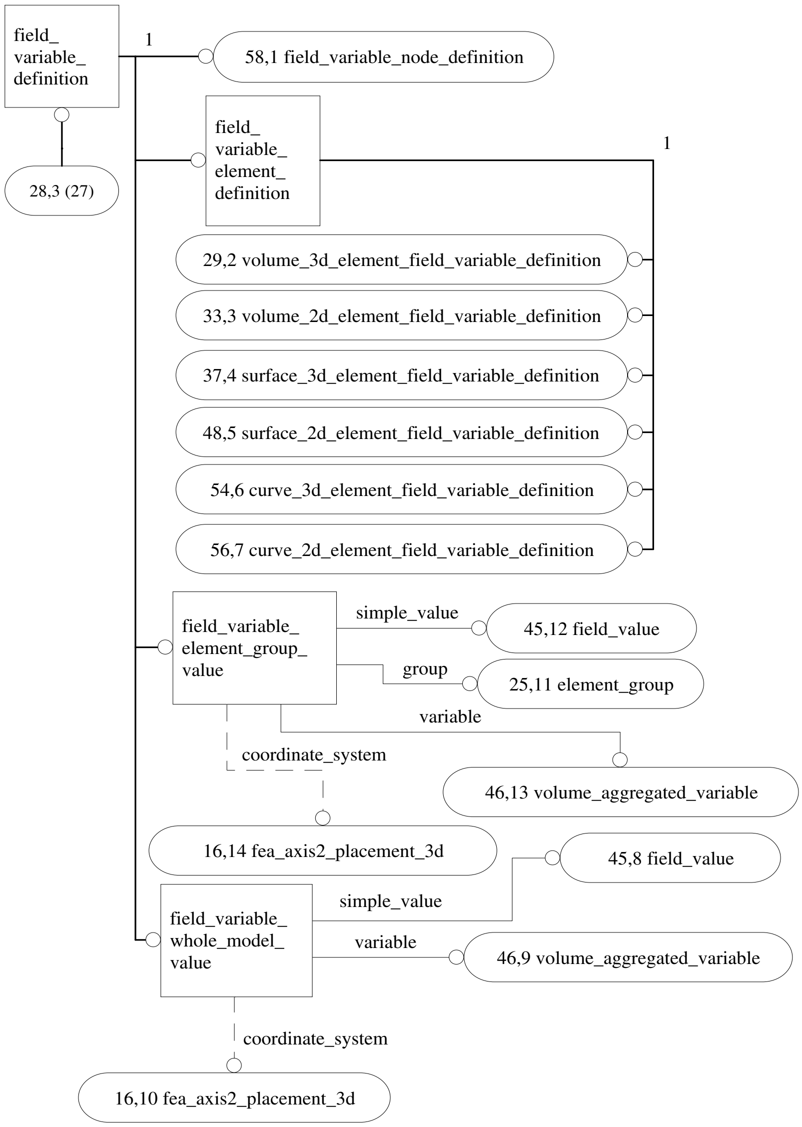

Figure H.28 - AIM EXPRESS-G diagram 28 of 94

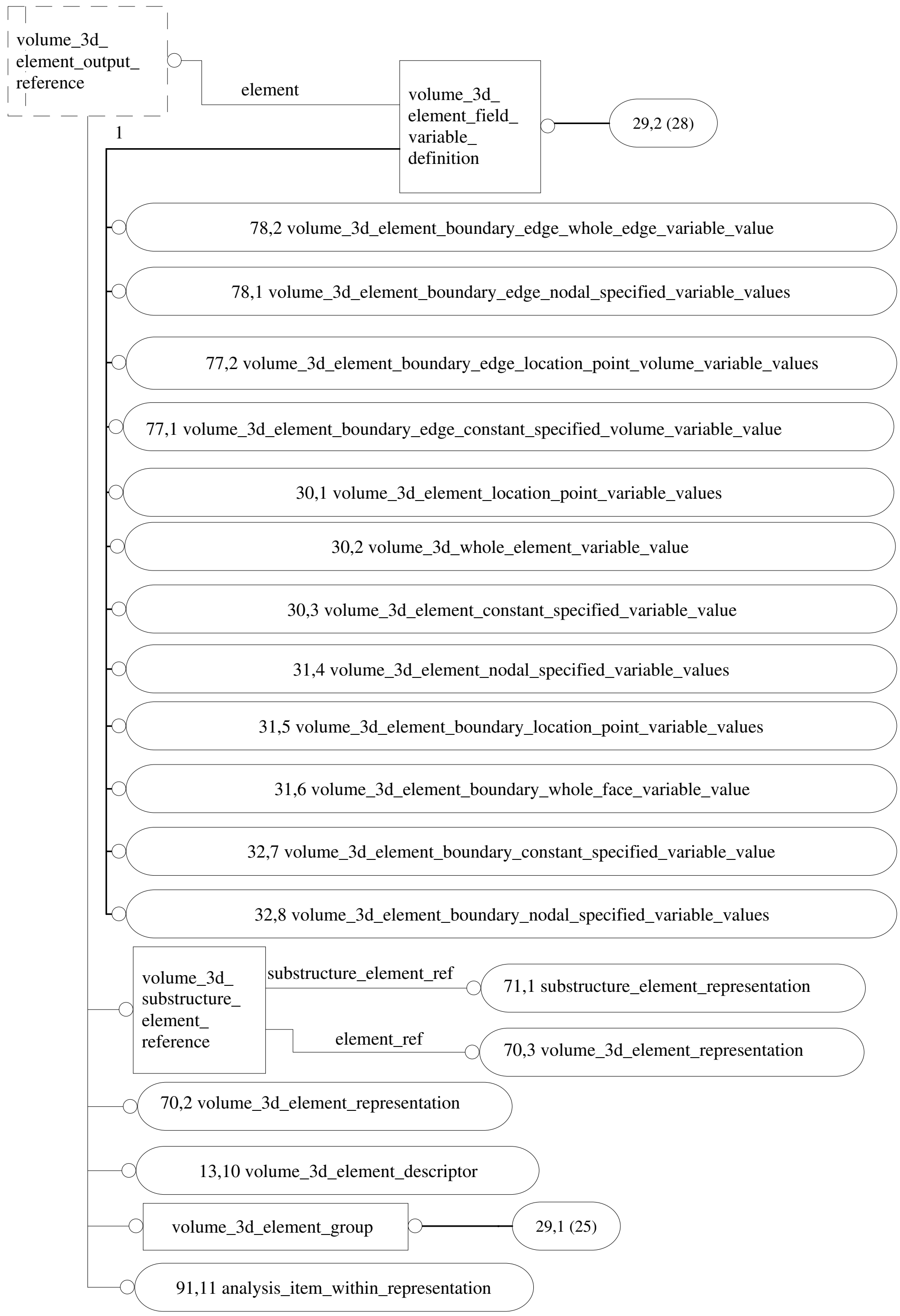

Figure H.29 - AIM EXPRESS-G diagram 29 of 94

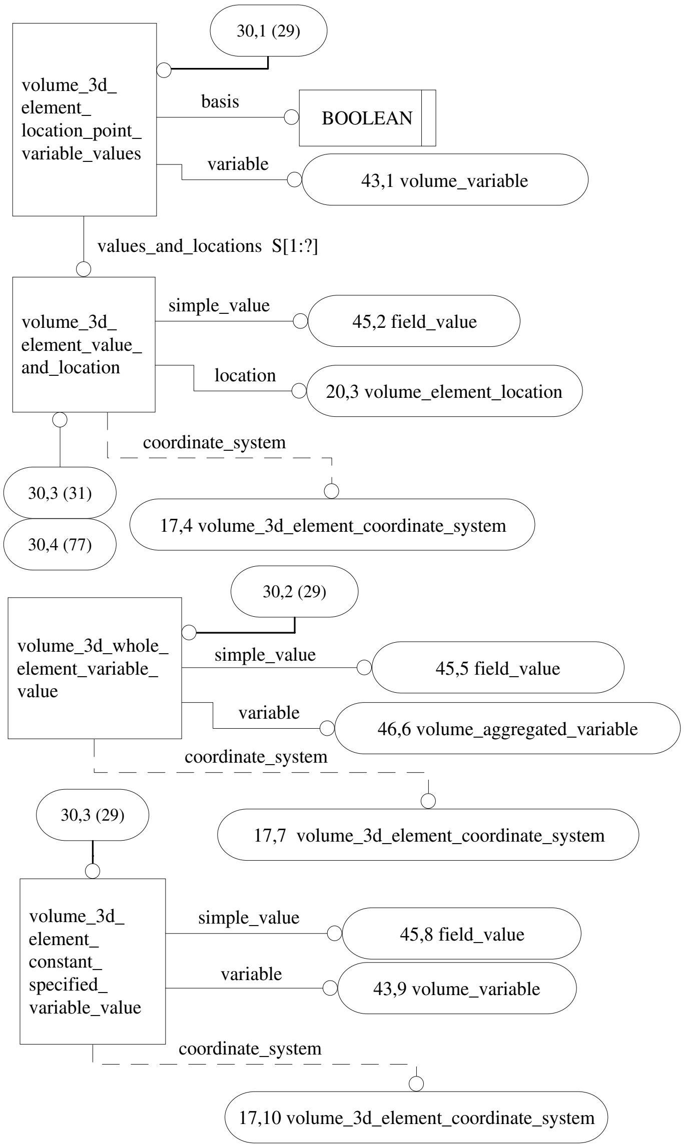

Figure H.30 - AIM EXPRESS-G diagram 30 of 94

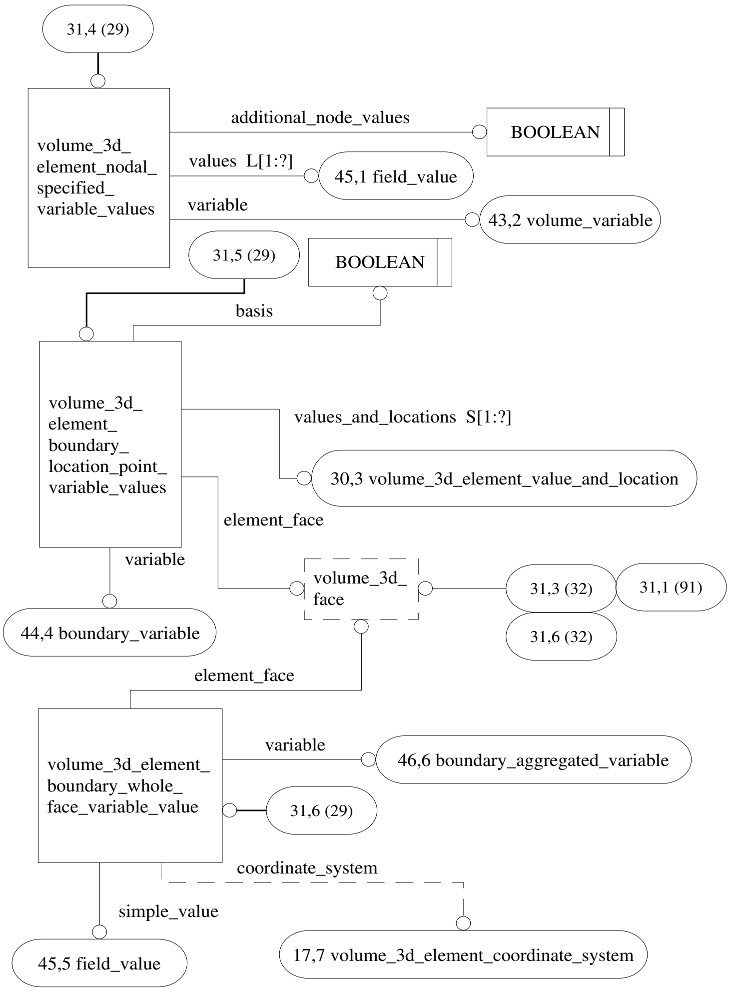

Figure H.31 - AIM EXPRESS-G diagram 31 of 94

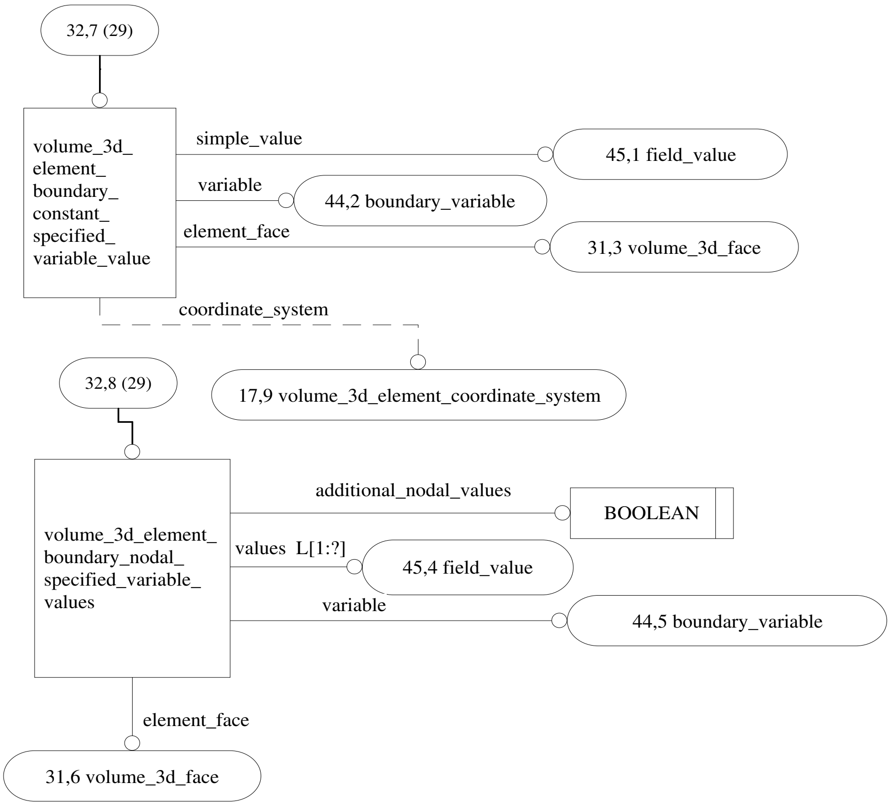

Figure H.32 - AIM EXPRESS-G diagram 32 of 94

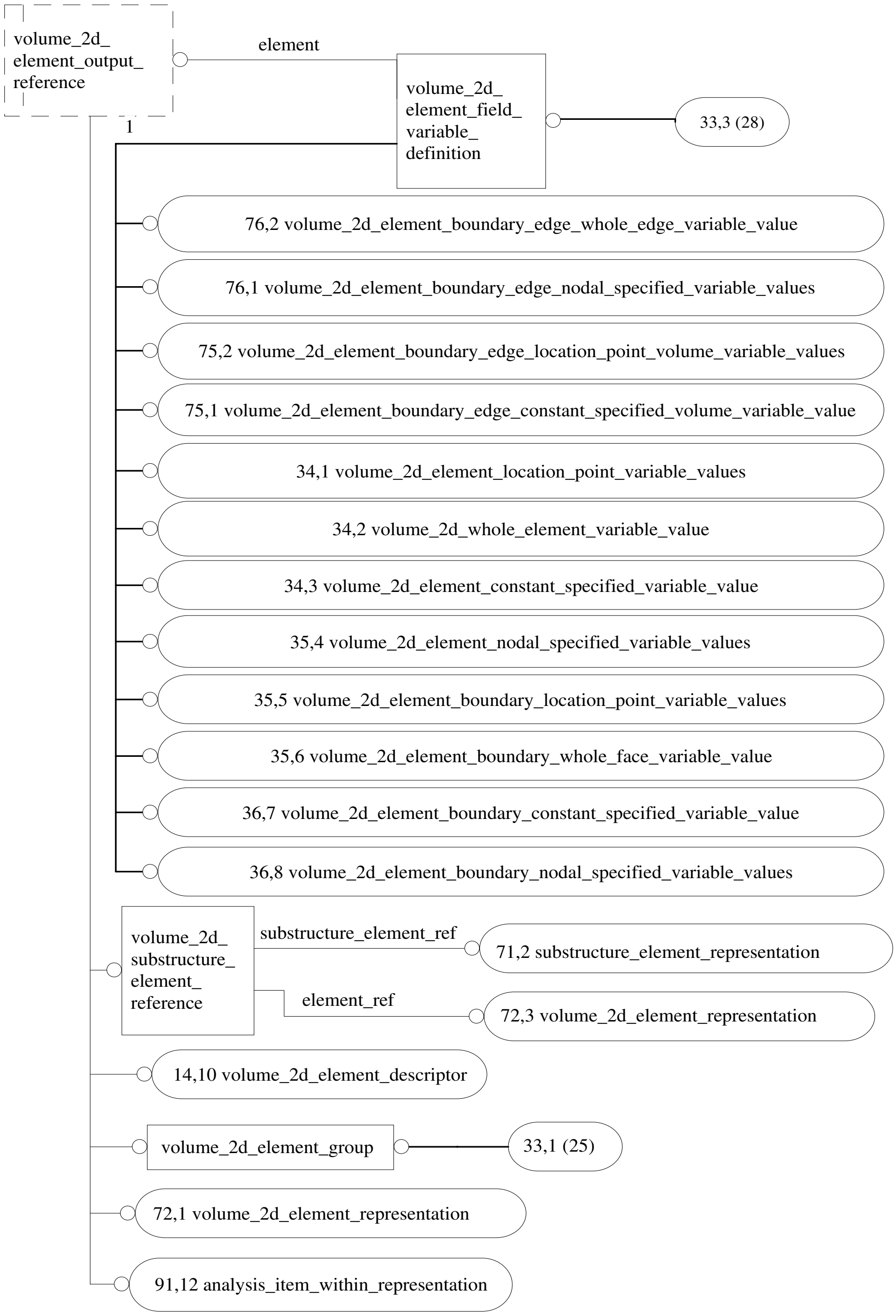

Figure H.33 - AIM EXPRESS-G diagram 33 of 94

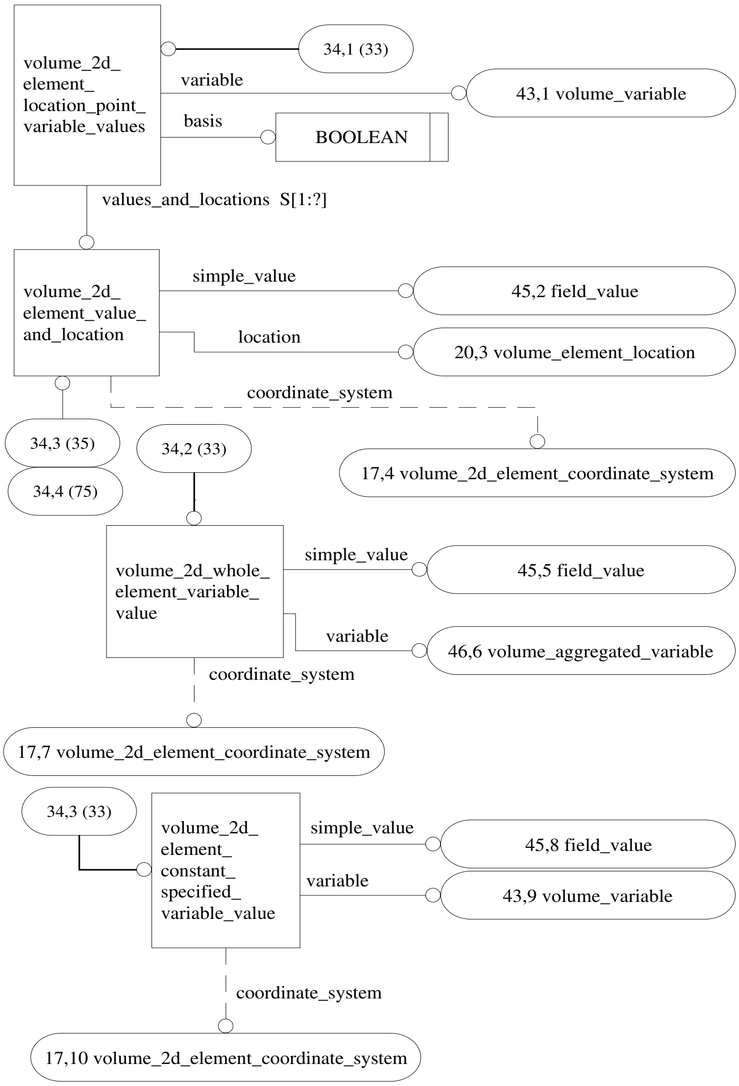

Figure H.34 - AIM EXPRESS-G diagram 34 of 94

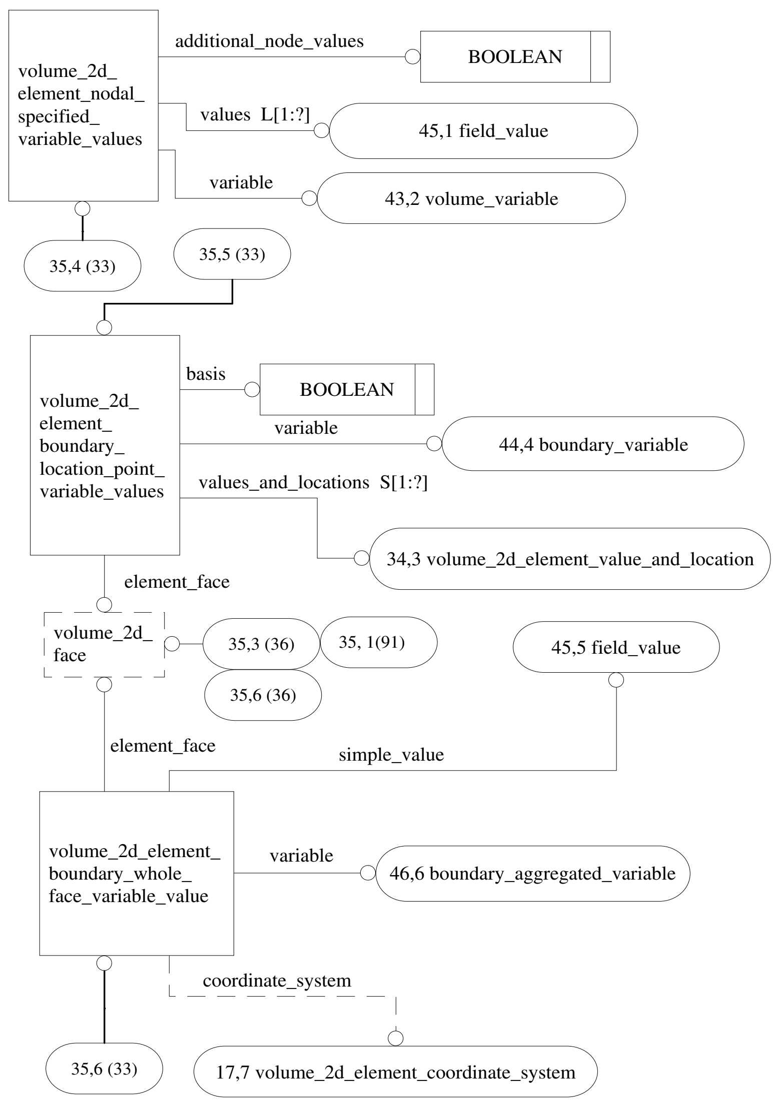

Figure H.35 - AIM EXPRESS-G diagram 35 of 94

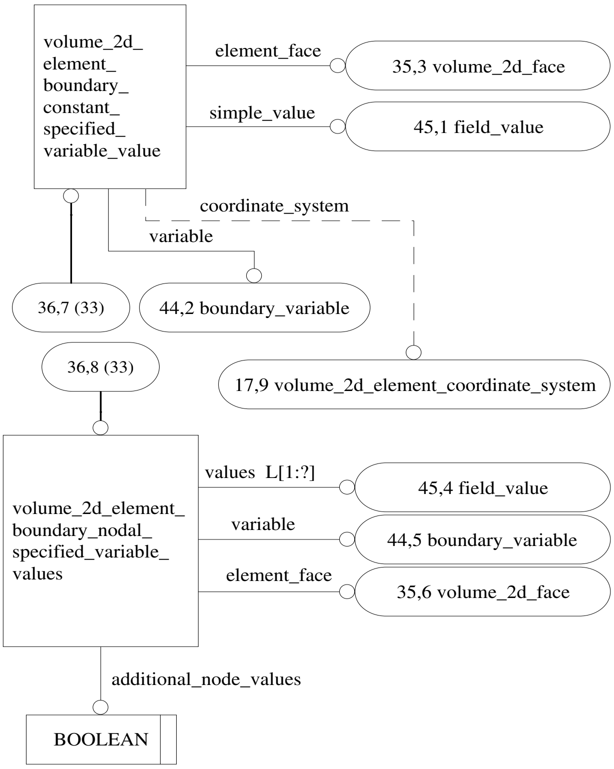

Figure H.36 - AIM EXPRESS-G diagram 36 of 94

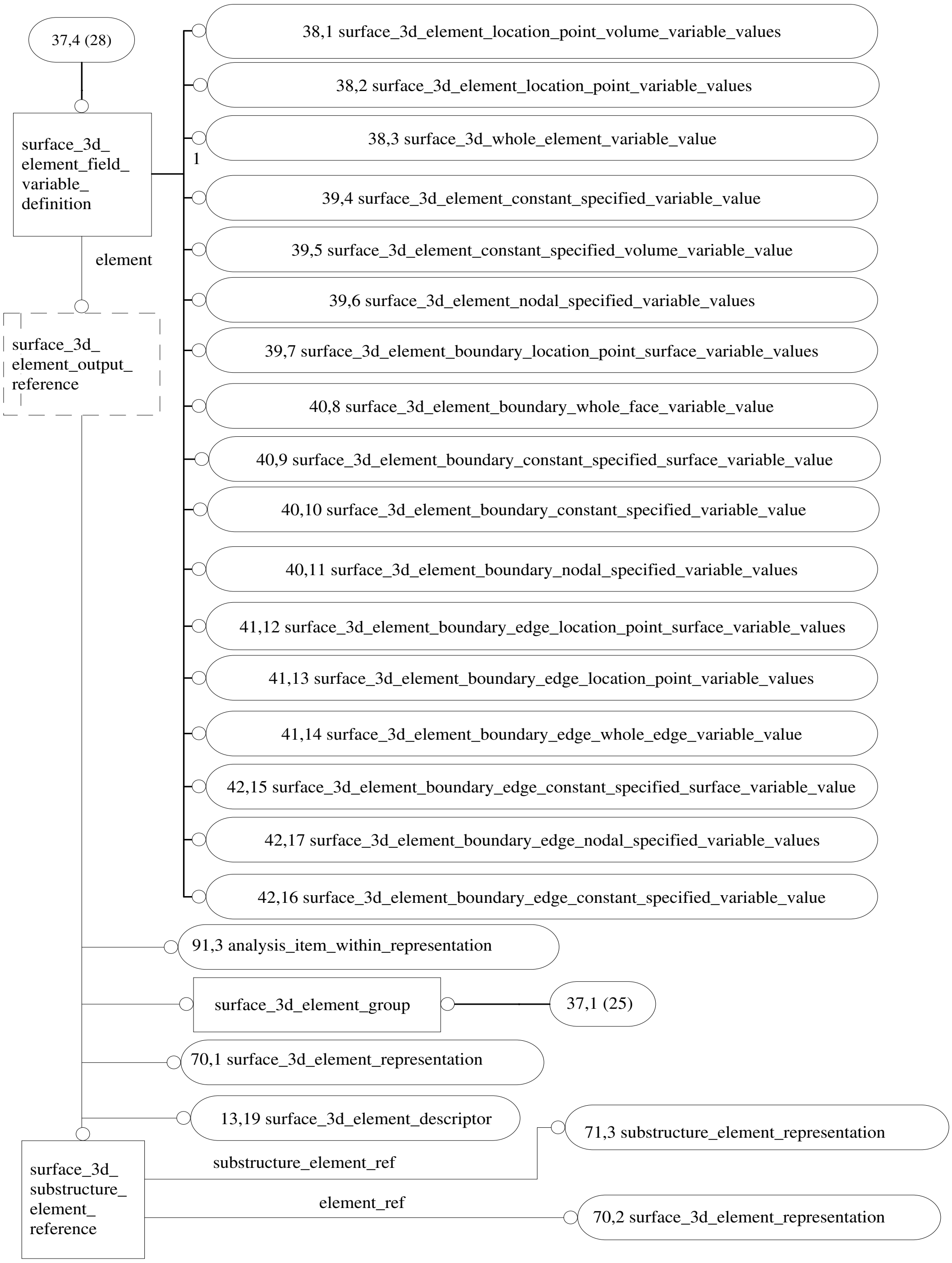

Figure H.37 - AIM EXPRESS-G diagram 37 of 94

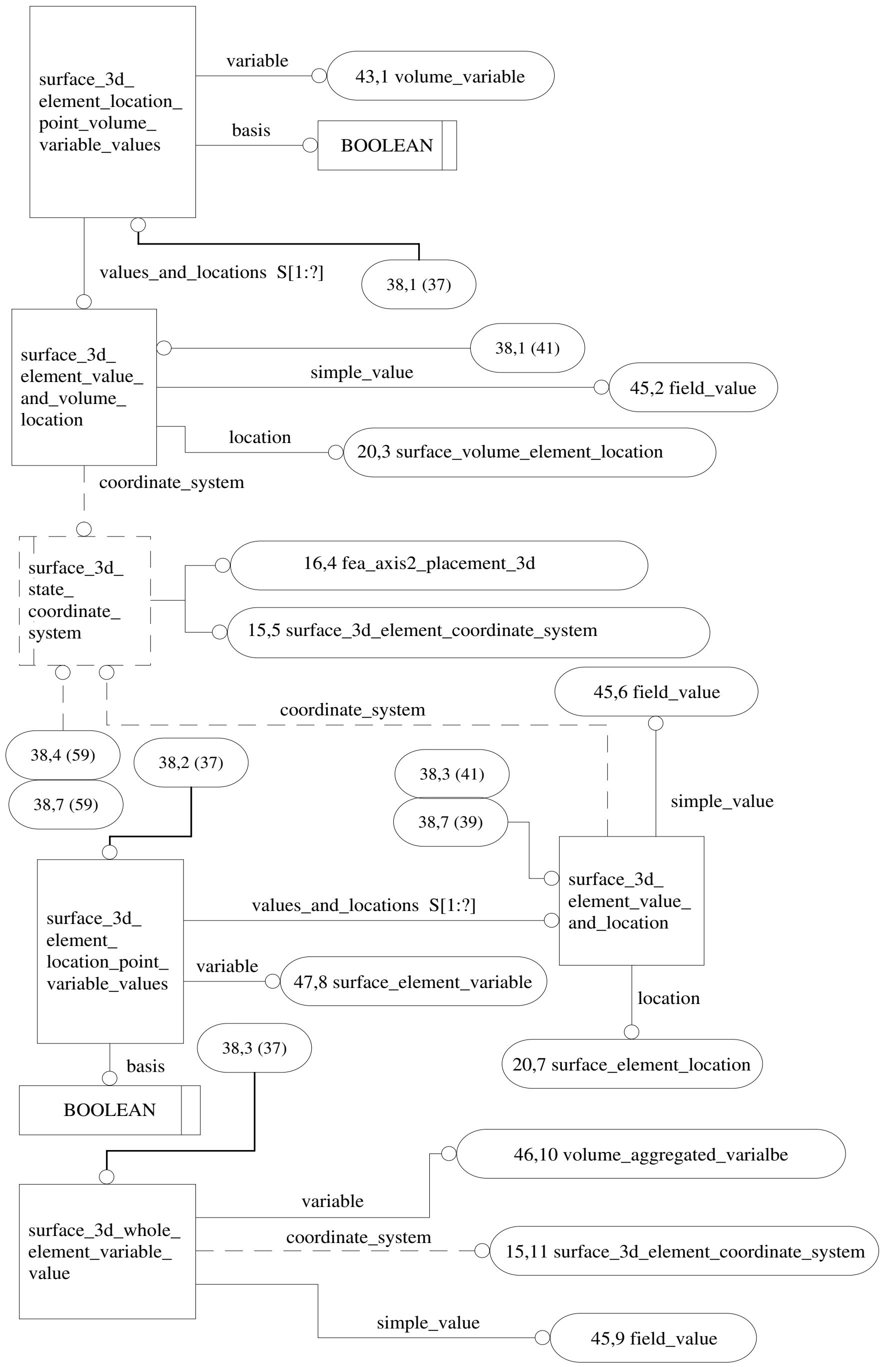

Figure H.38 - AIM EXPRESS-G diagram 38 of 94

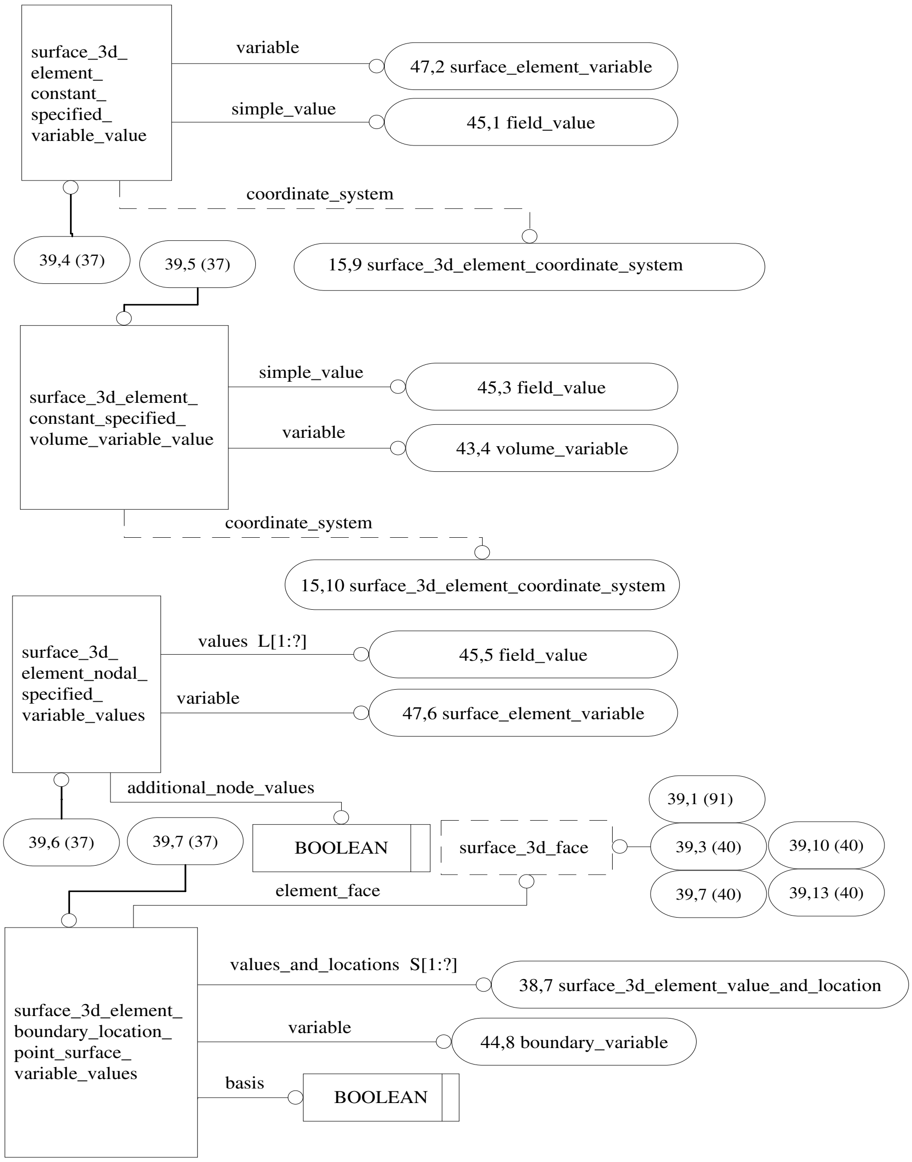

Figure H.39 - AIM EXPRESS-G diagram 39 of 94

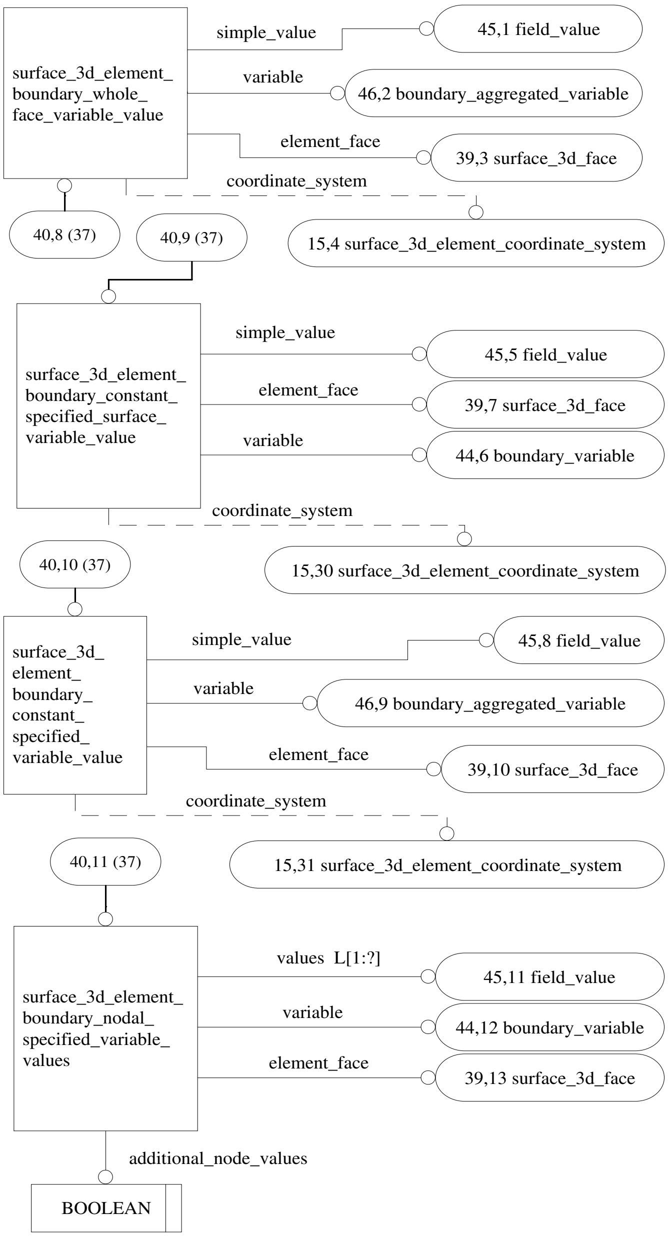

Figure H.40 - AIM EXPRESS-G diagram 40 of 94

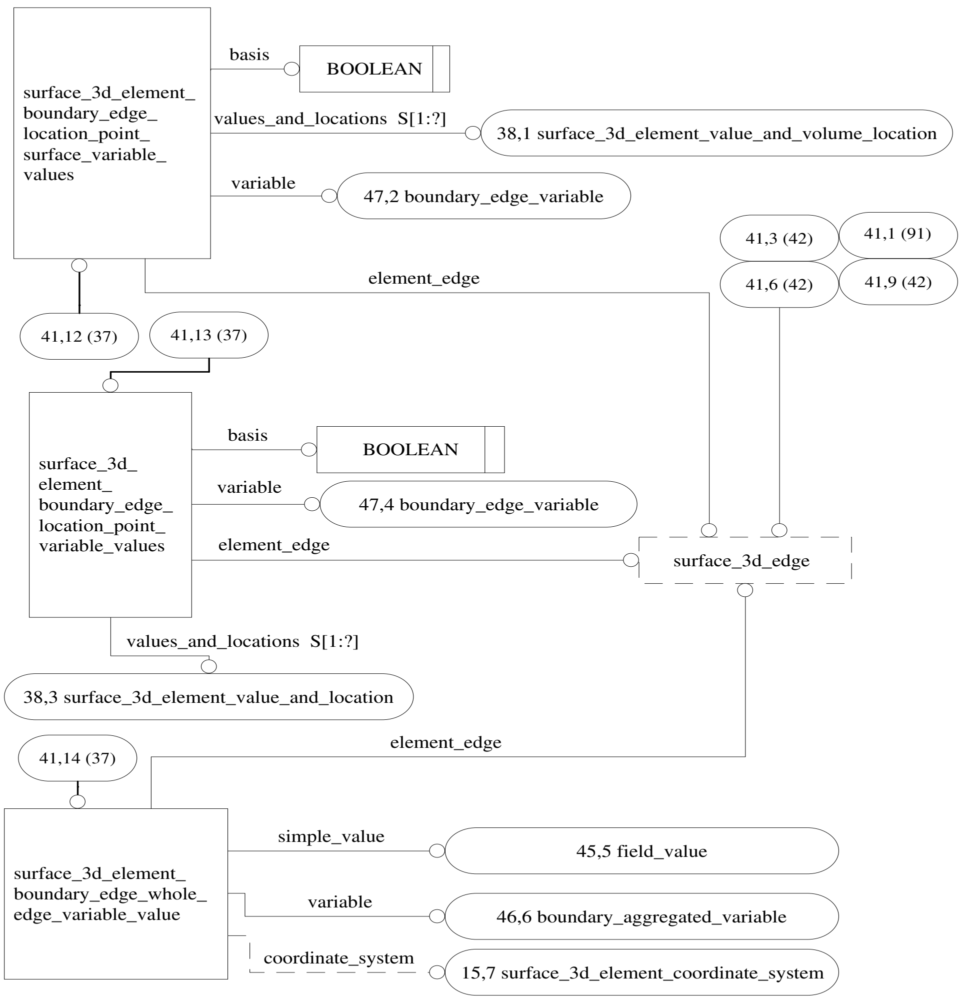

Figure H.41 - AIM EXPRESS-G diagram 41 of 94

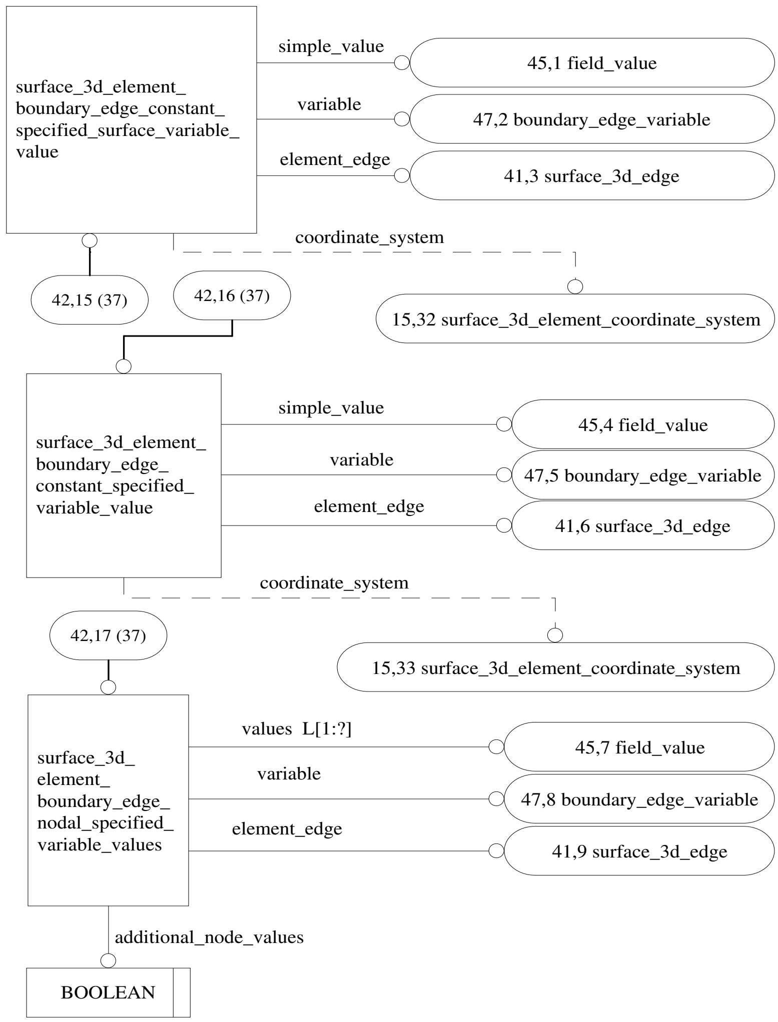

Figure H.42 - AIM EXPRESS-G diagram 42 of 94

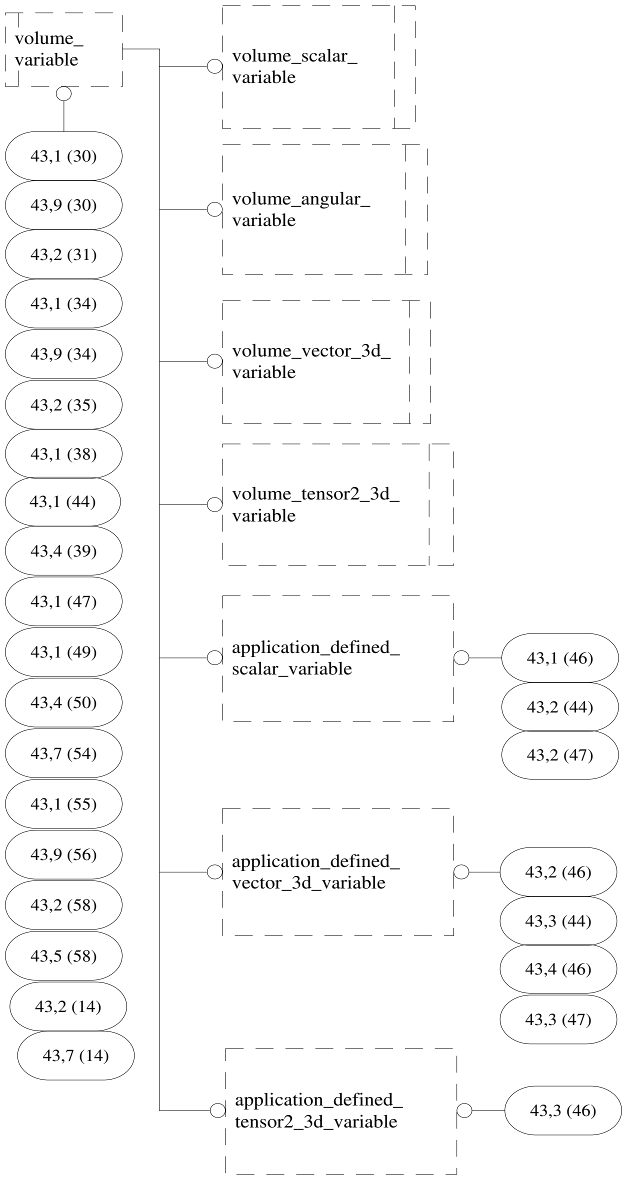

Figure H.43 - AIM EXPRESS-G diagram 43 of 94

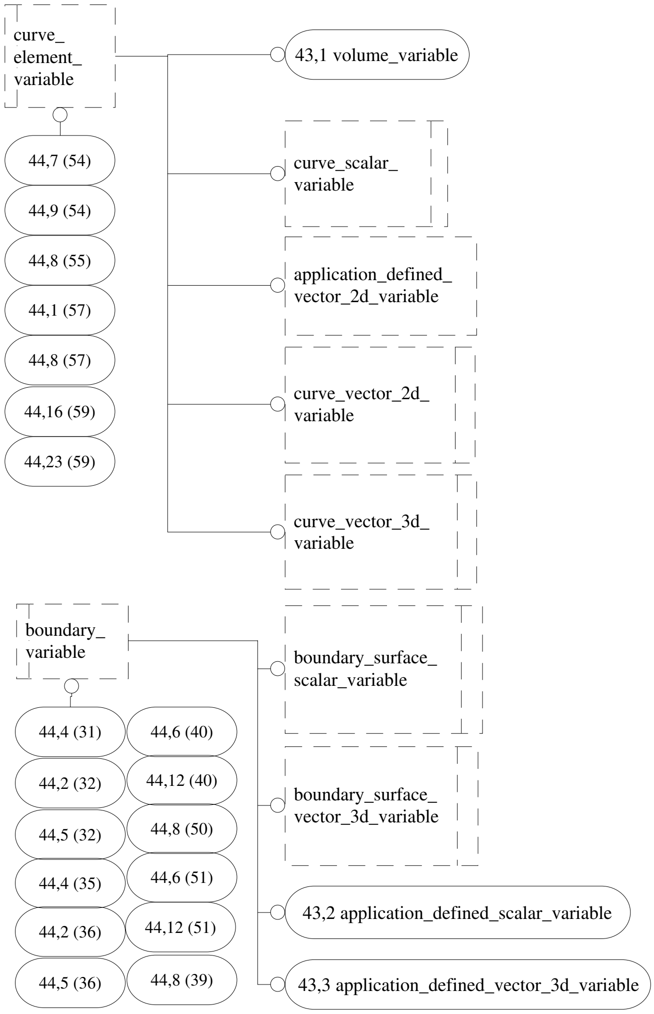

Figure H.44 - AIM EXPRESS-G diagram 44 of 94

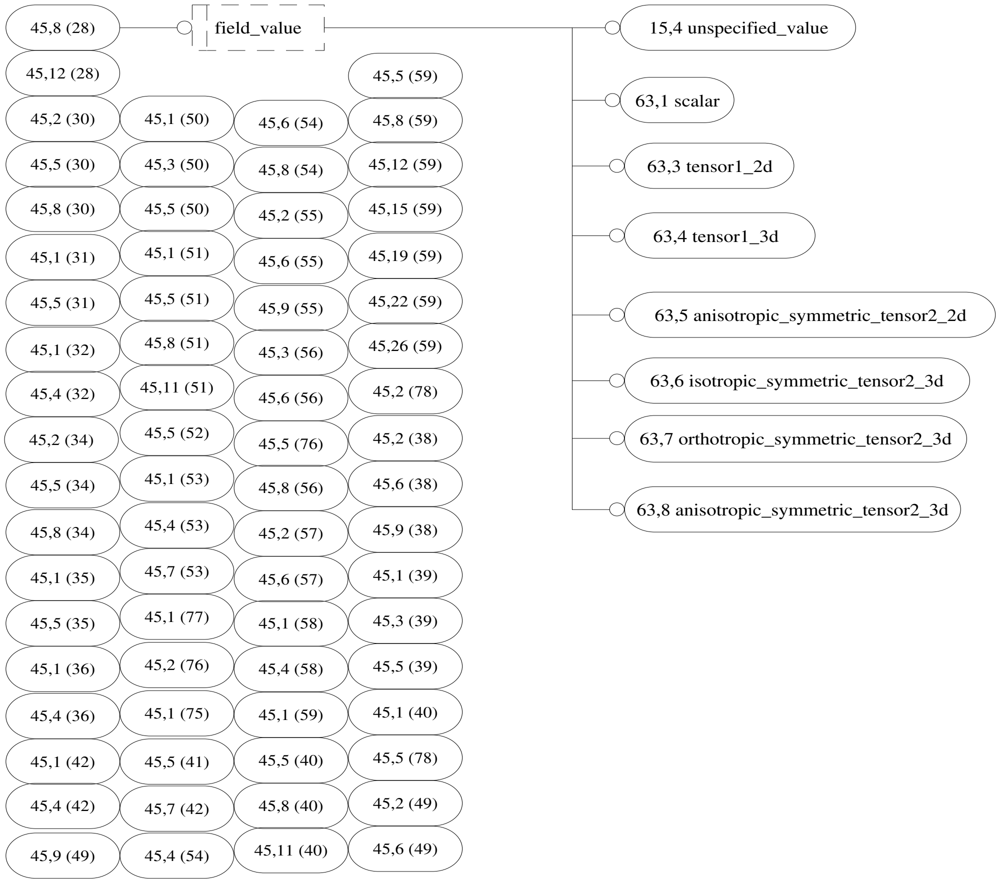

Figure H.45 - AIM EXPRESS-G diagram 45 of 94

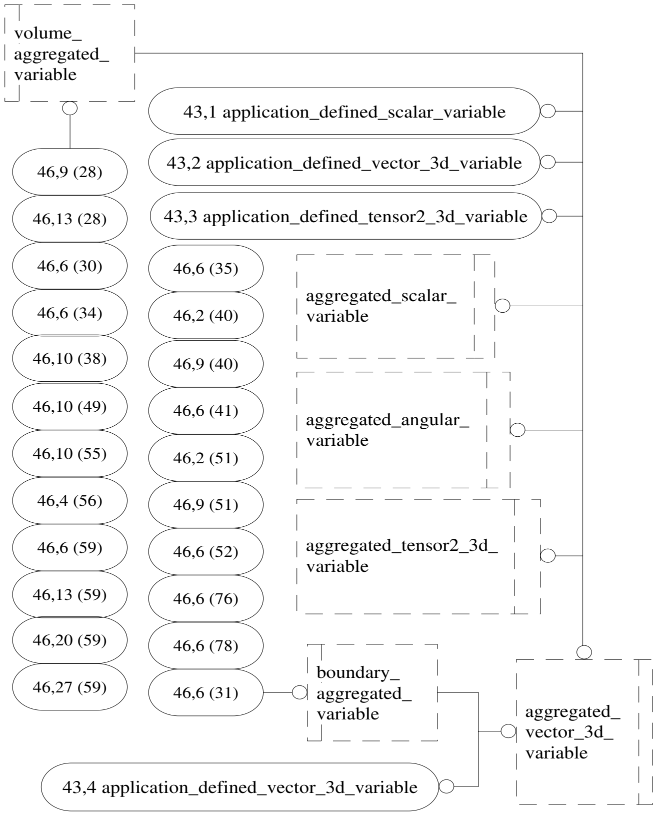

Figure H.46 - AIM EXPRESS-G diagram 46 of 94

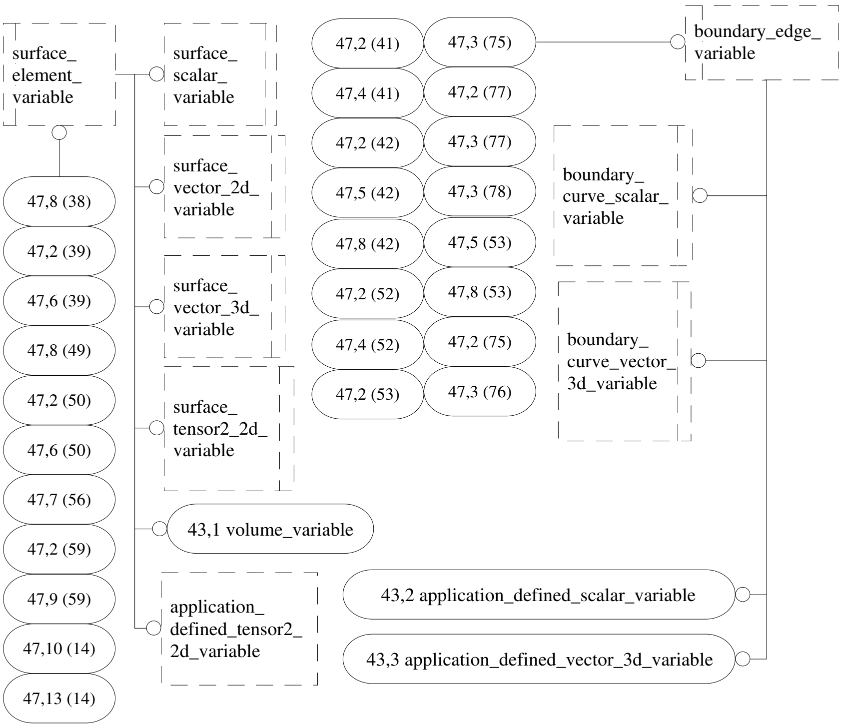

Figure H.47 - AIM EXPRESS-G diagram 47 of 94

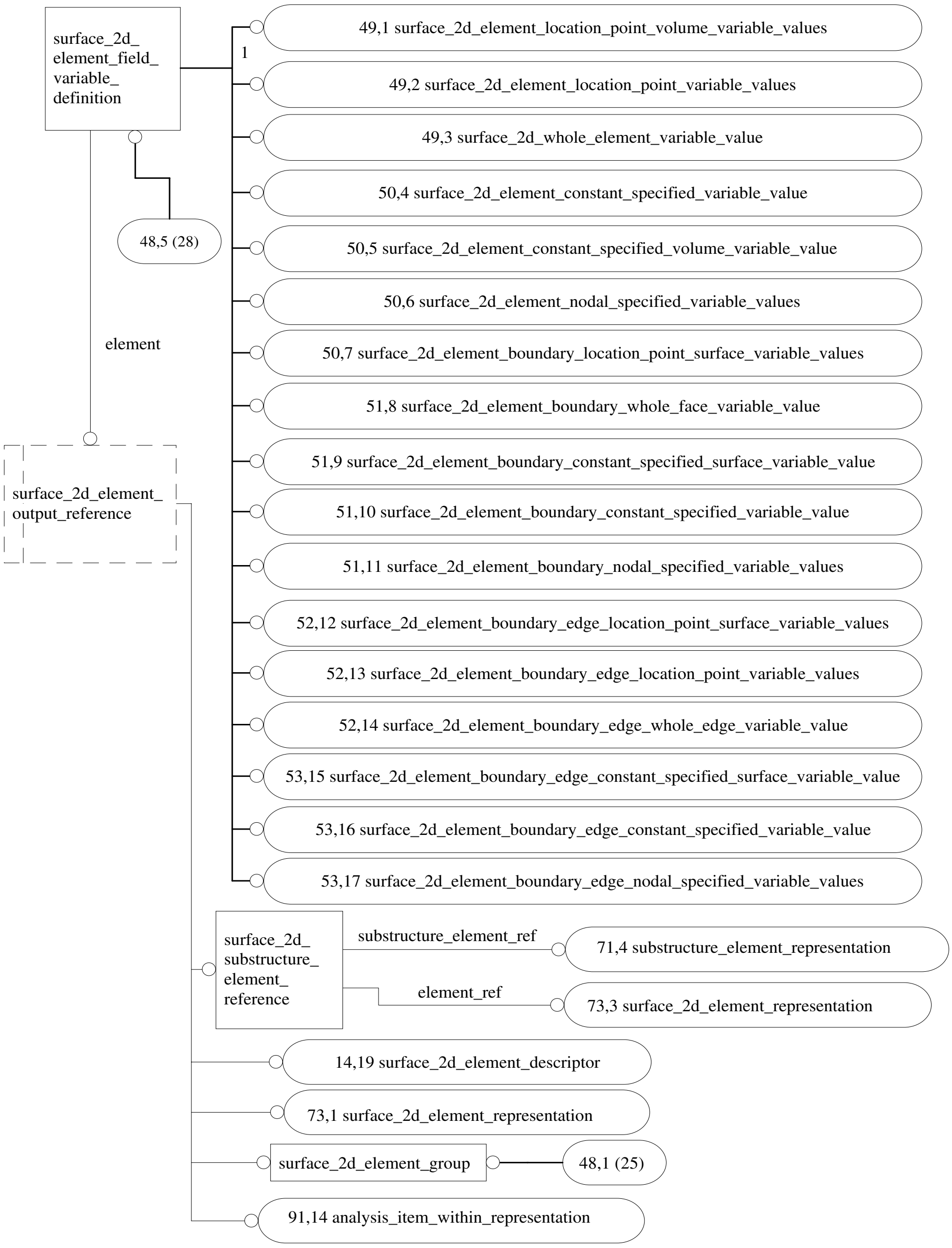

Figure H.48 - AIM EXPRESS-G diagram 48 of 94

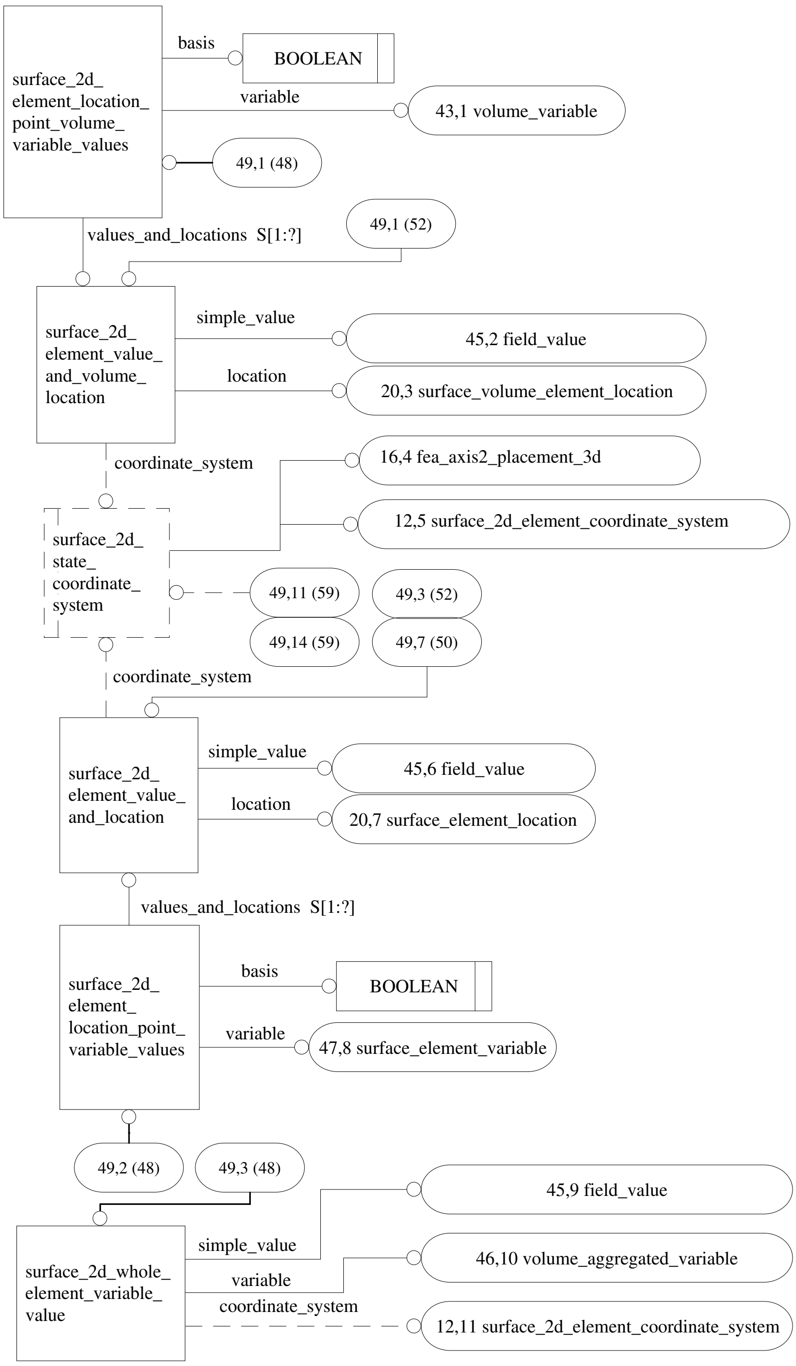

Figure H.49 - AIM EXPRESS-G diagram 49 of 94

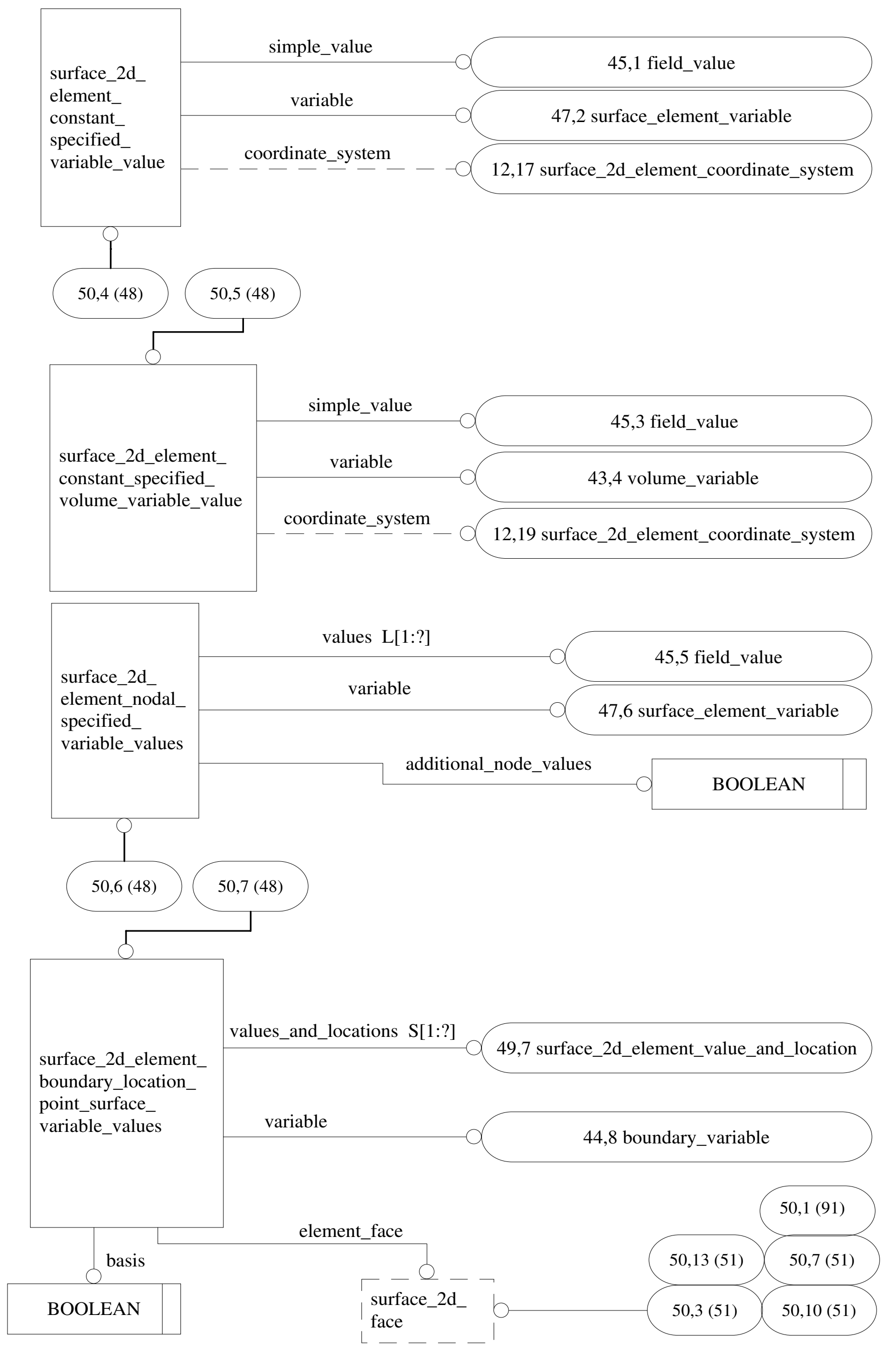

Figure H.50 - AIM EXPRESS-G diagram 50 of 94

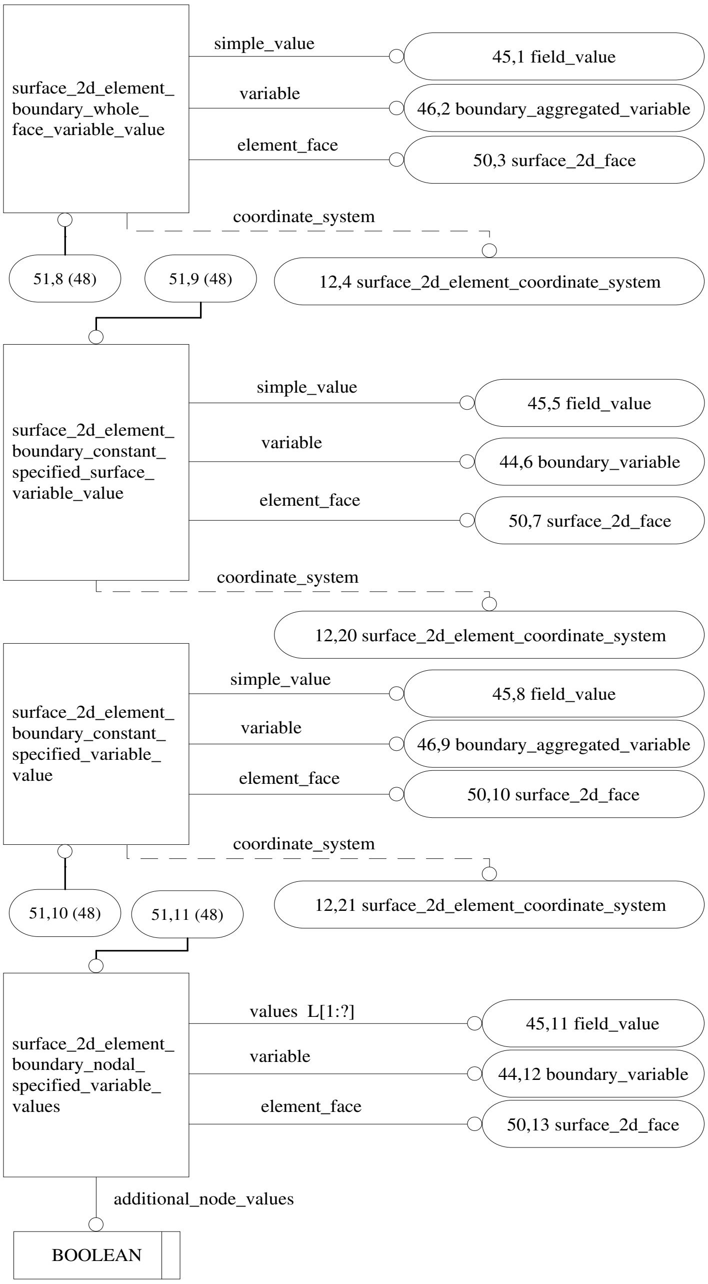

Figure H.51 - AIM EXPRESS-G diagram 51 of 94

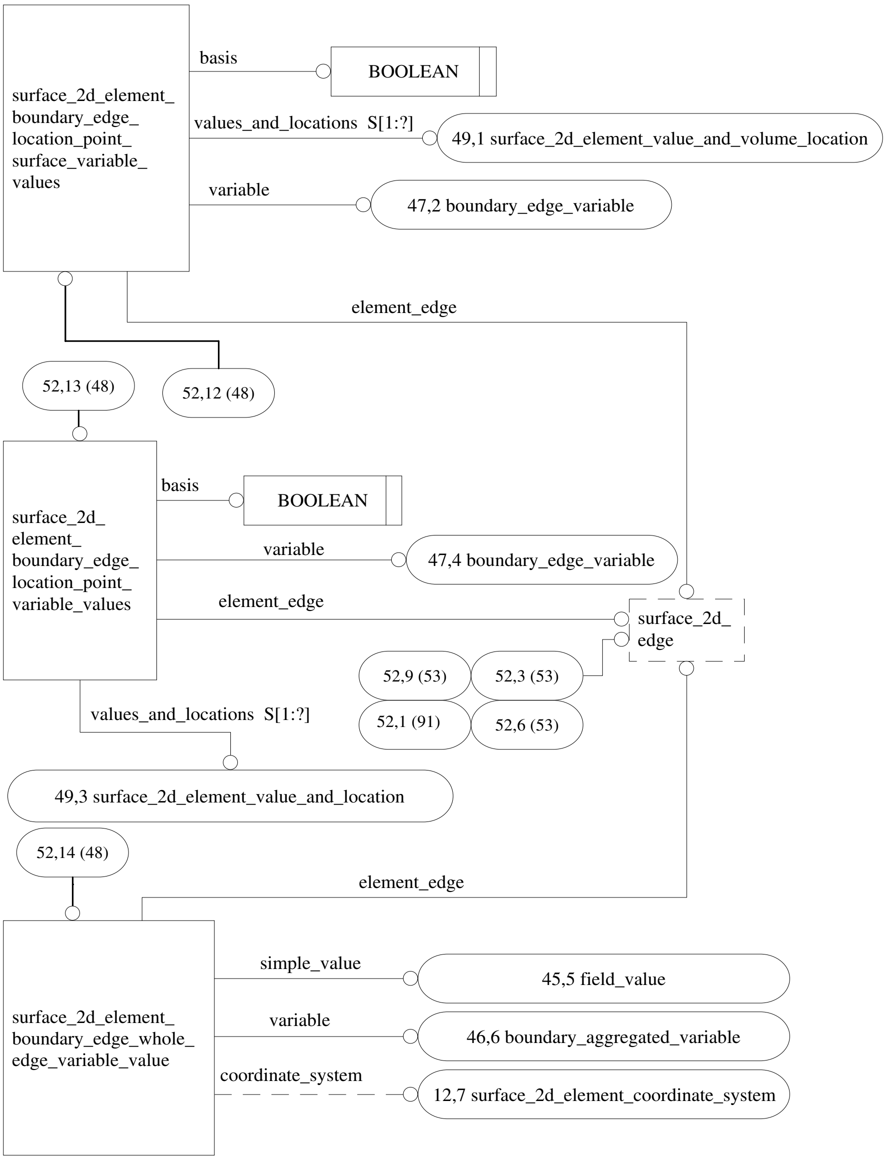

Figure H.52 - AIM EXPRESS-G diagram 52 of 94

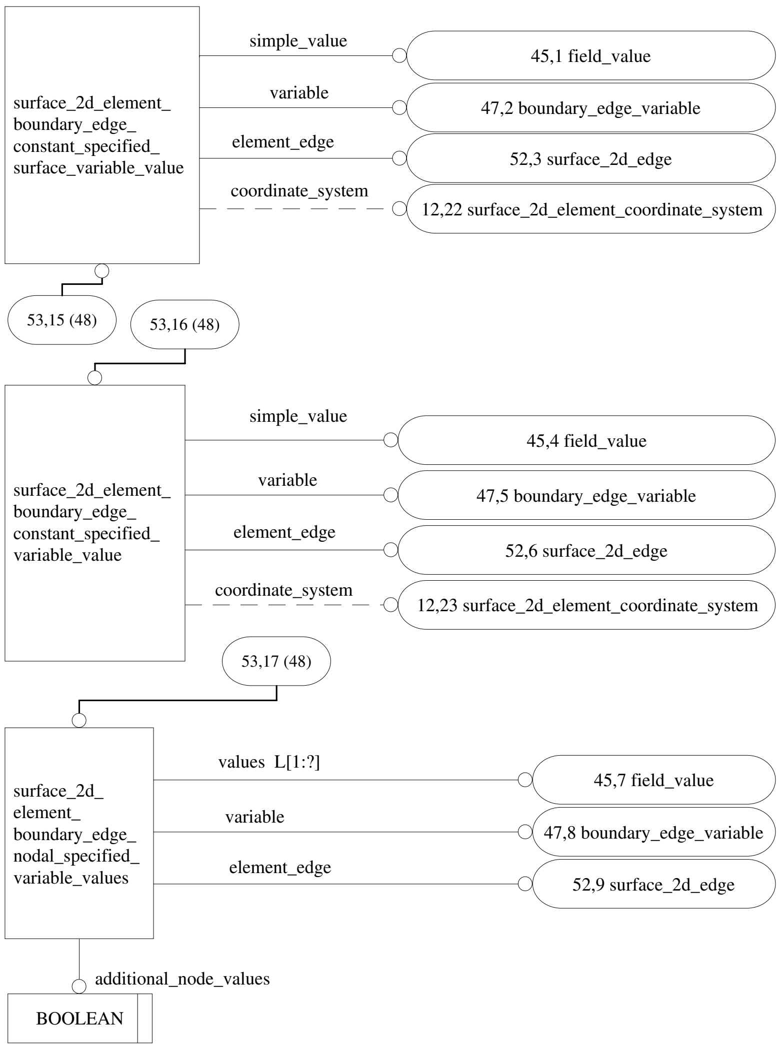

Figure H.53 - AIM EXPRESS-G diagram 53 of 94

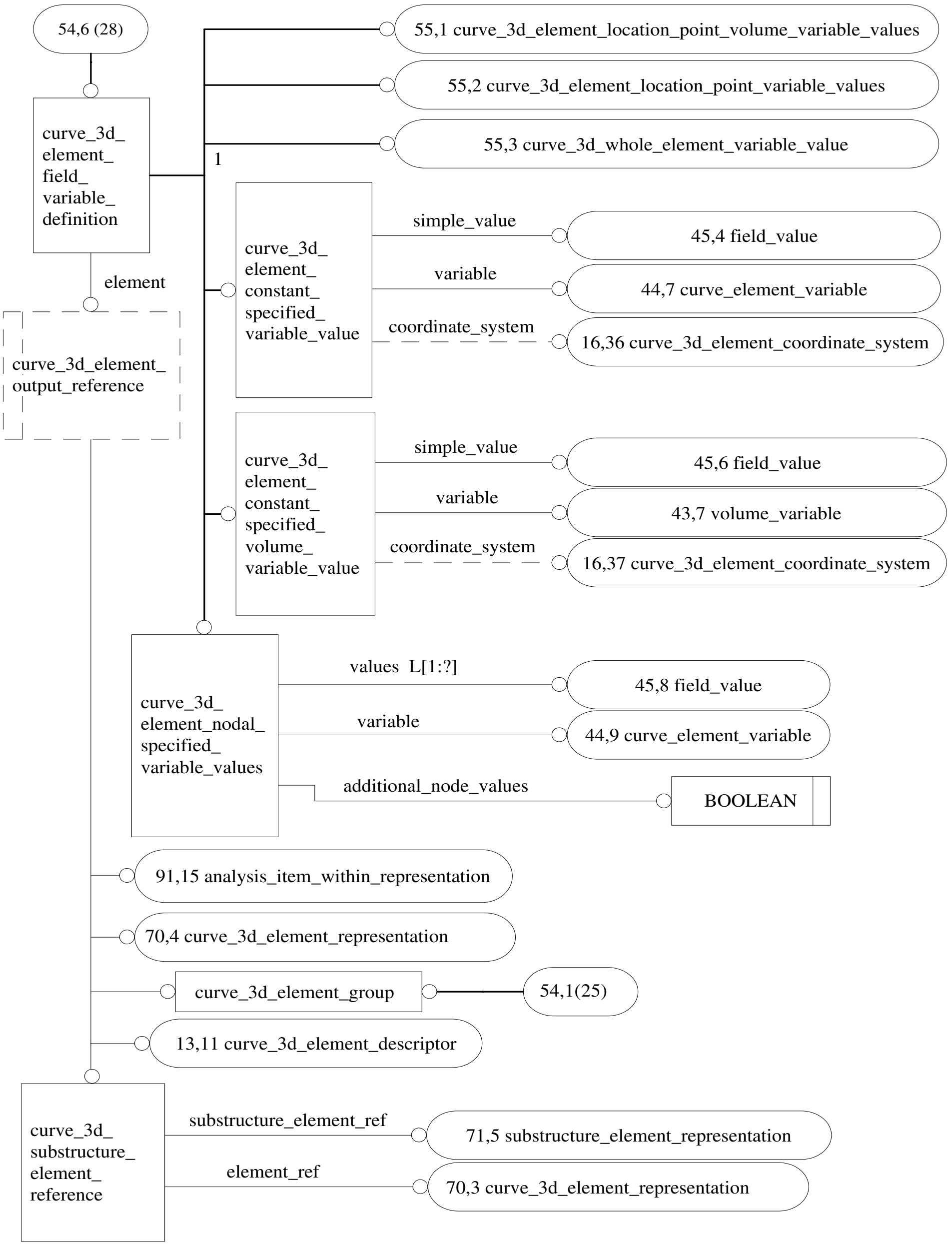

Figure H.54 - AIM EXPRESS-G diagram 54 of 94

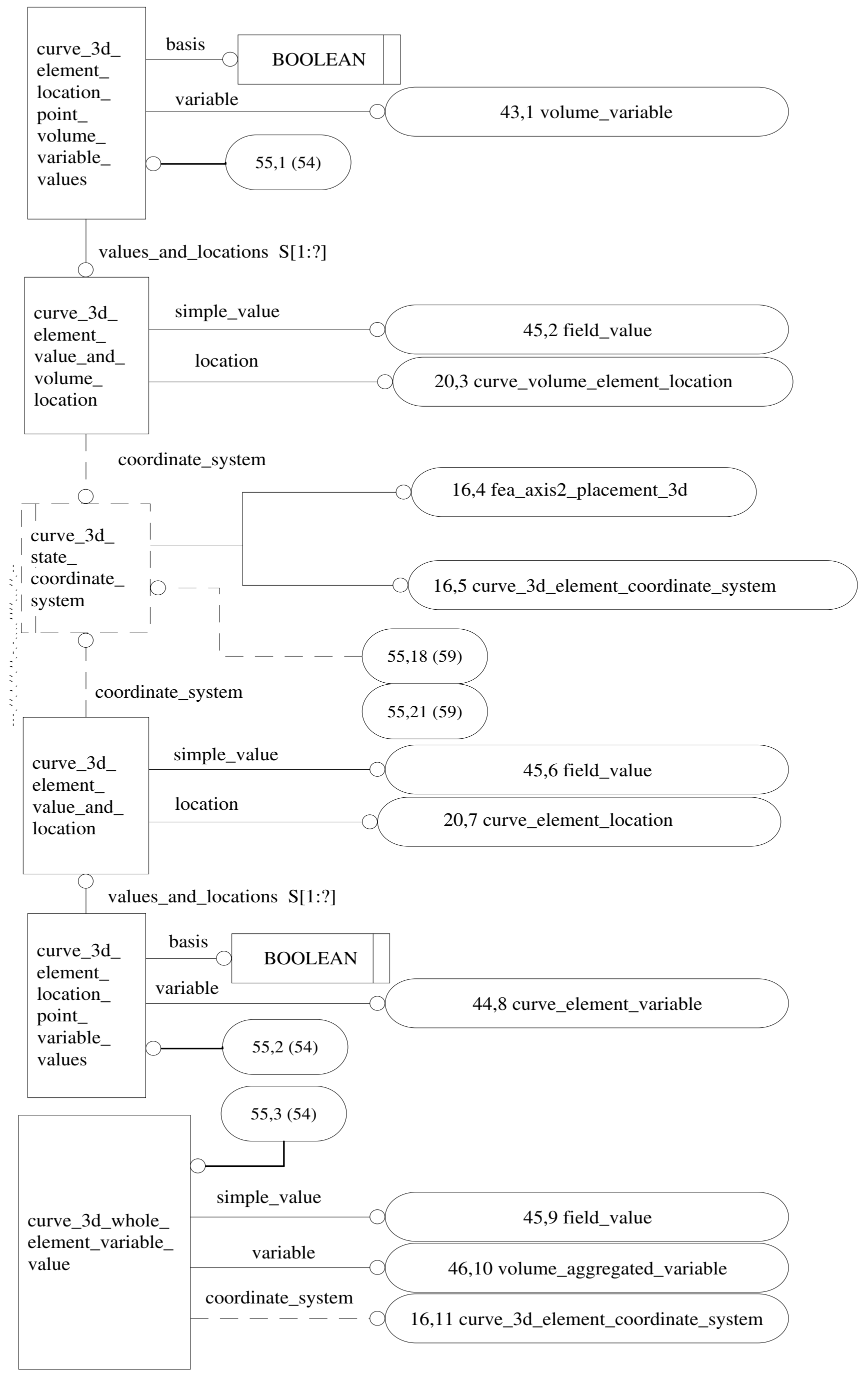

Figure H.55 - AIM EXPRESS-G diagram 55 of 94

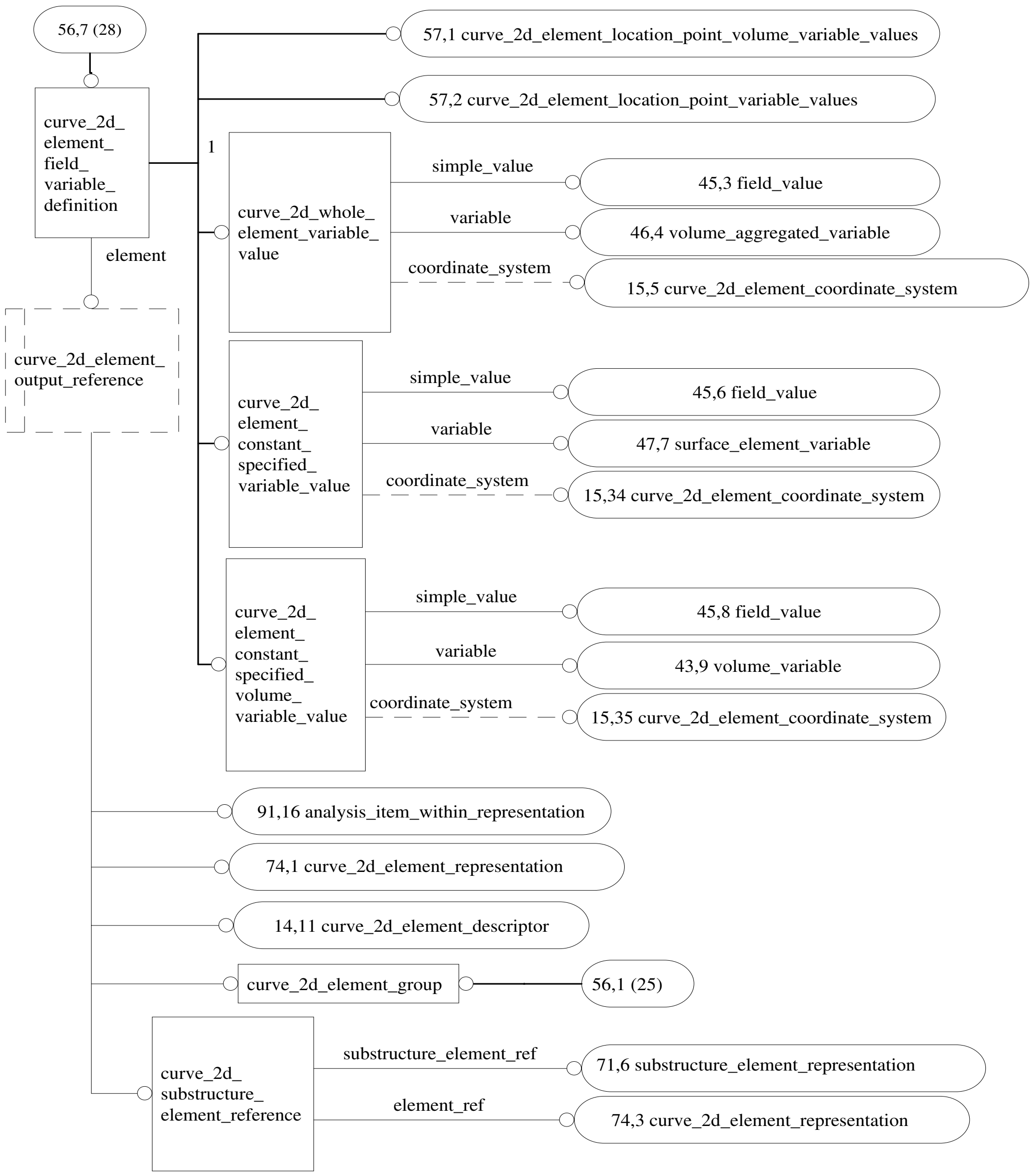

Figure H.56 - AIM EXPRESS-G diagram 56 of 94

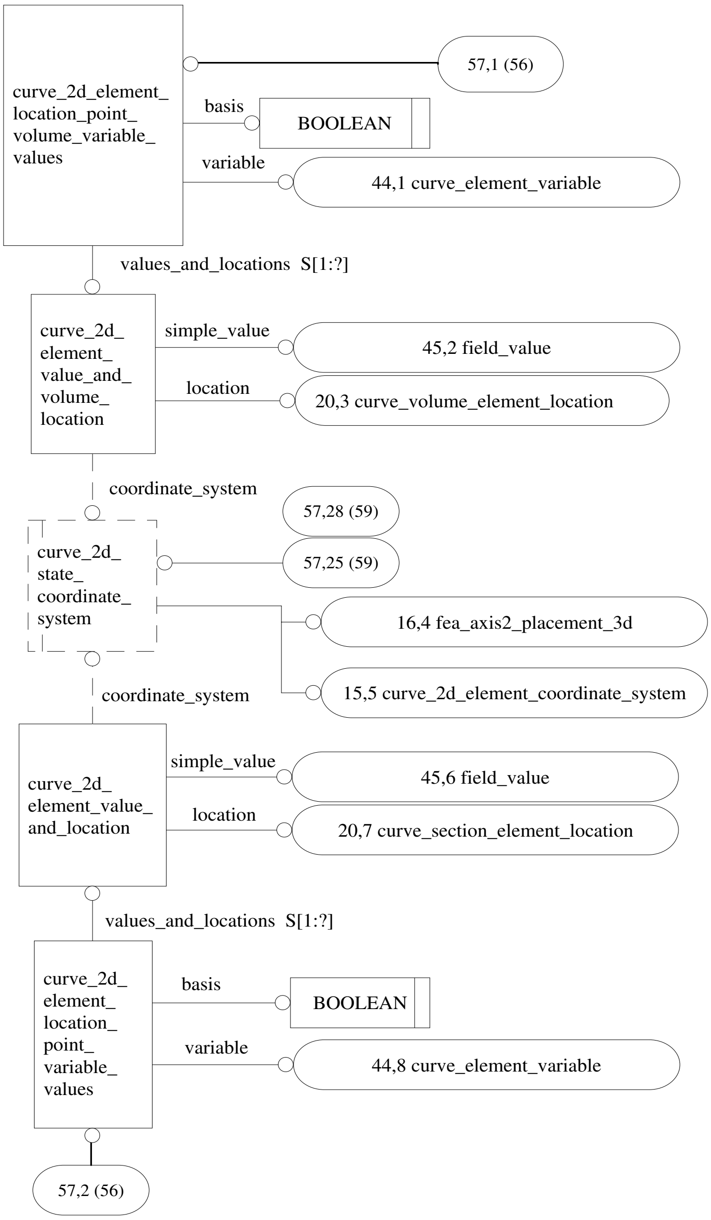

Figure H.57 - AIM EXPRESS-G diagram 57 of 94

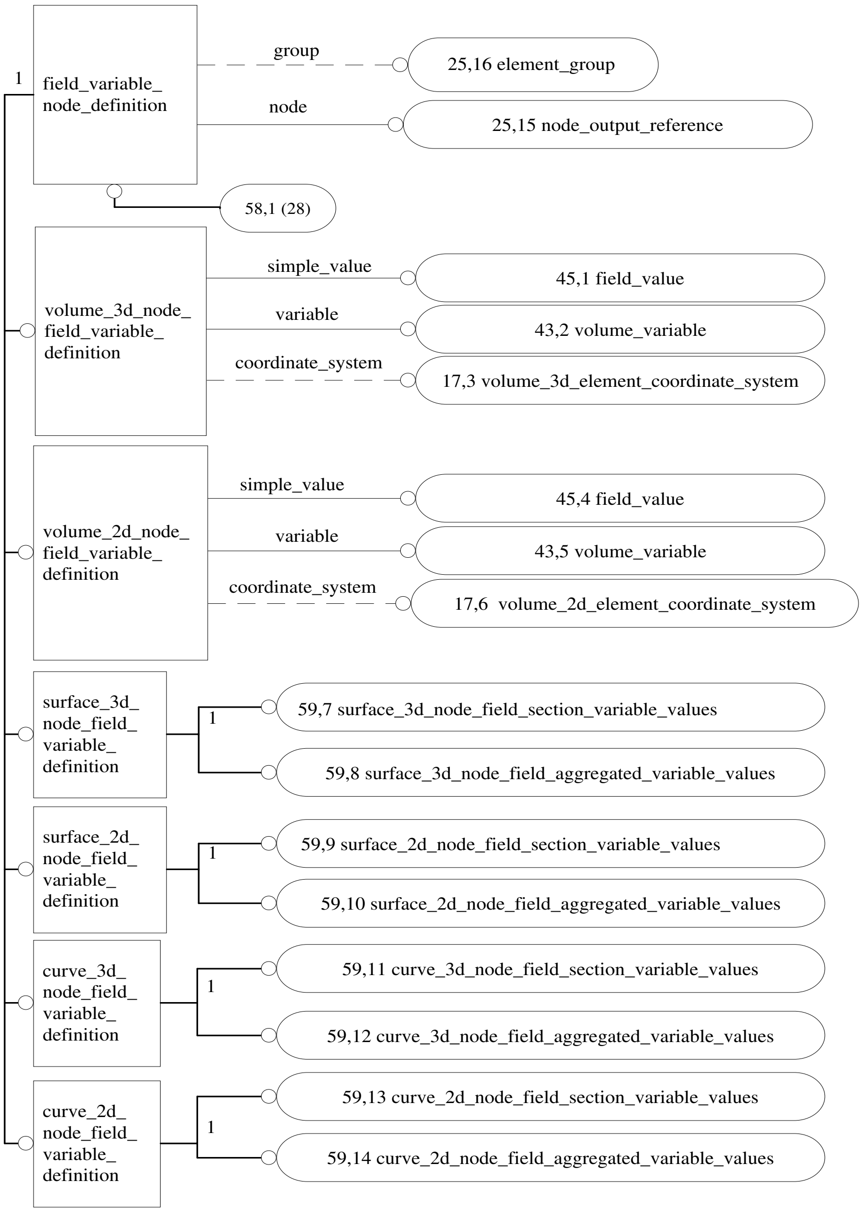

Figure H.58 - AIM EXPRESS-G diagram 58 of 94

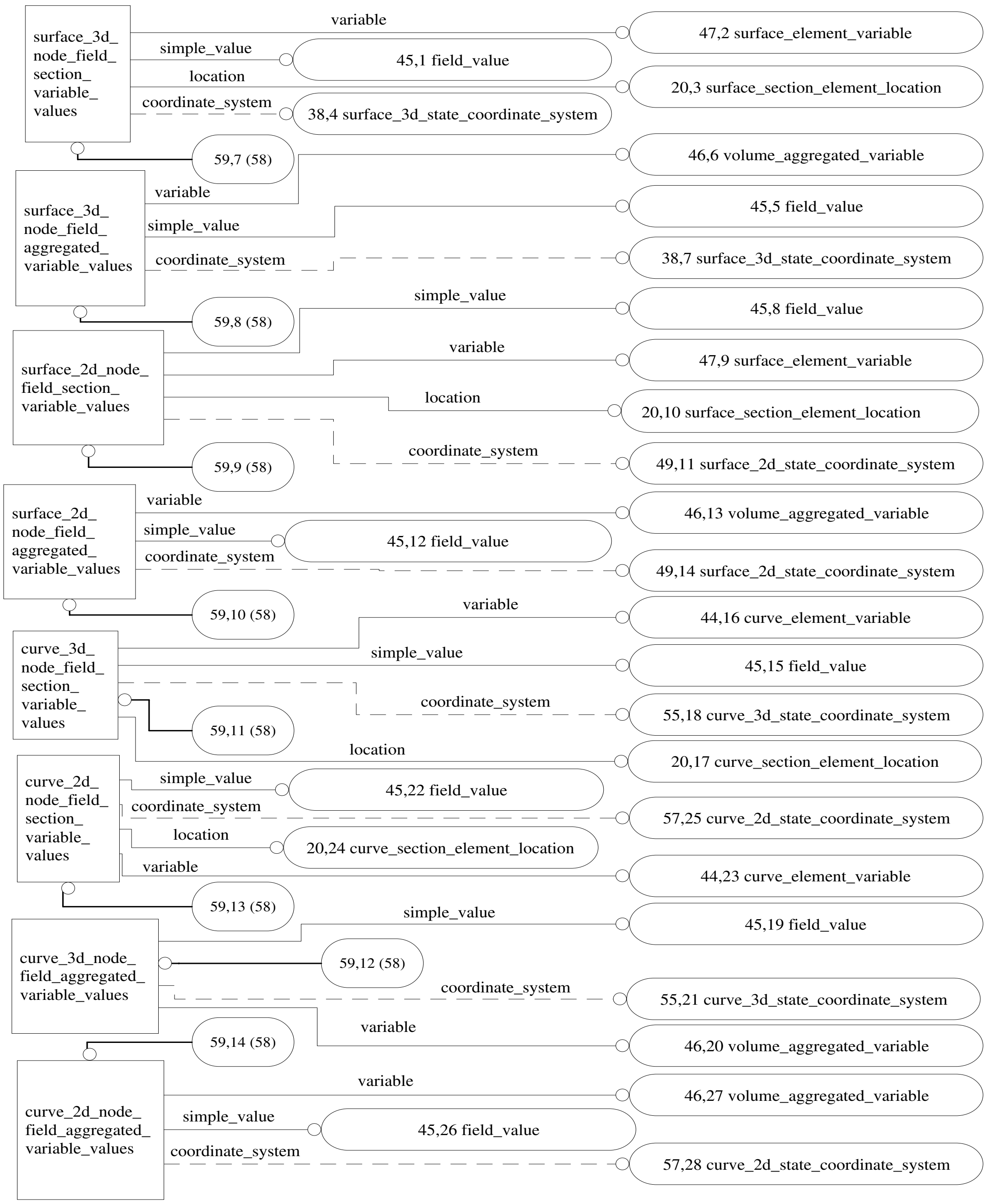

Figure H.59 - AIM EXPRESS-G diagram 59 of 94

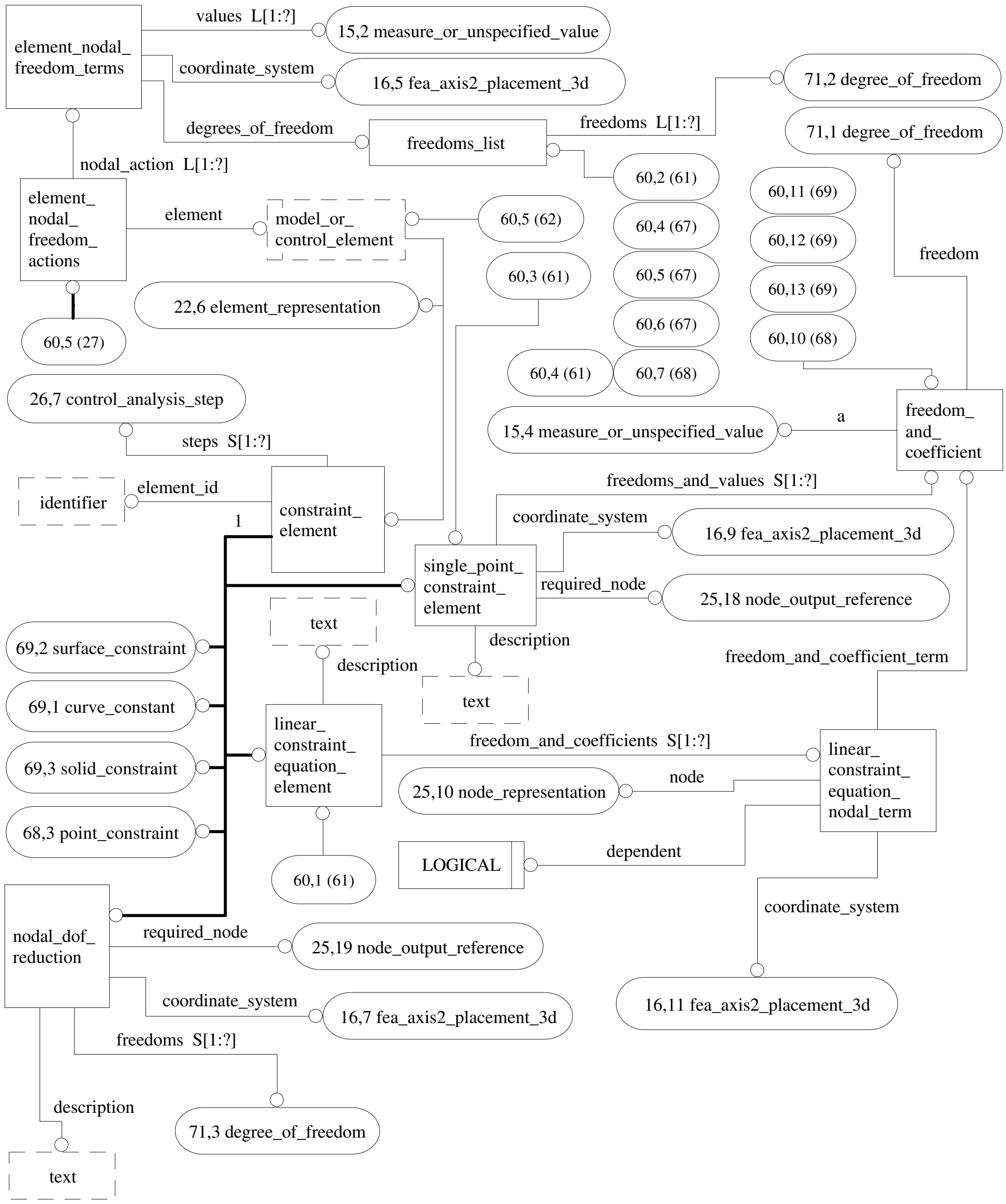

Figure H.60 - AIM EXPRESS-G diagram 60 of 94

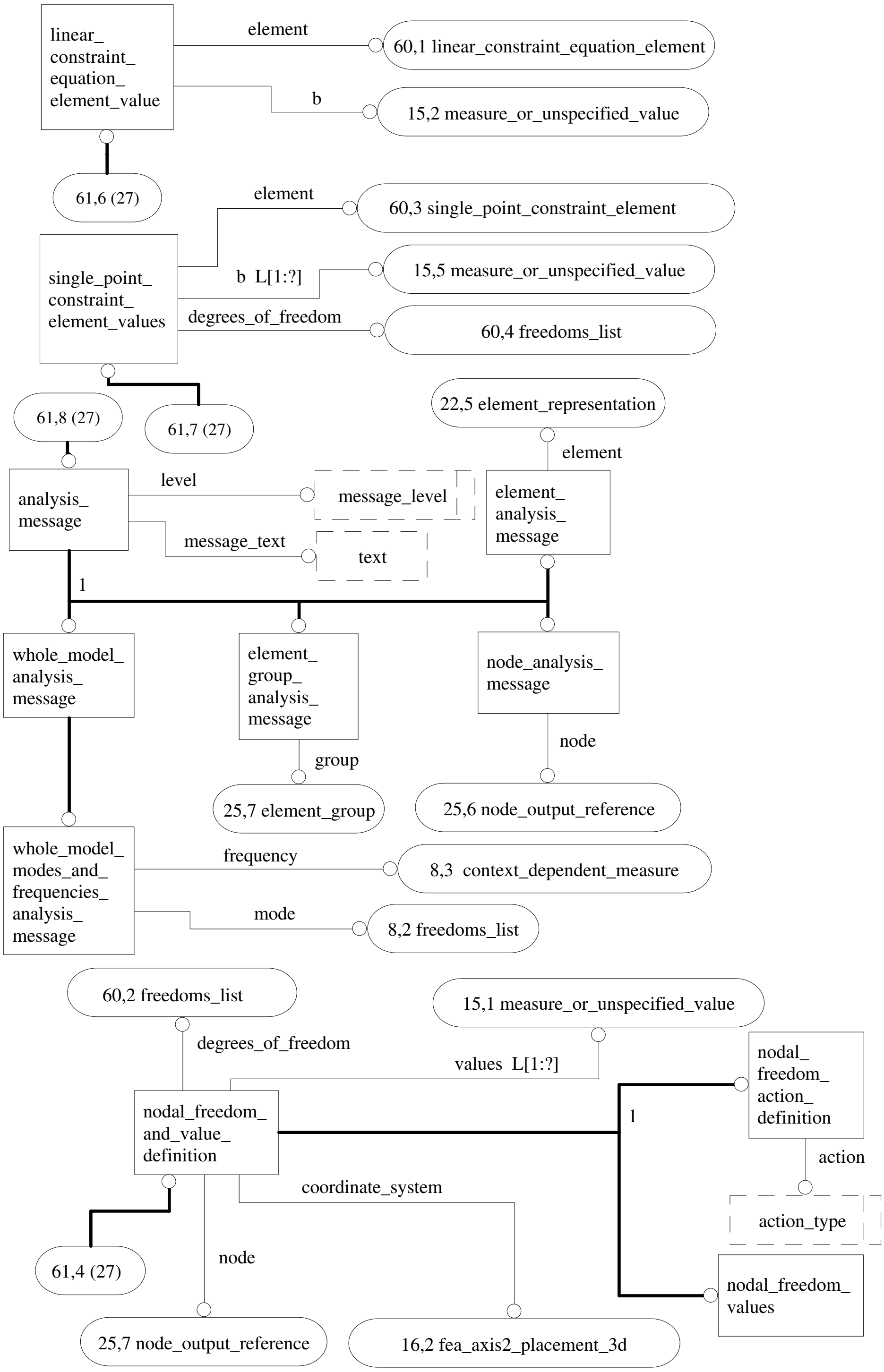

Figure H.61 - AIM EXPRESS-G diagram 61 of 94

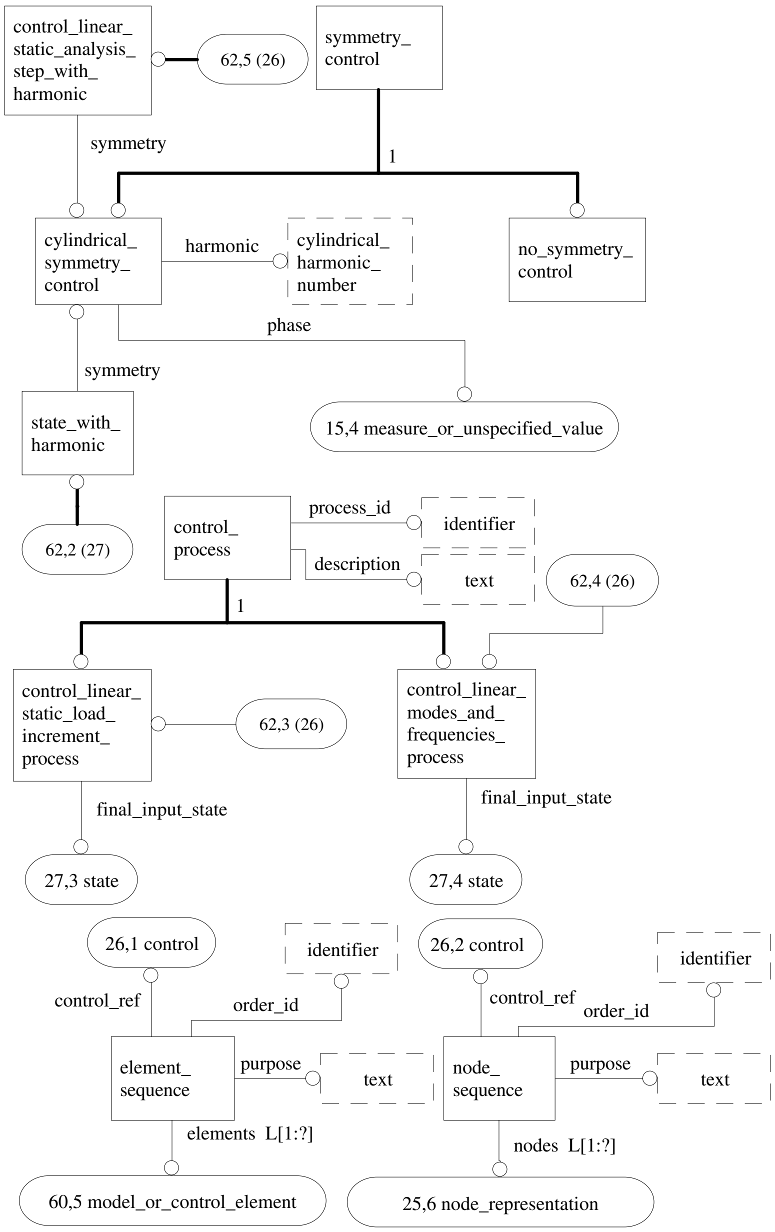

Figure H.62 - AIM EXPRESS-G diagram 62 of 94

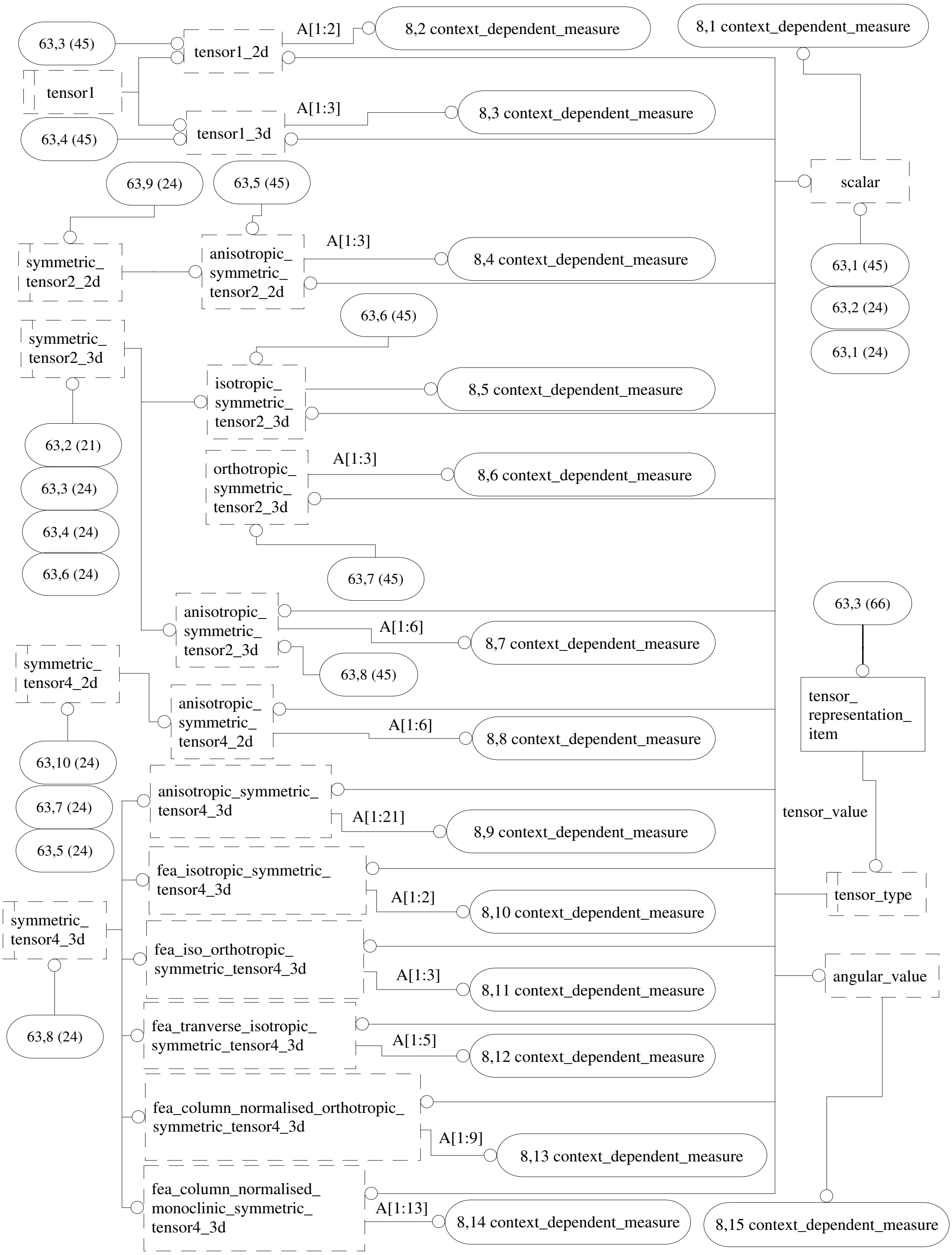

Figure H.63 - AIM EXPRESS-G diagram 63 of 94

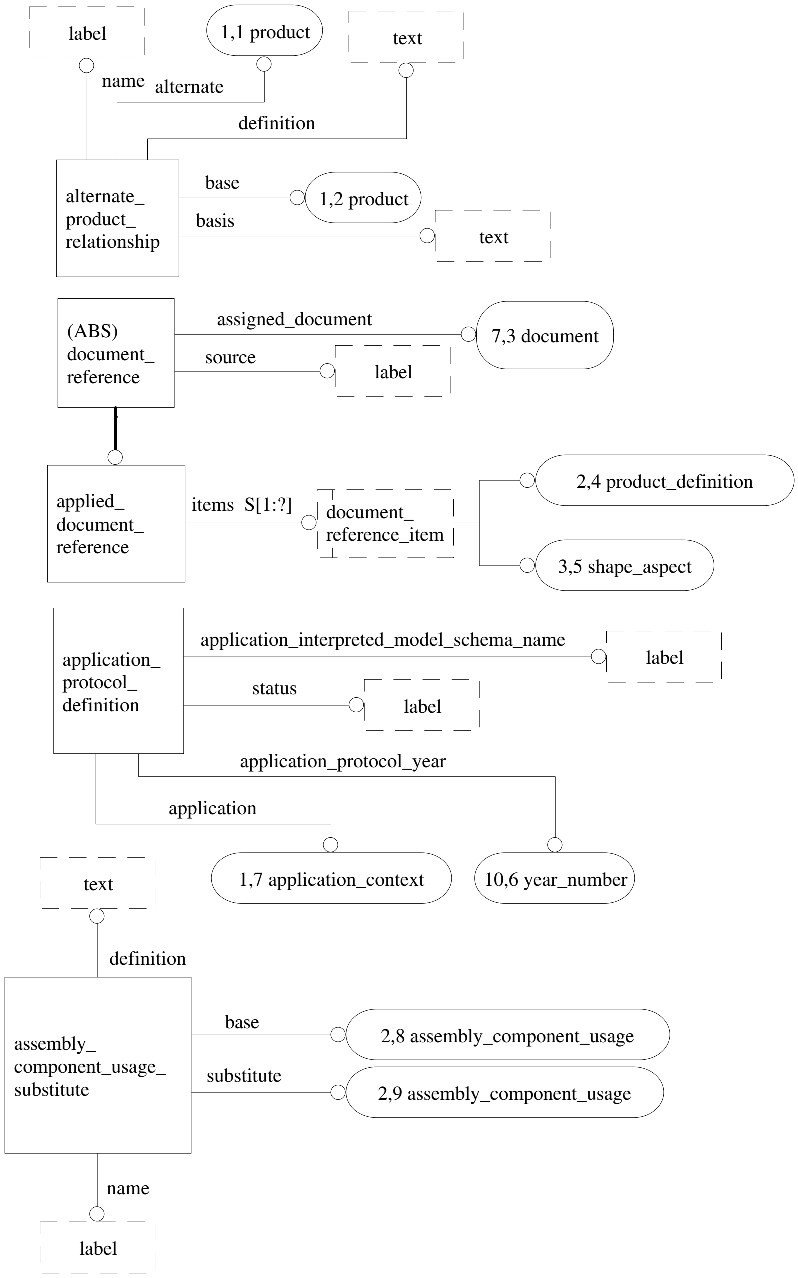

Figure H.64 - AIM EXPRESS-G diagram 64 of 94

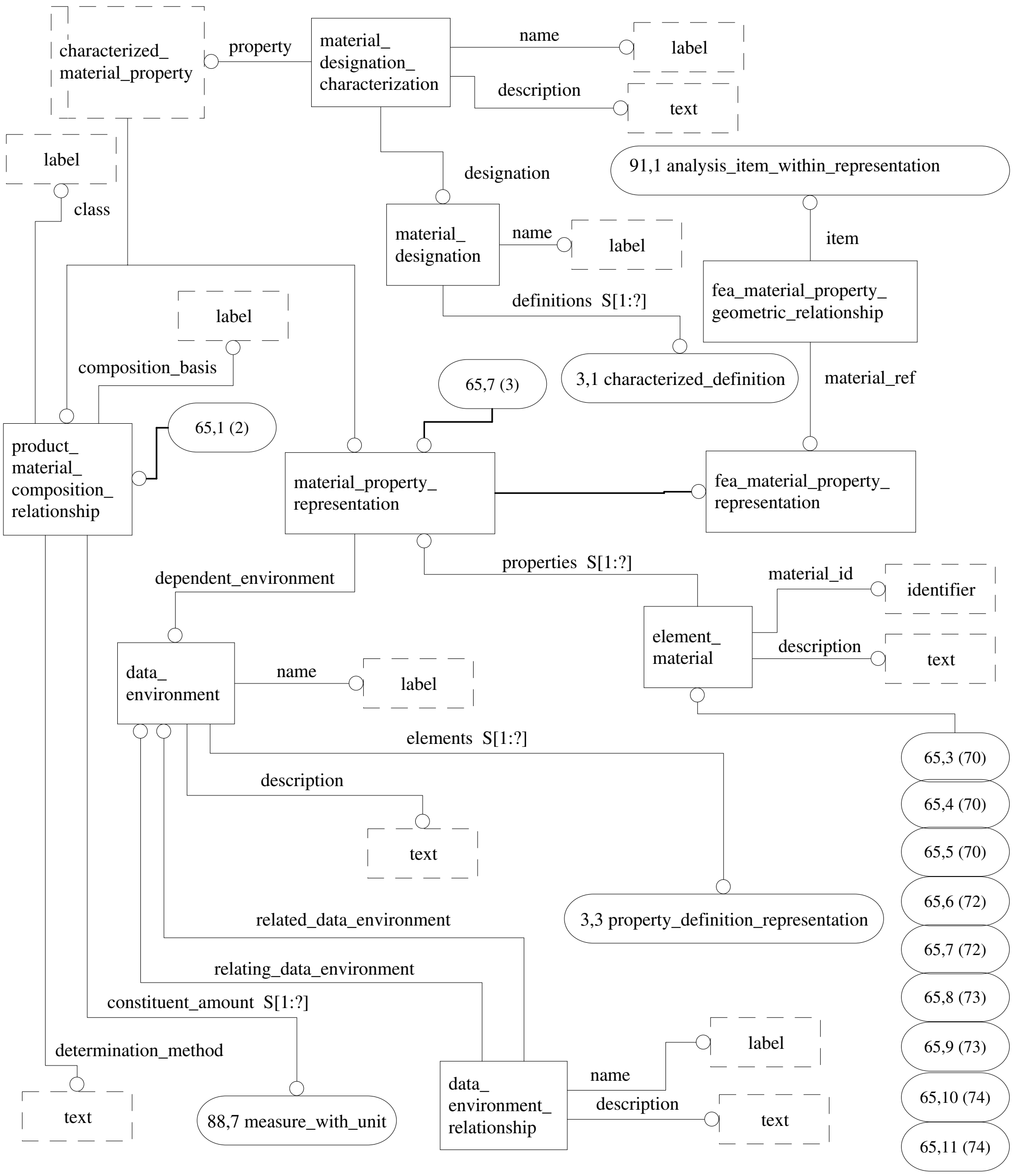

Figure H.65 - AIM EXPRESS-G diagram 65 of 94

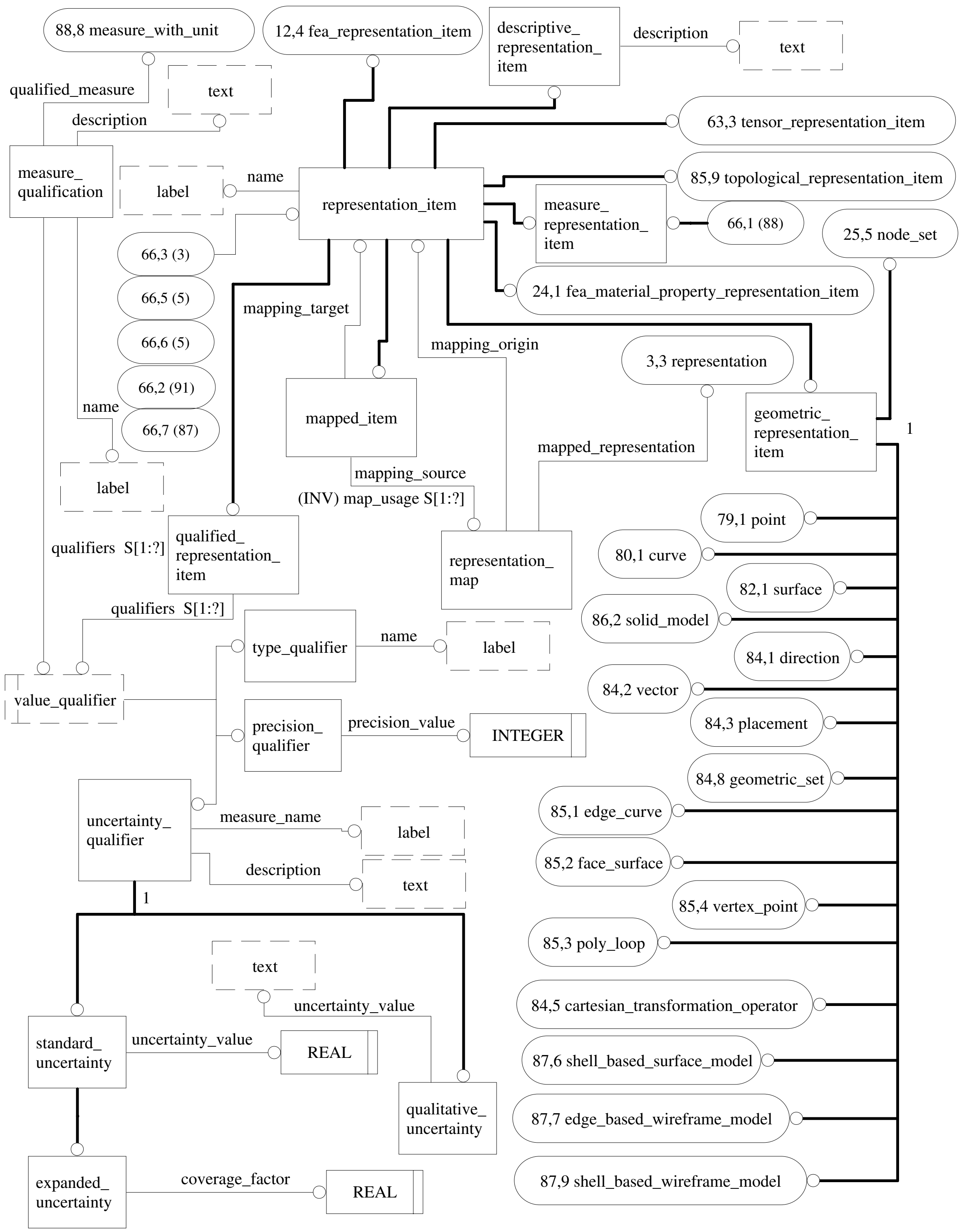

Figure H.66 - AIM EXPRESS-G diagram 66 of 94

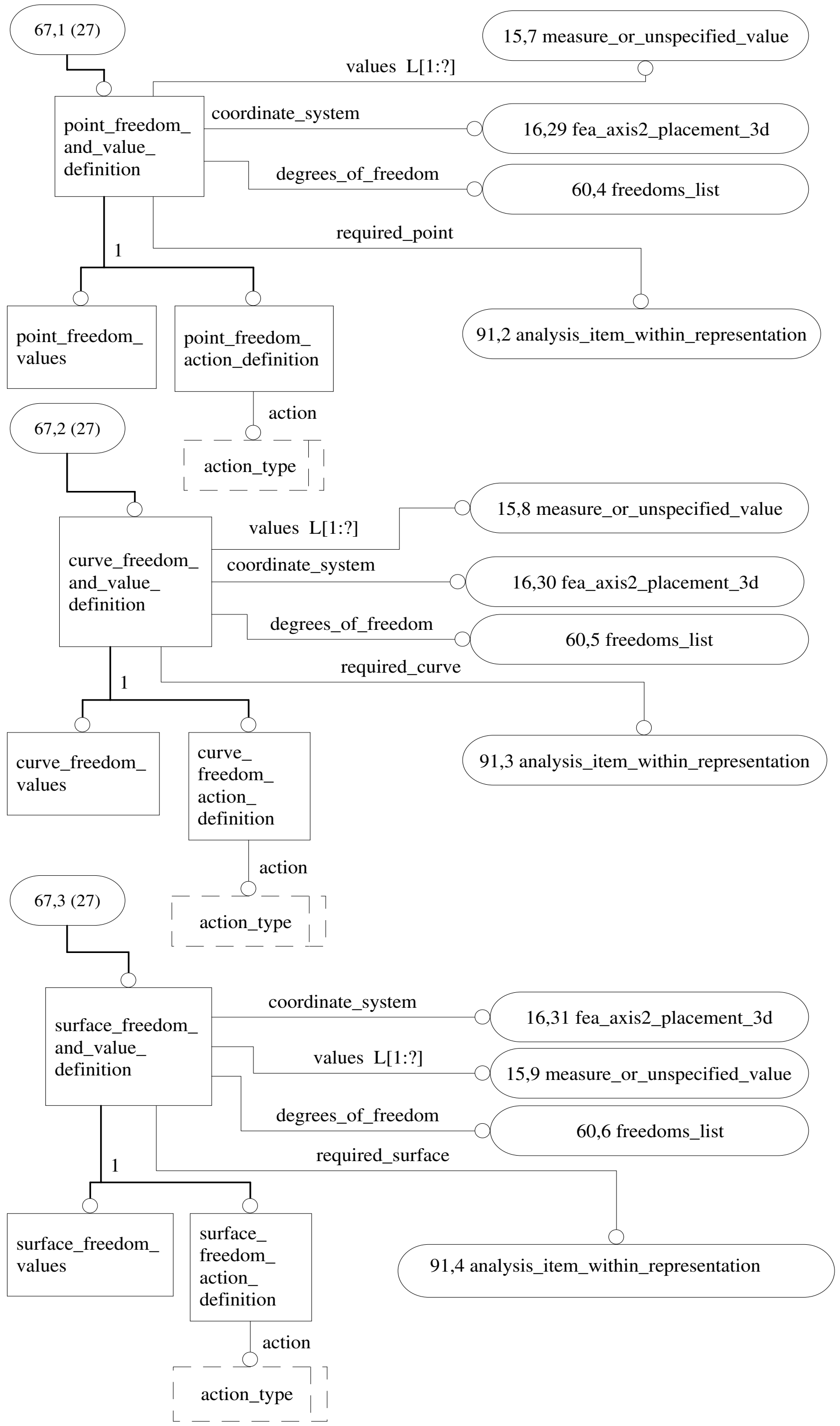

Figure H.67 - AIM EXPRESS-G diagram 67 of 94

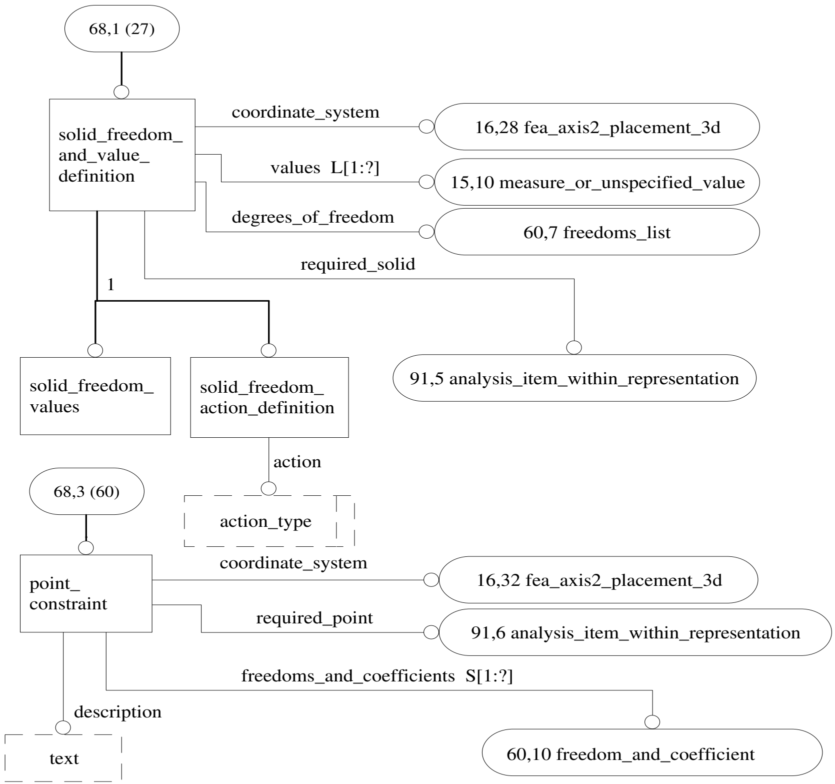

Figure H.68 - AIM EXPRESS-G diagram 68 of 94

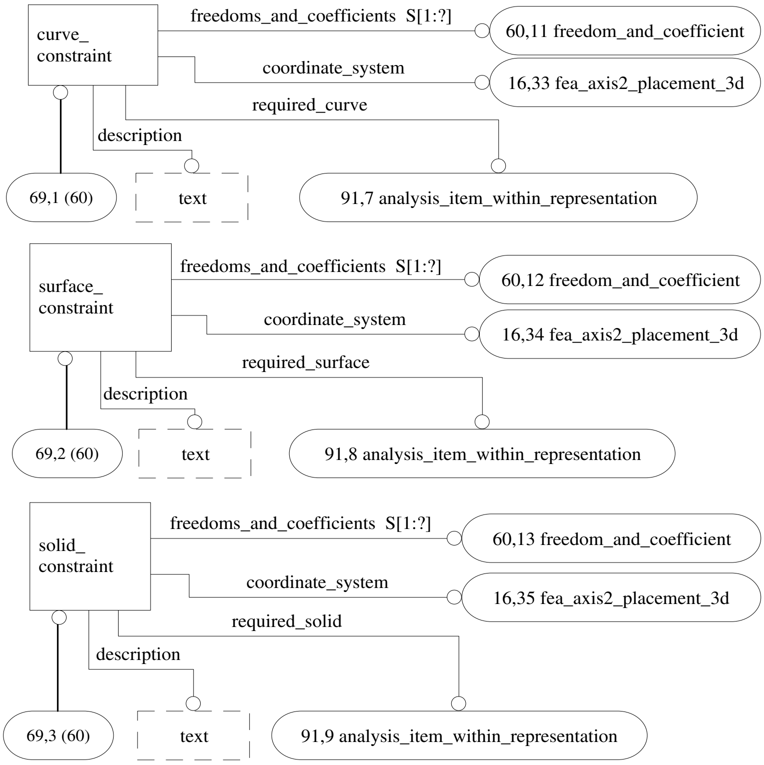

Figure H.69 - AIM EXPRESS-G diagram 69 of 94

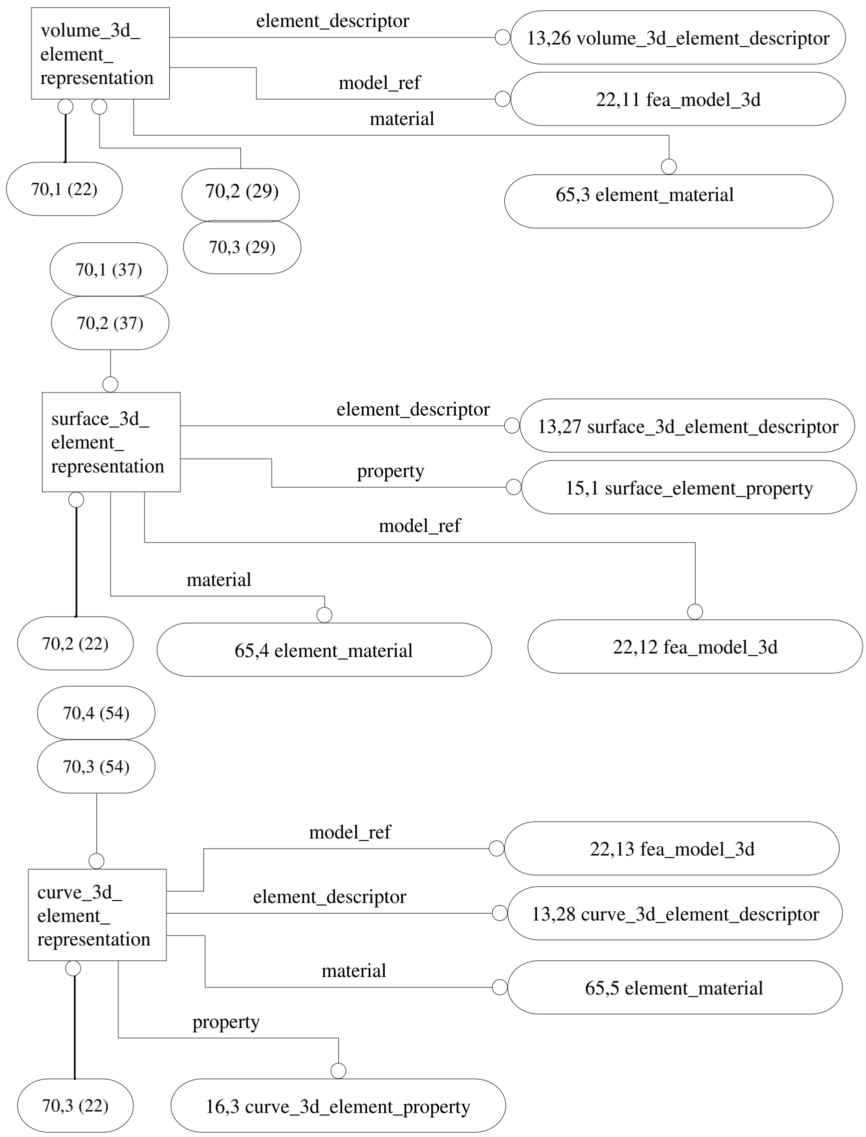

Figure H.70 - AIM EXPRESS-G diagram 70 of 94

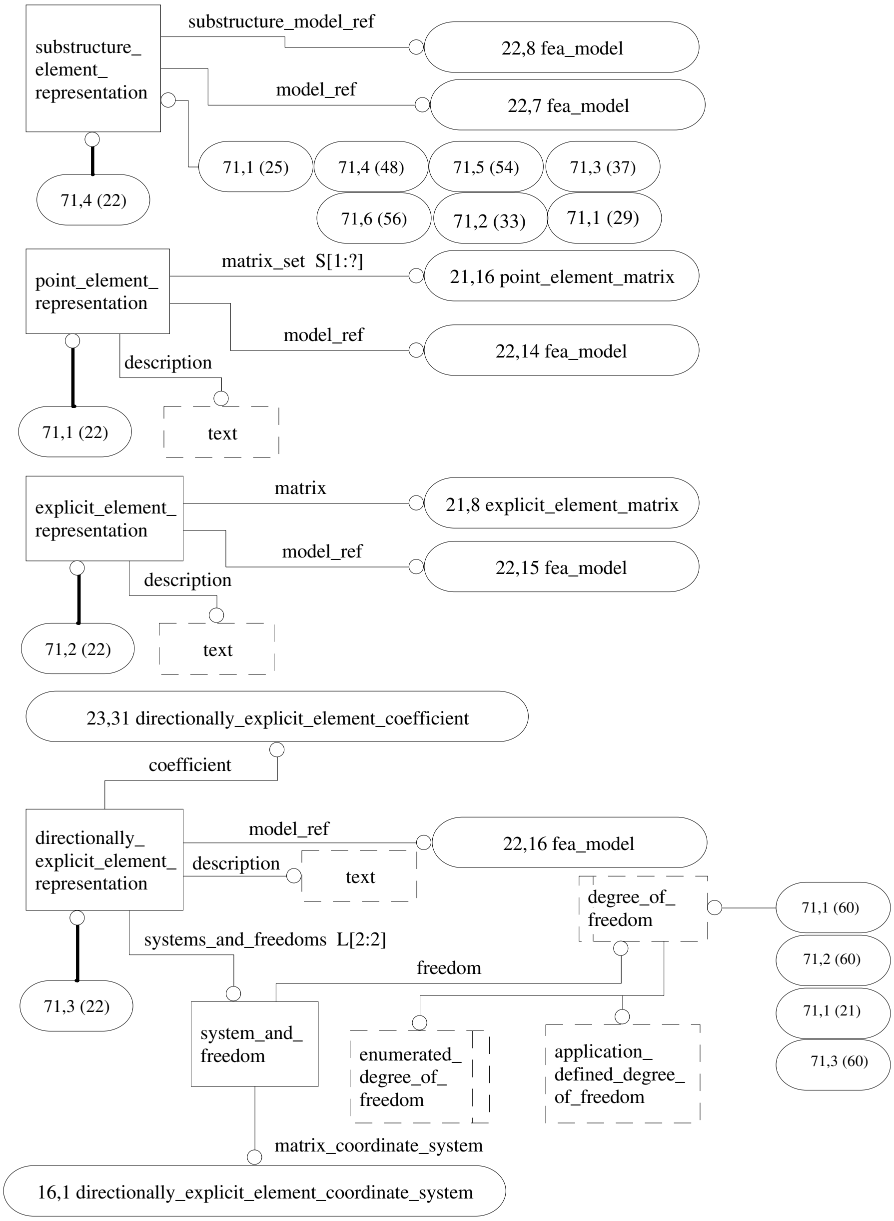

Figure H.71 - AIM EXPRESS-G diagram 71 of 94

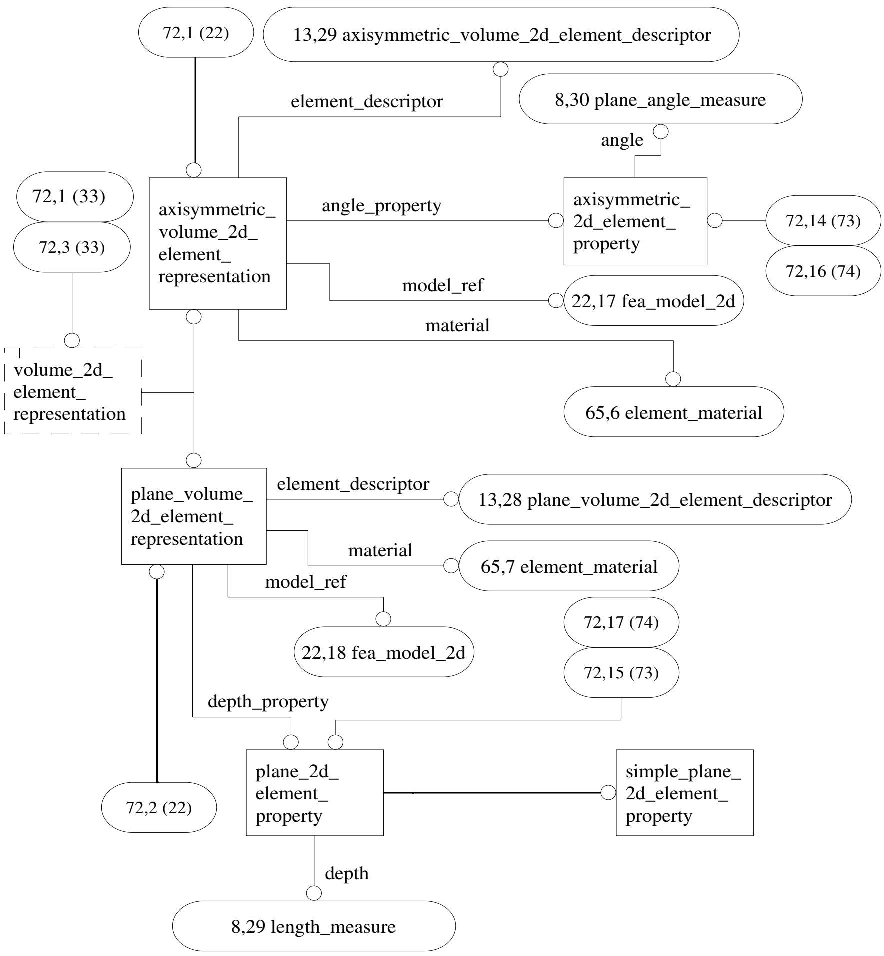

Figure H.72 - AIM EXPRESS-G diagram 72 of 94

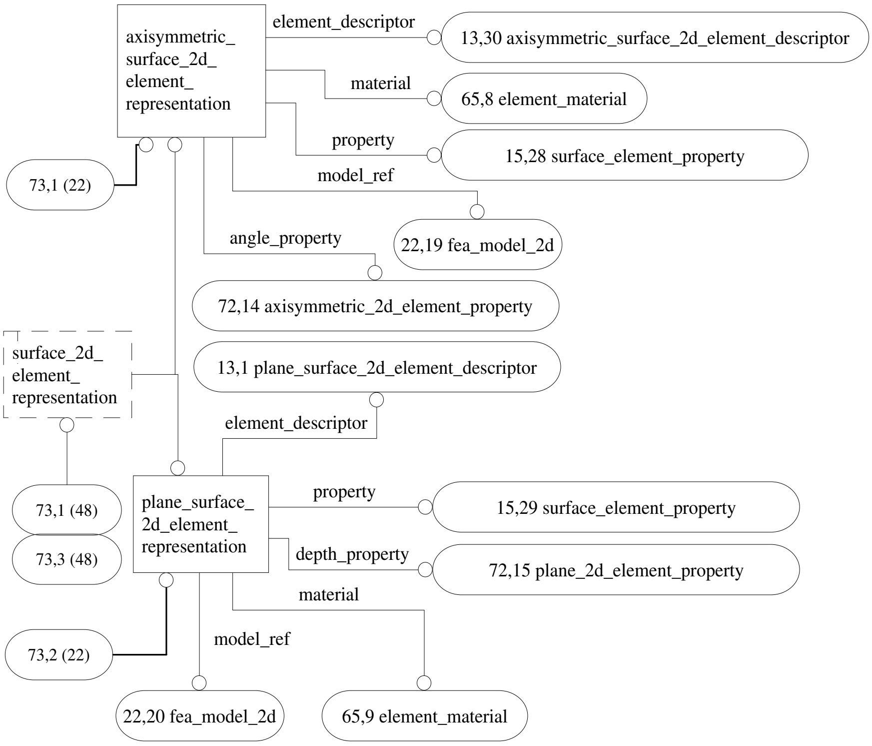

Figure H.73 - AIM EXPRESS-G diagram 73 of 94

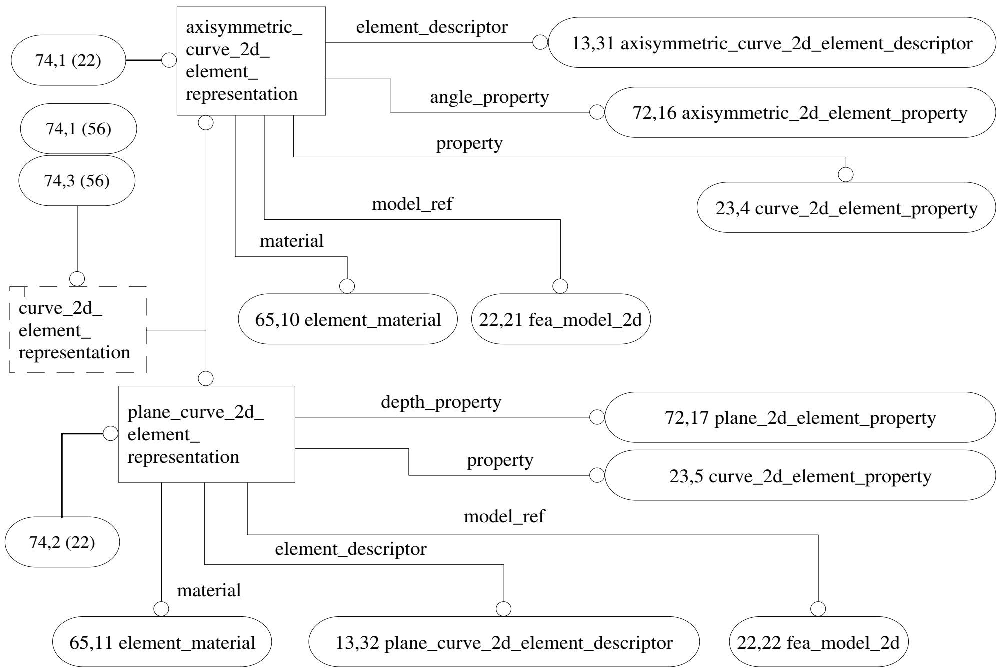

Figure H.74 - AIM EXPRESS-G diagram 74 of 91

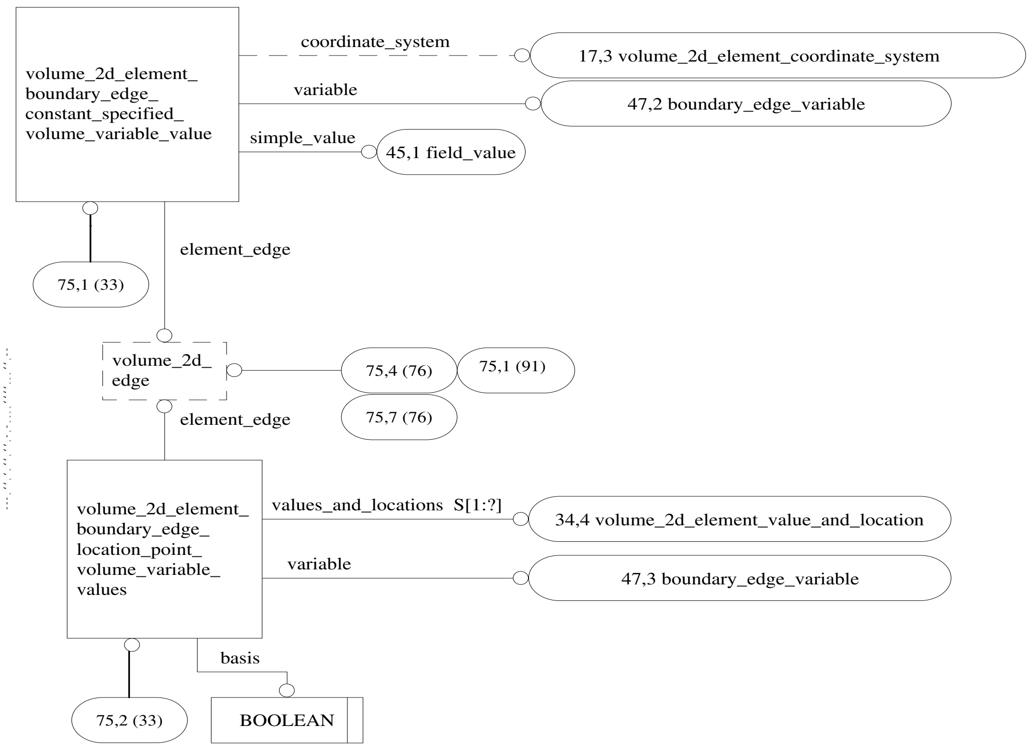

Figure H.75 - AIM EXPRESS-G diagram 75 of 94

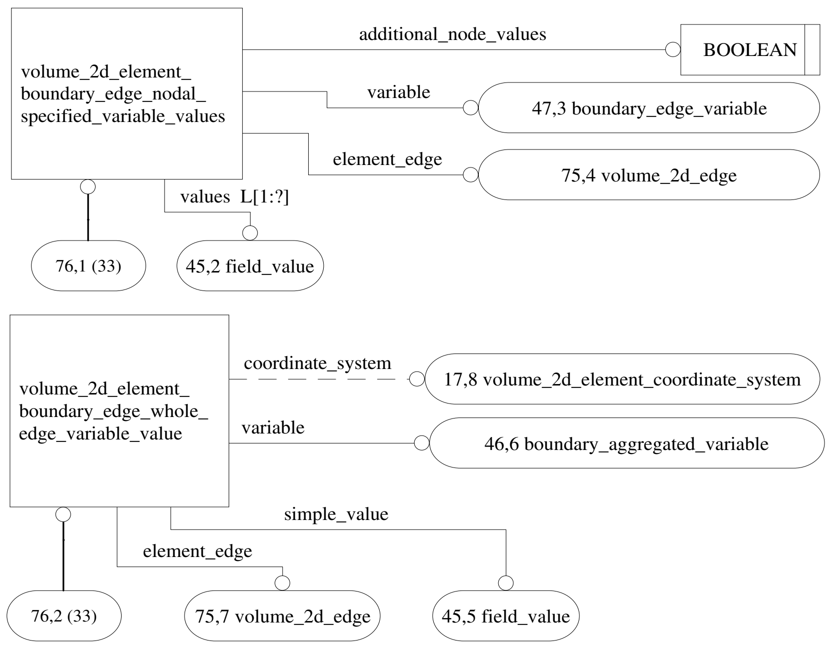

Figure H.76 - AIM EXPRESS-G diagram 76 of 94

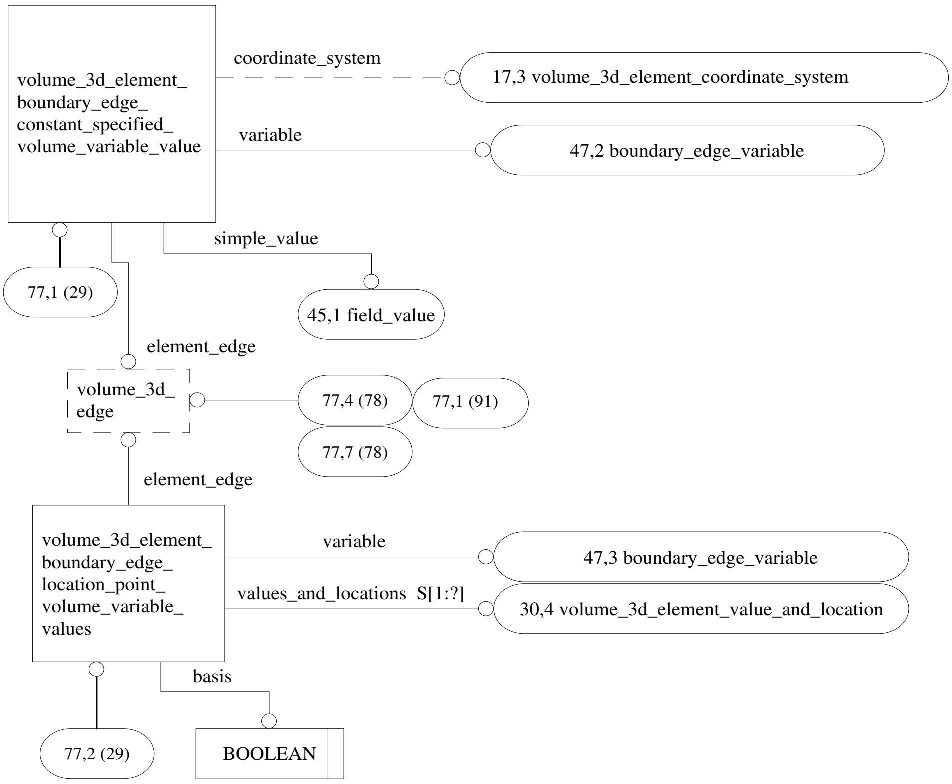

Figure H.77 - AIM EXPRESS-G diagram 77 of 94

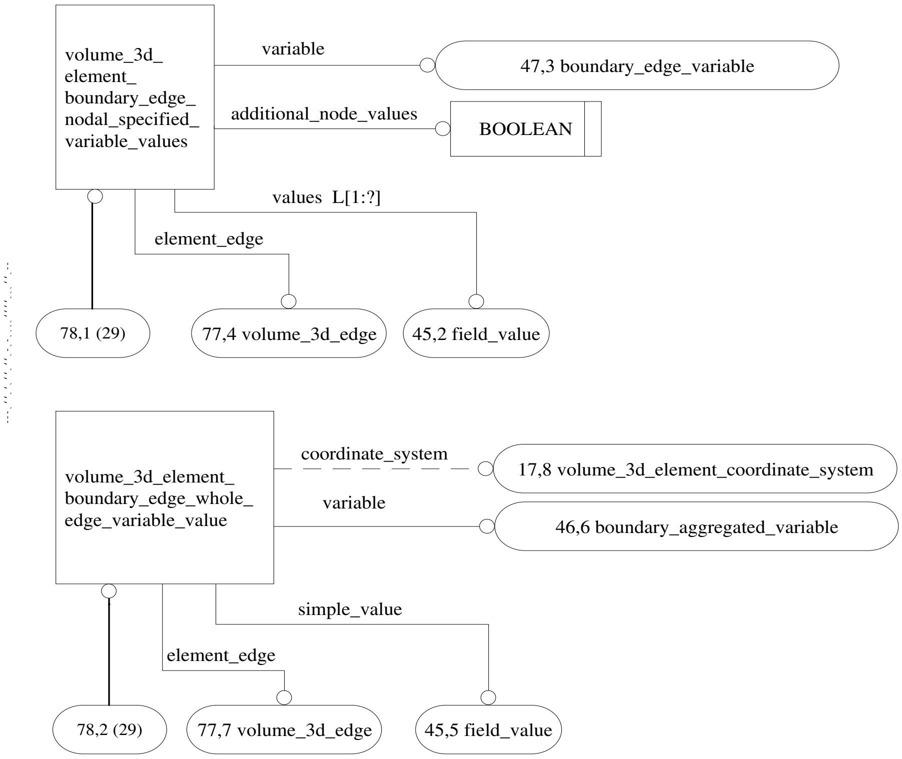

Figure H.78 - AIM EXPRESS-G diagram 78 of 94

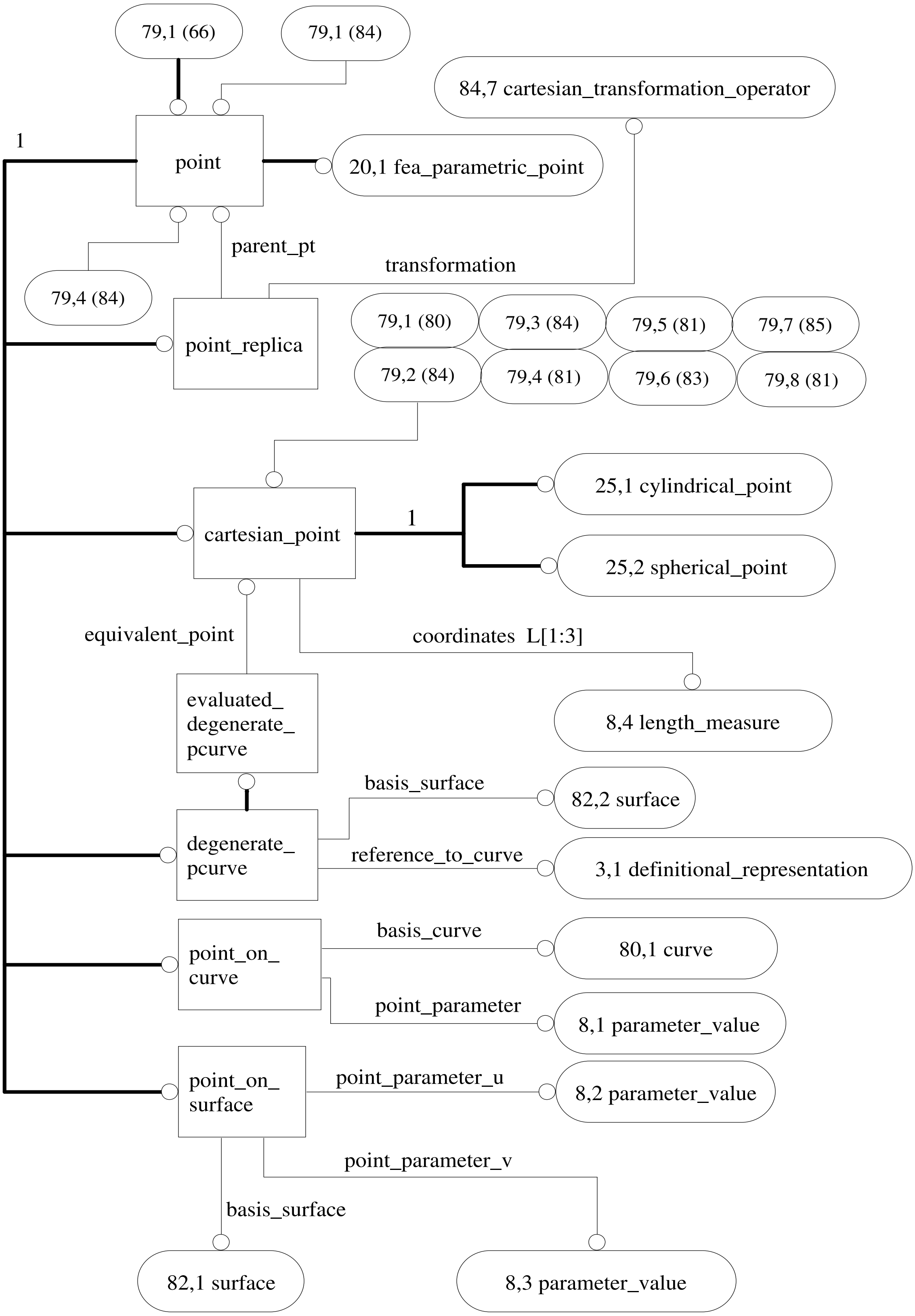

Figure H.79 - AIM EXPRESS-G diagram 79 or 94

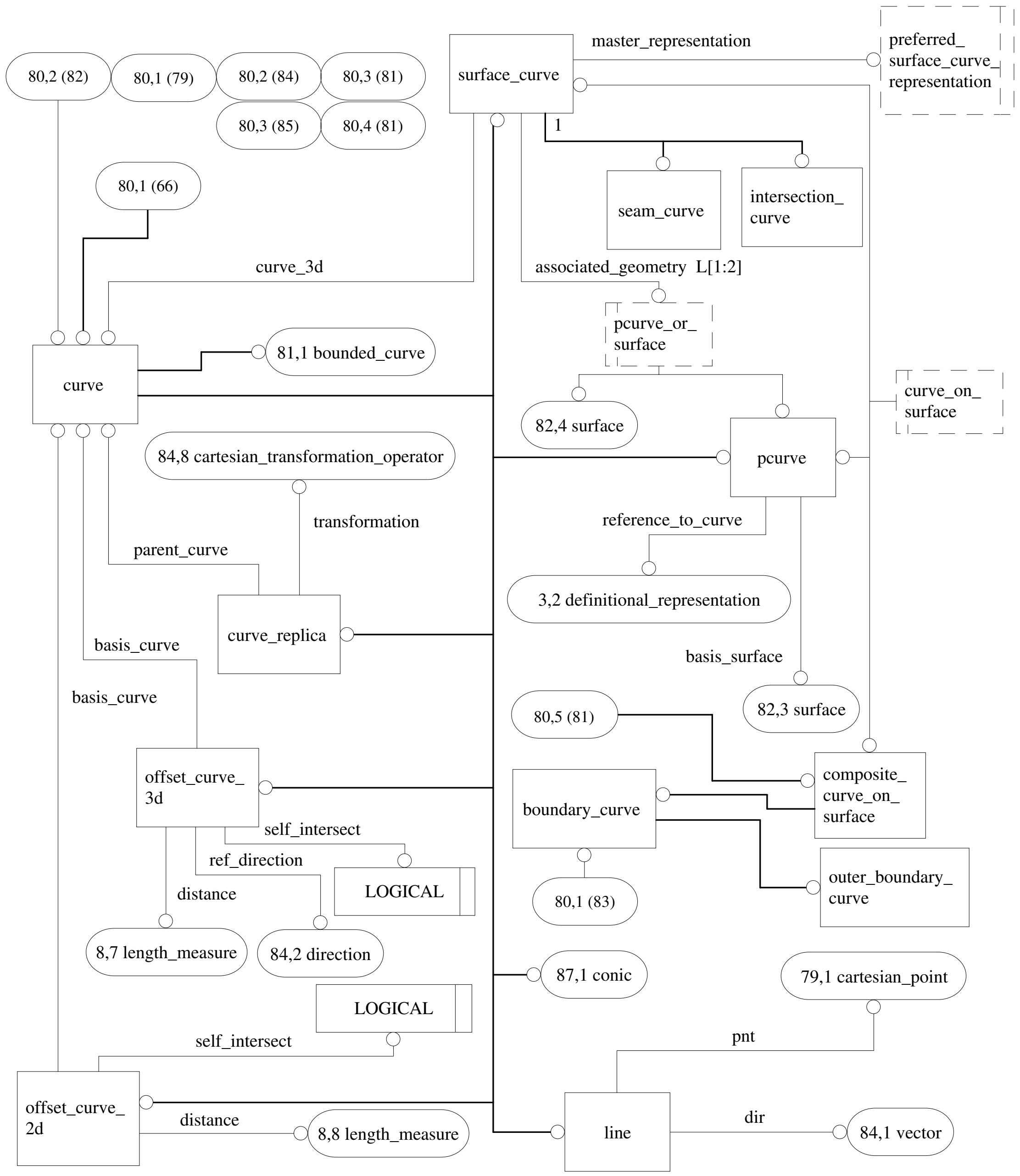

Figure H.80 -AIM EXPRESS-G diagram 80 of 94

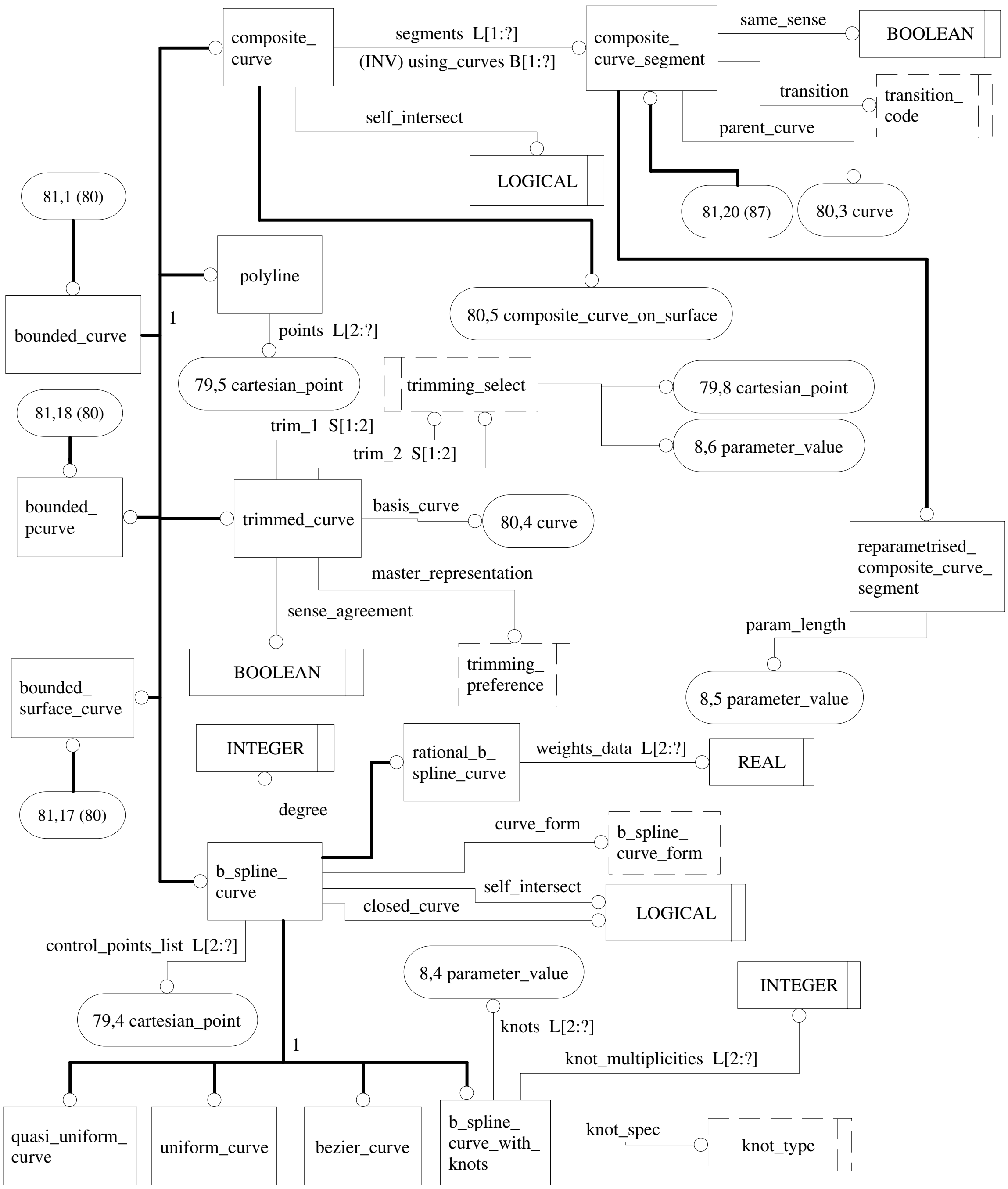

Figure H.81 - AIM EXPRESS-G diagram 81 of 94

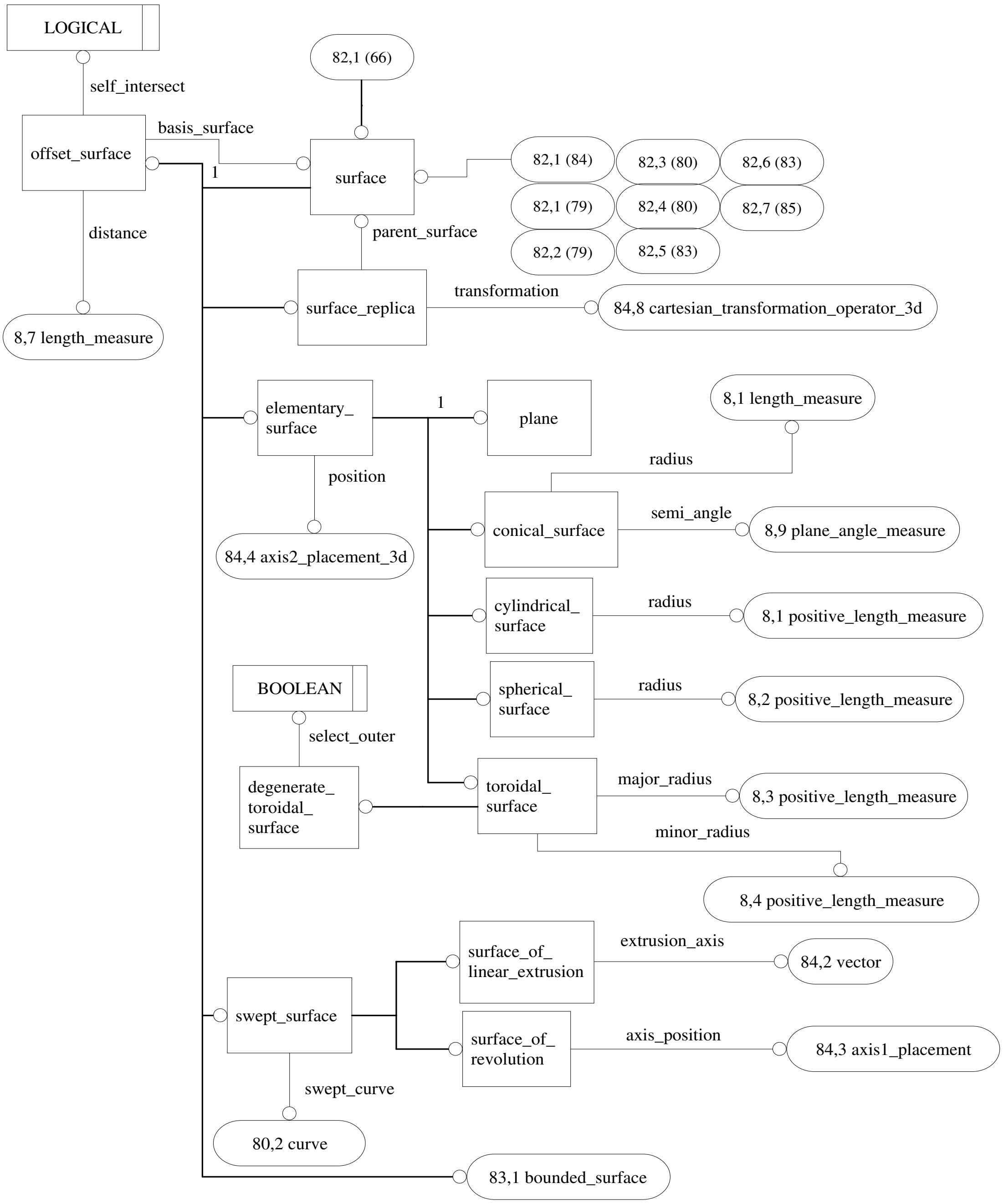

Figure H.82 - AIM EXPRESS-G diagram 82 of 94

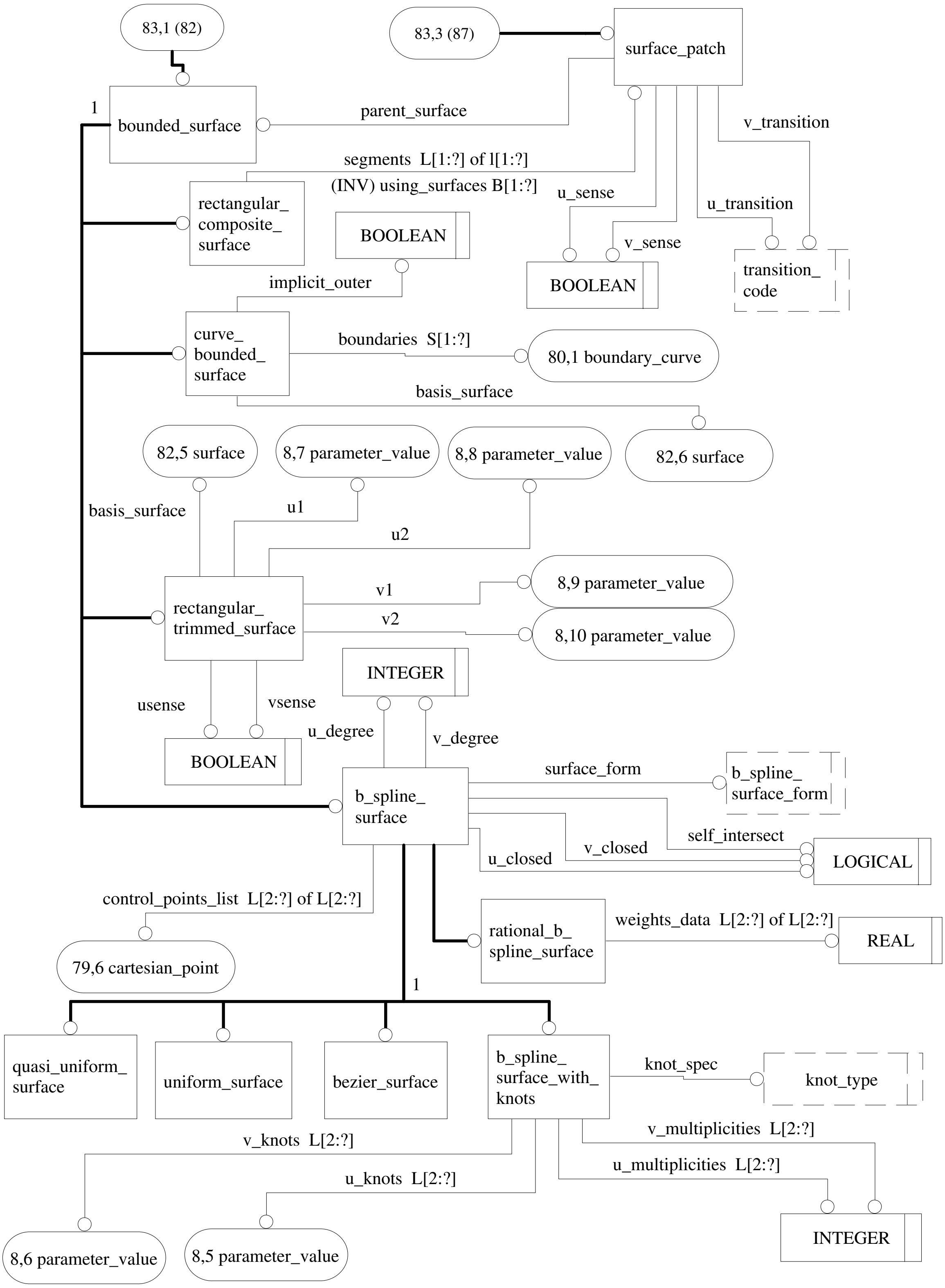

Figure H.83 - AIM EXPRESS-G diagram 83 of 94

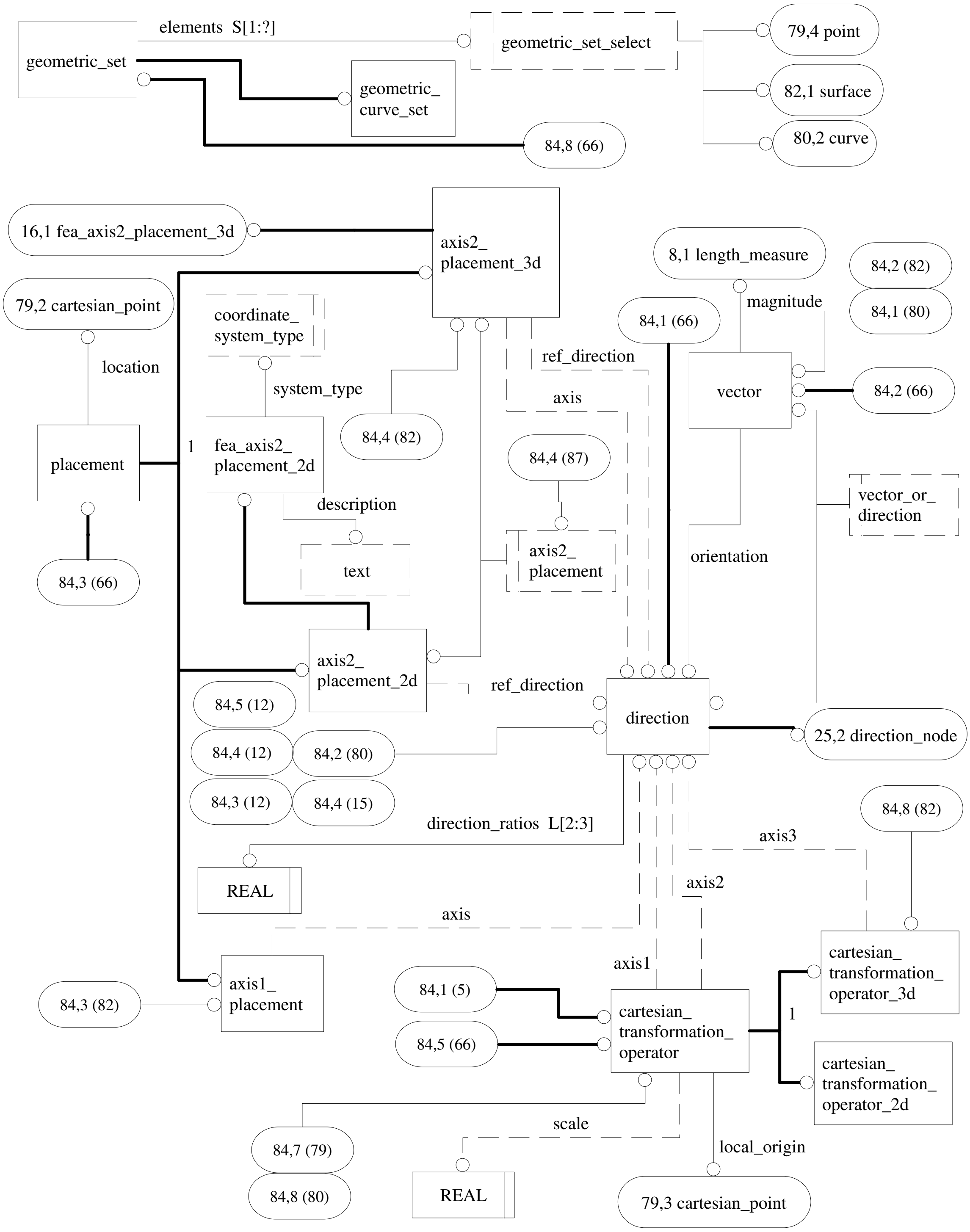

Figure H.84 - AIM EXPRESS-G diagram 84 of 94

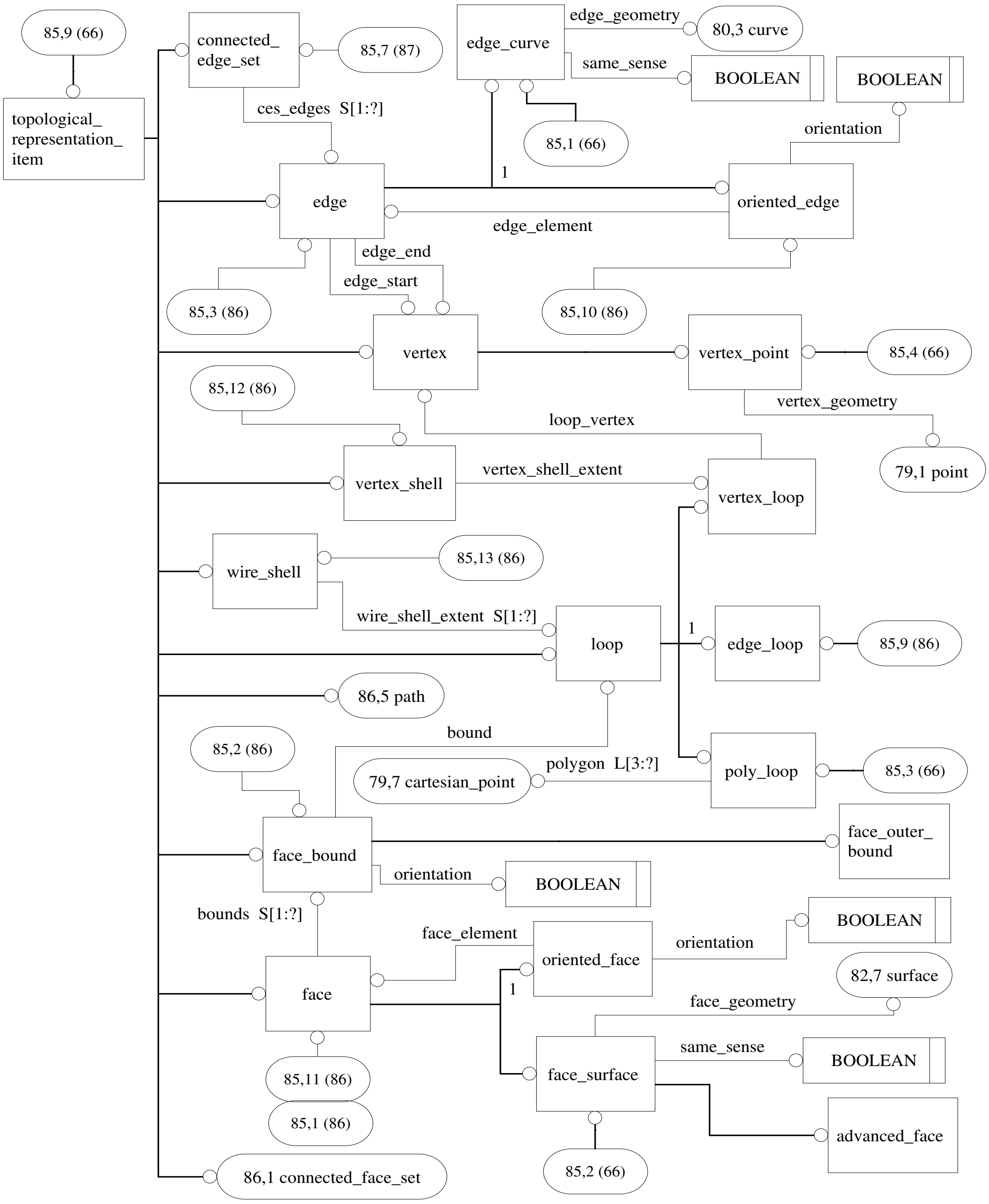

Figure H.85 - AIM EXPRESS-G diagram 85 of 94

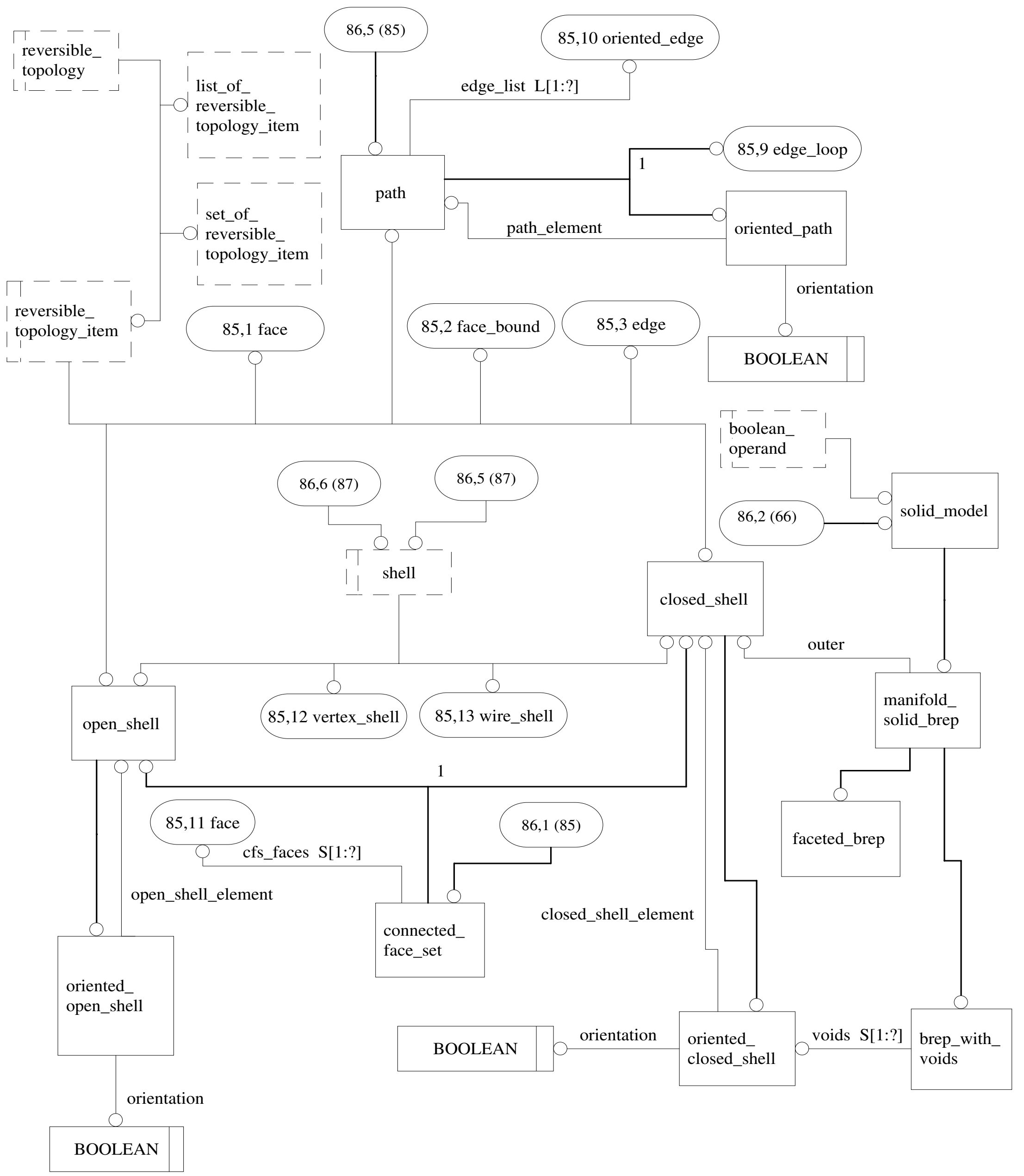

Figure H.86 - AIM EXPRESS-G diagram 86 of 94

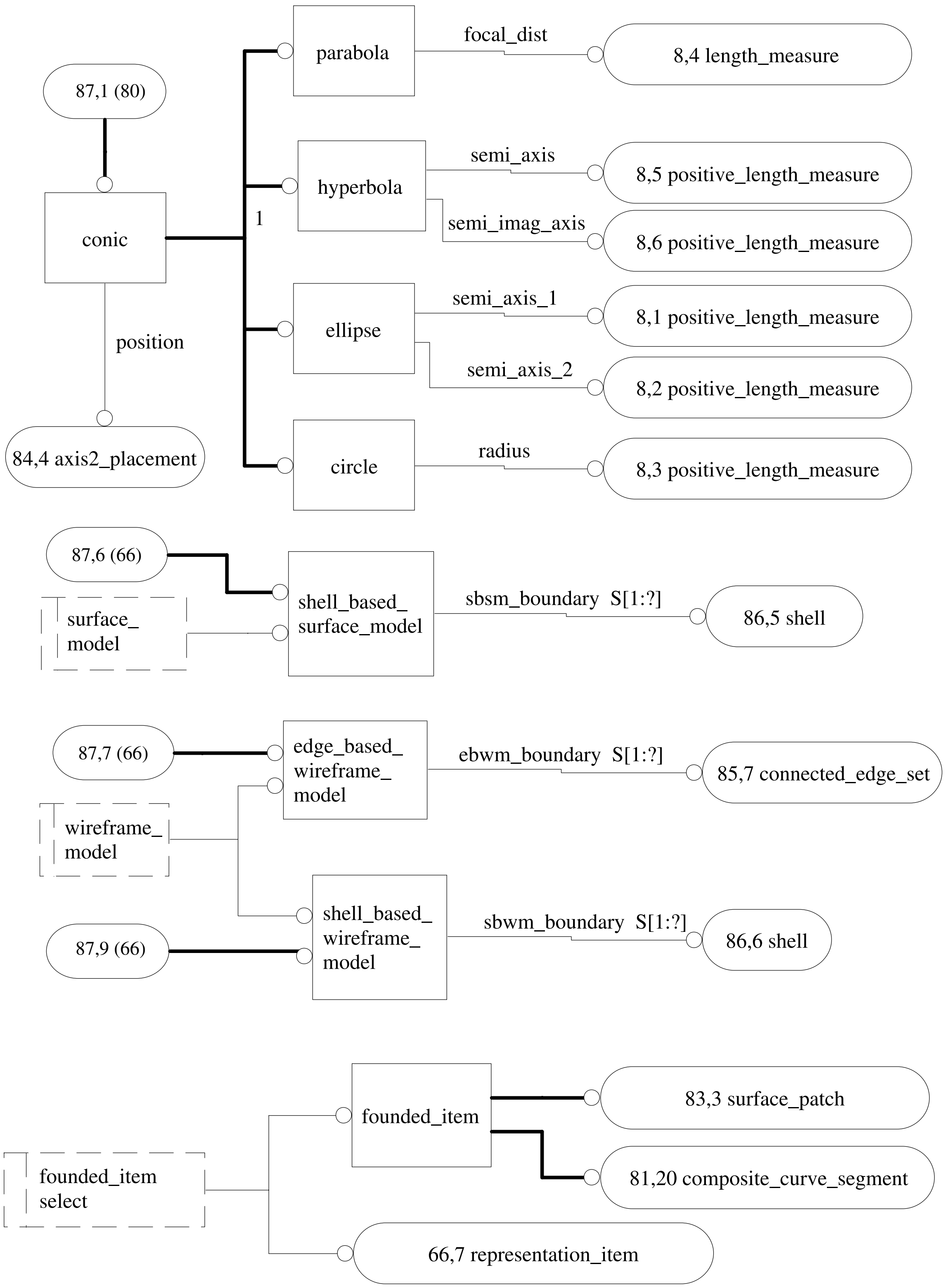

Figure H.87 - AIM EXPRESS-G diagram 87 of 94

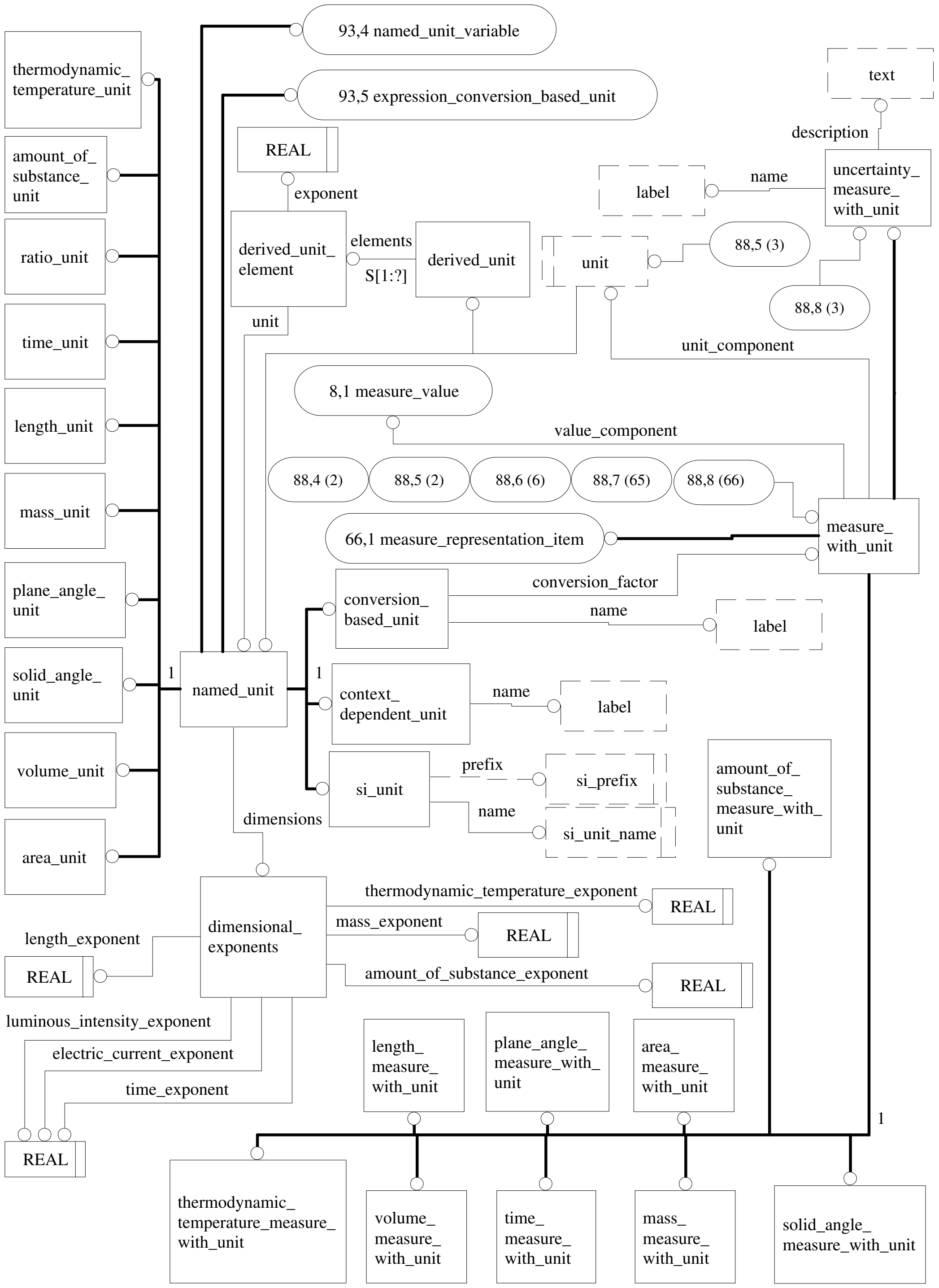

Figure H.88 - AIM EXPRESS-G diagram 88 of 94

Figure H.89 - AIM EXPRESS-G diagram 89 of 94

Figure H.90 - AIM EXPRESS-G diagram 90 of 94

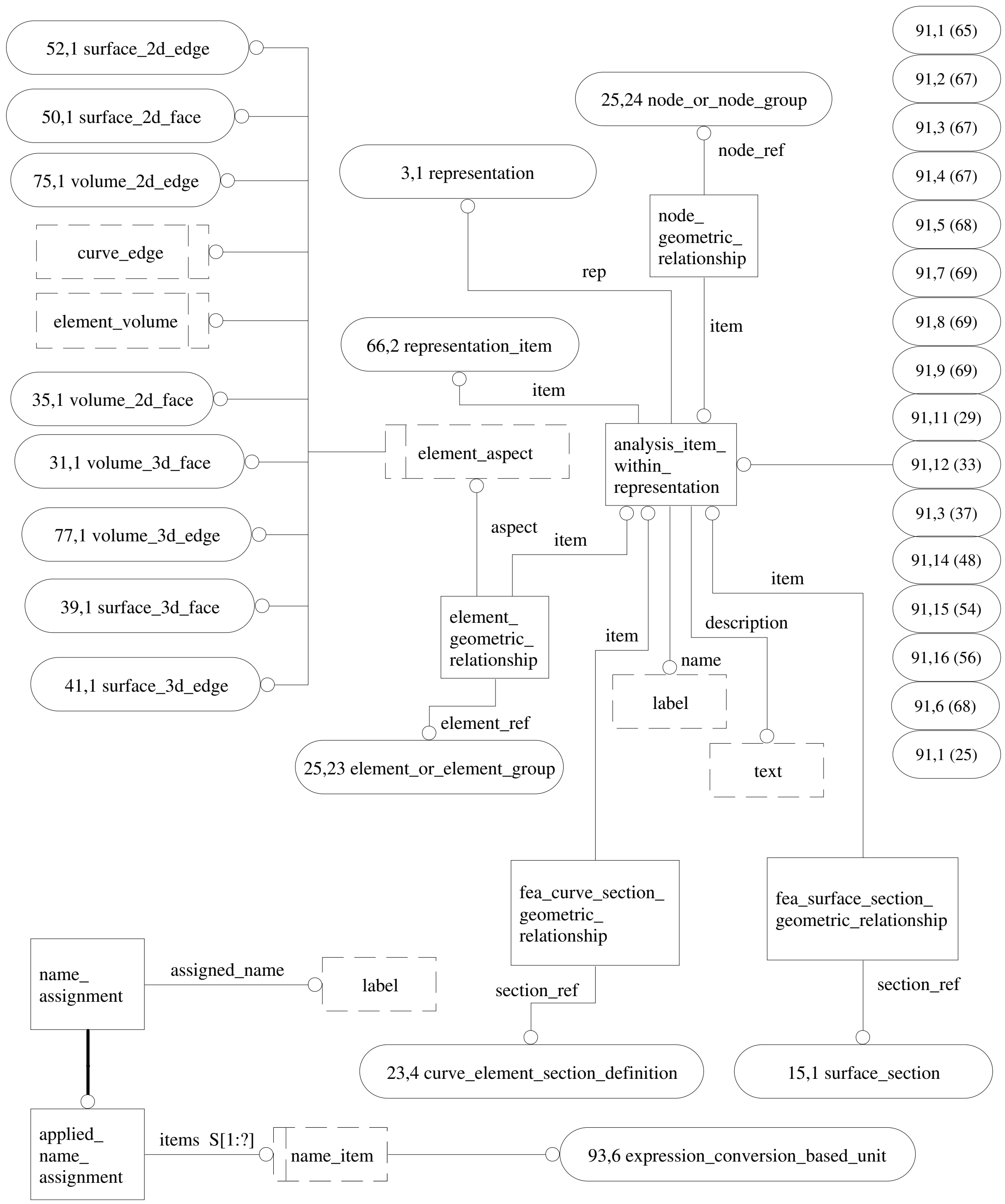

Figure H.91 - AIM EXPRESS-G diagram 91 of 94

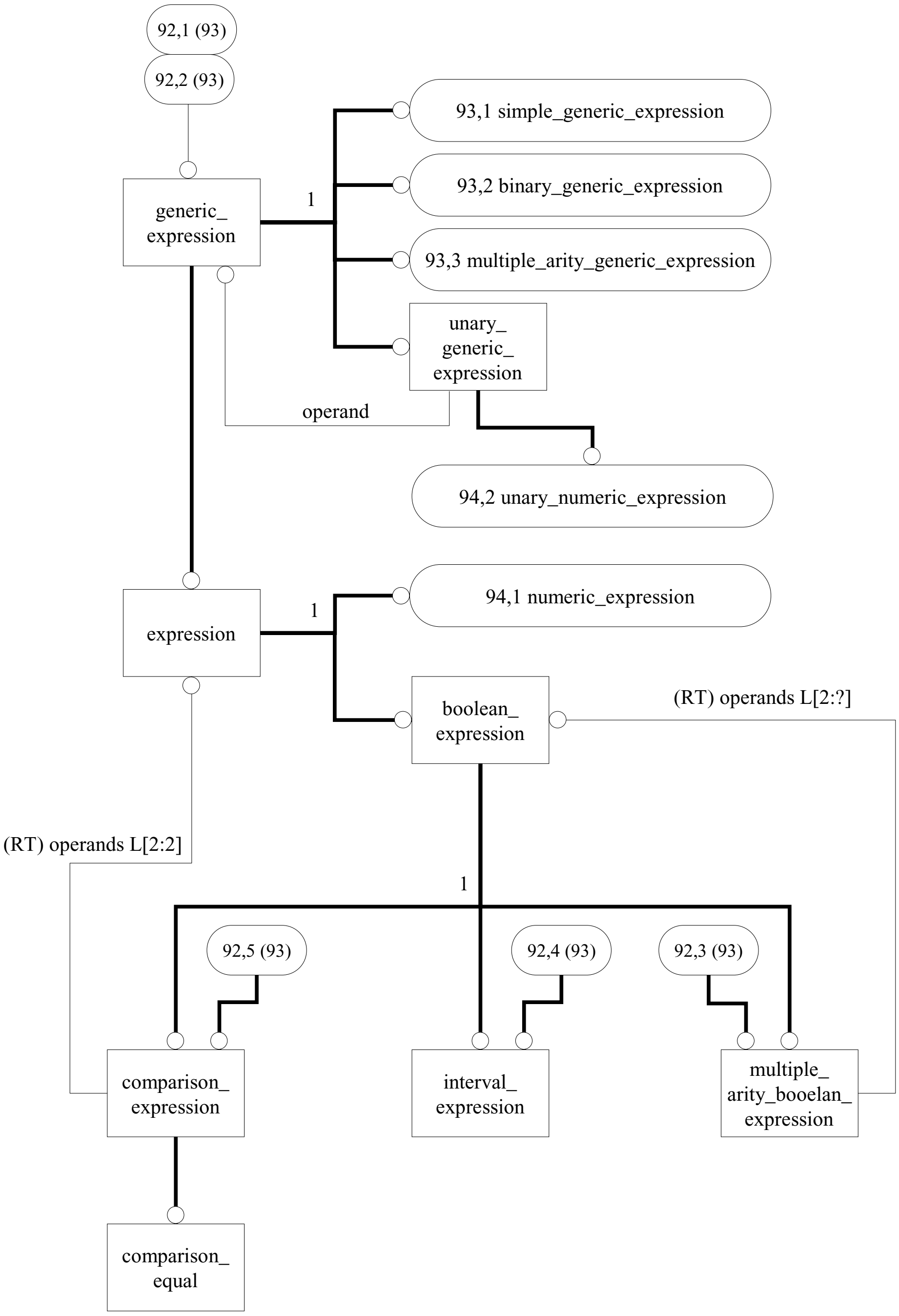

Figure H.92 - AIM EXPRESS-G diagram 92 of 94

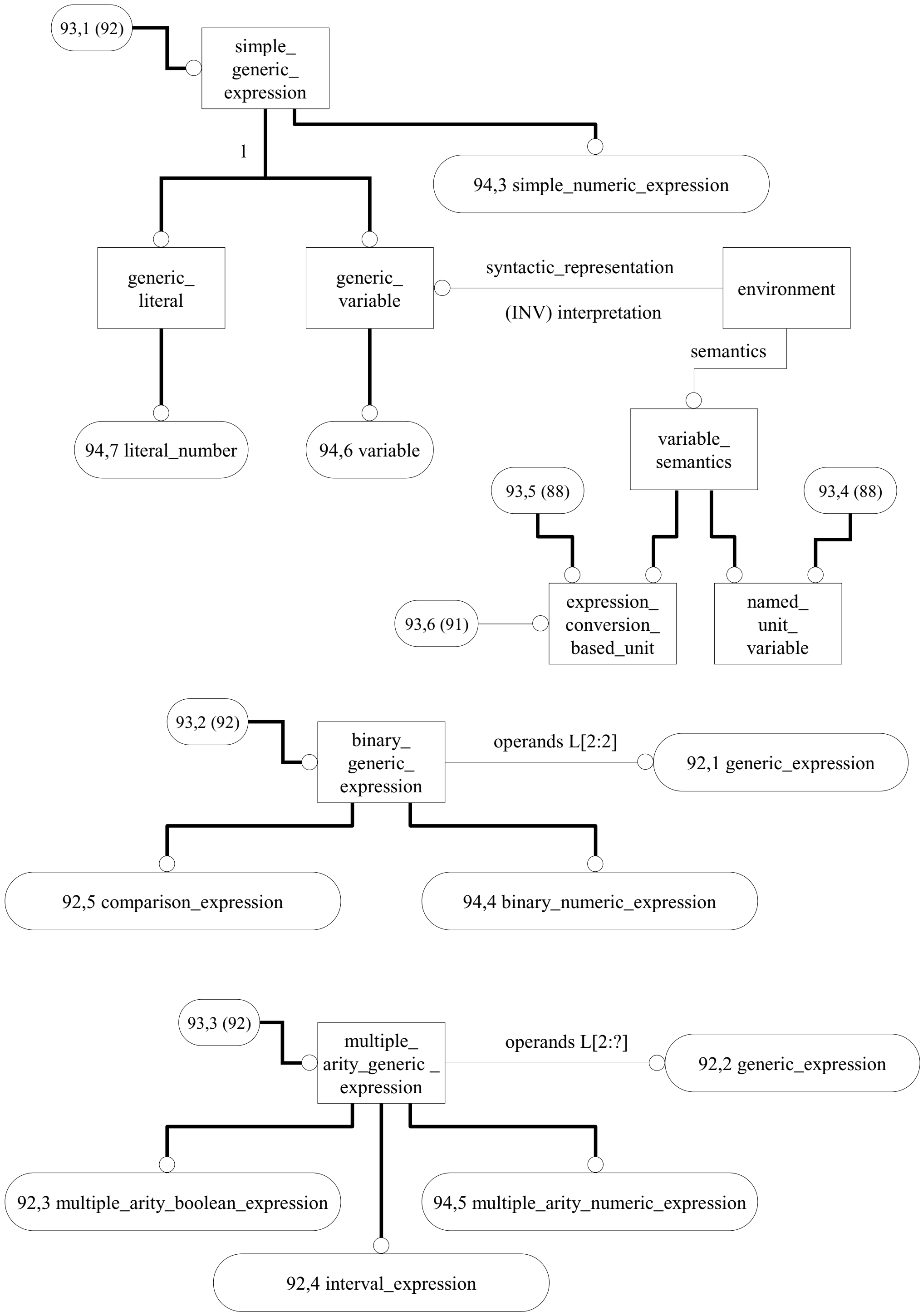

Figure H.93 - AIM EXPRESS-G diagram 93 of 94

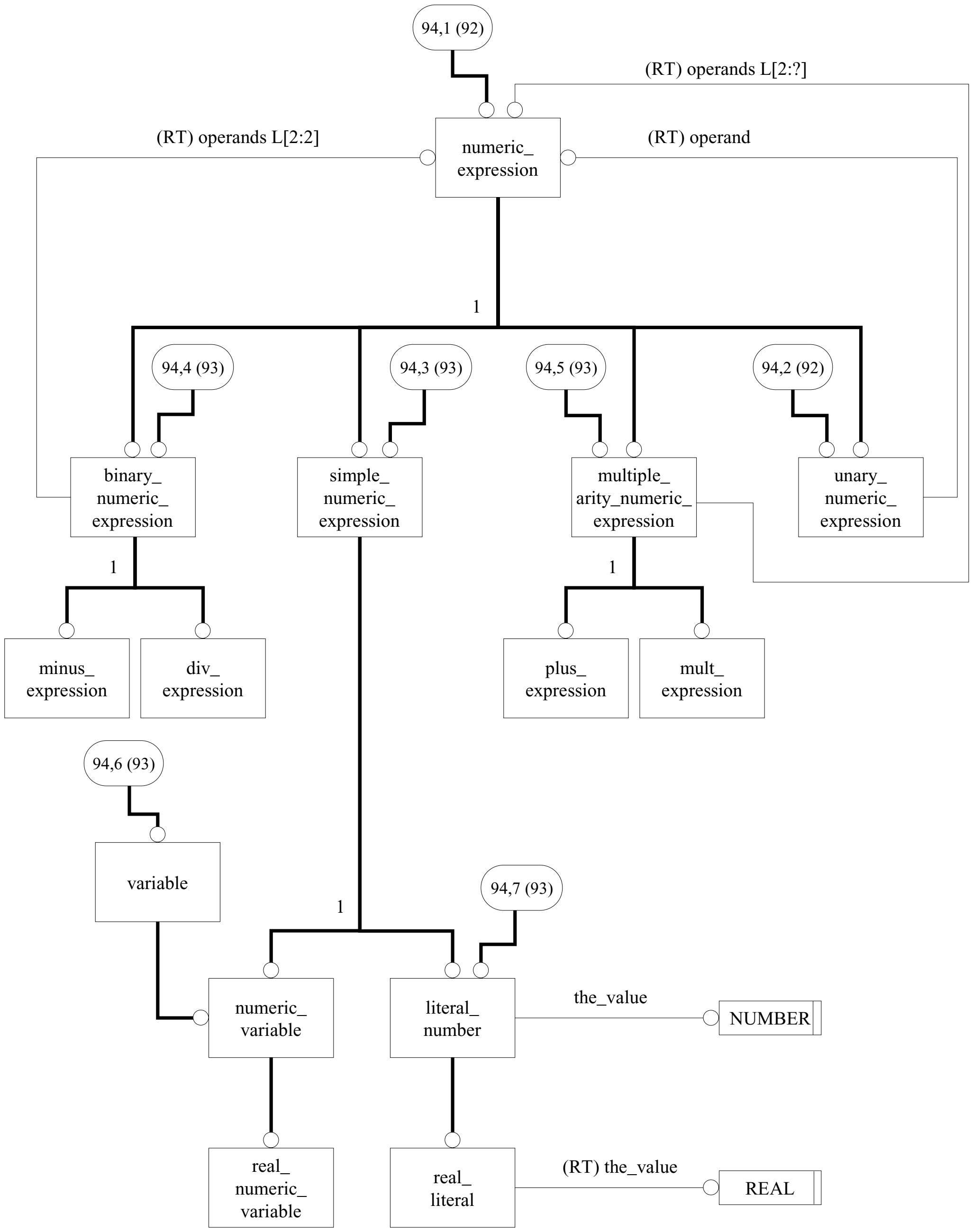

Figure H.94 - AIM EXPRESS-G diagram 94 of 94

Annex J
(informative)

Computer-interpretable listings

This annex provides a listing of the EXPRESS entity names and corresponding short najmes as specified in this part of ISO 10303. It also provides a listing of the complete EXPRESS schema specified in this part of ISO 10303 without comments or other explanatory text. This annex is available in computer-interpretable form and can be found at the following URLs:

Short names: http://www.mel.nist.gov/div826/subject/spde/snr/
EXPRESS: http://www.mel.nist.gov/step/parts/part209/is/

If there is difficulty accessing these sites, contact ISO Central Secretariat or contact the ISO TC 184/SC4 Secretariat directly at: sc4sec@cme.nist.gov.

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)

ARM EXPRESS listing

This annex provides the EXPRESS representation of the ARM presented graphically in annex G. In the event of any discrepancy between the EXPRESS and graphical representation, the EXPRESS representation shall be used.

```
*)
SCHEMA AP209_ARM;

TYPE angle_select = SELECT
          (DIRECTION,
           CURVE,
           point_path);
END_TYPE;

TYPE boundary_class_zone = SELECT
          (boundary_curve_representation,
           LOOP);
END_TYPE;

TYPE component_class_for_assembly = SELECT
          (ply,
           ply_laminate,
           filament_laminate,
           processed_core,
           composite_assembly);
END_TYPE;

TYPE composite_assembly_shape = SELECT
          (advanced_boundary_representation,
           three_d_geometry_set,
           faceted_boundary_representation,
           manifold_surface_with_topology,
           non_topological_surface_and_wireframe,
           wireframe_with_topology);
END_TYPE;

TYPE constraint_definition_reference = SELECT
          (geometry_element,
           group,
           node);
END_TYPE;

TYPE core_shape = SELECT
          (advanced_boundary_representation,
           beveled_sheet_representation,
           faceted_boundary_representation);
END_TYPE;

TYPE cross_section_shape_select = SELECT
          (idealized_analysis_shape,
           nominal_design_shape);
END_TYPE;
```

```
TYPE definition_element = SELECT
          (curve_element,
           shape_aspect,
           surface_element,
           volume_element);
END_TYPE;

TYPE definition_reference = SELECT
          (element,
           geometry_element,
           group,
           node);
END_TYPE;

TYPE design_or_analysis = SELECT
          (analysis_version,
           analysis_discipline_product_definition,
           assembly,
           design_discipline_product_definition,
           part_version);
END_TYPE;

TYPE element_property_select = SELECT
          (curve_property,
           surface_property);
END_TYPE;

TYPE material_select = SELECT
          (anisotropic_material,
           isotropic_material,
           laminate_table);
END_TYPE;

TYPE ply_laminate_shape = SELECT
           (composite_sheet_representation,
            three_d_geometry_set);
END_TYPE;

TYPE ply_piece_shape = SELECT
           (ply_shape);
END_TYPE;

TYPE ply_stock_material = SELECT
           (discontinuous_fiber_assembly,
            filament_assembly,
            isotropic_material);
END_TYPE;

TYPE projection_method = ENUMERATION OF
           (projection_reference,
            surface_normal);
END_TYPE;

TYPE retention_data_select = SELECT
          (additional_design_information,
           analysis,
           analysis_discipline_product_definition,
           analysis_report_representation,
           analysis_version,
           assembly,
```



```
            design_discipline_product_definition,
            design_material,
            fe_analysis,
            fe_analysis_results,
            fea_model,
            material_property,
            material_specification,
            part,
            part_version);
END_TYPE;

TYPE thickness_laminate_table_component = SELECT
           (filament_laminate,
            ply,
            processed_core);
END_TYPE;

ENTITY additional_design_information;
 additional_information  : SET [1:?] OF specification;
 design                  : design_discipline_product_definition;
END_ENTITY;

ENTITY advanced_boundary_representation
  SUBTYPE OF (geometric_model_representation);
END_ENTITY;

ENTITY alternate_part;
 alternate  : part;
END_ENTITY;

ENTITY analysis;
 analysis_type   : TEXT;
 owner           : person_organization;
END_ENTITY;

ENTITY analysis_design_version_relationship;
 analysis  : analysis_version;
 design    : part_version;
END_ENTITY;

ENTITY analysis_discipline_product_definition;
 approved_by                 : OPTIONAL approval;
 creation_date               : DATE;
 creator                     : person_organization;
 description                 : TEXT;
 discipline_identification   : IDENTIFIER;
 cae_filename                : OPTIONAL FILE;
 version                     : analysis_version;
END_ENTITY;

ENTITY analysis_message
  SUBTYPE OF (fe_analysis_state_definition);
 message_level  : LABEL;
 quality        : TEXT;
END_ENTITY;

ENTITY analysis_report_representation
  SUPERTYPE OF (graphical_representation ANDOR
                tabular_representation);
 control    : OPTIONAL fe_analysis;
```

```
 model_ref  : fea_model;
 result     : OPTIONAL fe_analysis_results;
END_ENTITY;

ENTITY analysis_shape
  SUPERTYPE OF (idealized_analysis_shape ANDOR
               node_shape);
 analysis_view  : analysis_discipline_product_definition;
END_ENTITY;

ENTITY analysis_version;
 analysis_number  : analysis;
 approved_by      : OPTIONAL approval;
 contract_number  : OPTIONAL IDENTIFIER;
 creator          : person_organization;
 release_status   : LABEL;
 revision_letter  : IDENTIFIER;
 security_code    : LABEL;
END_ENTITY;

ENTITY ANGLE_MEASURE;
END_ENTITY;

ENTITY anisotropic_material
  SUBTYPE OF (stock_material);
END_ENTITY;

ENTITY approval;
 authorized_by   : SET [1:?] OF person_organization;
 effective_date  : DATE;
 purpose         : TEXT;
 status          : LABEL;
END_ENTITY;

ENTITY assembly
  SUPERTYPE OF (ONEOF (next_higher_assembly,
                       promissory_usage));
 assembly_part   : design_discipline_product_definition;
 component_part  : design_discipline_product_definition;
 security_code   : LABEL;
END_ENTITY;

ENTITY beveled_sheet_representation
  SUBTYPE OF (constituent_shape_representation);
 bevel_angle              : ANGLE_MEASURE;
 boundary_surface         : composite_sheet_representation;
 vertical_profile_height  : LENGTH_MEASURE;
END_ENTITY;

ENTITY boundary_curve_representation;
 bounds  : SET [1:?] OF CURVE;
END_ENTITY;

ENTITY calculated_state
  SUBTYPE OF (fe_analysis_state);
END_ENTITY;

ENTITY change_order
  SUBTYPE OF (work_order);
 adopted_solution  : TEXT;
```



```
 change_date       : DATE;
END_ENTITY;

ENTITY change_request
  SUBTYPE OF (work_request);
 consequence           : TEXT;
 recommended_solution  : TEXT;
 version               : LABEL;
END_ENTITY;

ENTITY component_assembly_position;
 assembly_shape   : geometric_model_representation;
 component_shape  : geometric_model_representation;
 definition       : next_higher_assembly;
 transformation   : LABEL;
END_ENTITY;

ENTITY composite_assembly
  SUBTYPE OF (constituent_part);
 layup_part  : composite_assembly_table;
 shape       : OPTIONAL composite_assembly_shape;
END_ENTITY;

ENTITY composite_assembly_sequence_definition;
 components_in_sequence  : BAG [1:?] OF component_class_for_assembly;
 properties              : OPTIONAL fea_material_property;
END_ENTITY;

ENTITY composite_assembly_table
  SUBTYPE OF (part_laminate_table);
 sequence_groups  : LIST [2:?] OF
                     composite_assembly_sequence_definition;
END_ENTITY;

ENTITY composite_sheet_representation
  SUPERTYPE OF (ONEOF (face_based_sheet_representation,
                       geometric_sheet_representation))
  SUBTYPE OF (constituent_shape_representation);
END_ENTITY;

ENTITY constituent_part
  SUPERTYPE OF (ONEOF (ply,
                       ply_piece,
                       ply_laminate,
                       filament_laminate,
                       processed_core,
                       composite_assembly));
 constituent_part_identification  : IDENTIFIER;
 of_part                          : part_version;
 weight                           : OPTIONAL MASS_MEASURE;
END_ENTITY;

ENTITY constituent_shape_representation
  SUPERTYPE OF (ONEOF (three_d_geometry_set,
                       composite_sheet_representation,
                       beveled_sheet_representation))
  SUBTYPE OF (geometric_model_representation);
END_ENTITY;
```

```
ENTITY constraint
  SUPERTYPE OF (ONEOF (single_point_constraint,
                       linear_constraint_equation,
                       nodal_degree_of_freedom_reduction));
 identification  : IDENTIFIER;
 steps           : SET [1:?] OF fe_analysis_control_step;
END_ENTITY;

ENTITY CURVE;
END_ENTITY;

ENTITY curve_cross_section;
 property  : curve_property;
 section   : OPTIONAL cross_section_shape_select;
END_ENTITY;

ENTITY curve_element
  SUBTYPE OF (element);
END_ENTITY;

ENTITY curve_property;
 defined_elements   : SET [1:?] OF curve_element;
END_ENTITY;

ENTITY curve_section_properties;
 property  : curve_property;
END_ENTITY;

ENTITY damping_matrix
  SUBTYPE OF (matrix);
END_ENTITY;

ENTITY DATE;
END_ENTITY;

ENTITY date_effectivity
  SUBTYPE OF (effectivity);
 end_date    : OPTIONAL DATE;
 start_date  : DATE;
END_ENTITY;

ENTITY design_discipline_product_definition;
 approved_by                : OPTIONAL approval;
 cad_filename               : OPTIONAL FILE;
 creation_date              : DATE;
 creator                    : person_organization;
 description                : TEXT;
 discipline_identification  : IDENTIFIER;
 version                    : part_version;
END_ENTITY;

ENTITY design_material;
 material_callout  : material_select;
 part_defined      : design_discipline_product_definition;
END_ENTITY;

ENTITY design_specification
  SUBTYPE OF (specification);
END_ENTITY;
```



```
ENTITY DIRECTION;
END_ENTITY;

ENTITY directionally_explicit_element
  SUBTYPE OF (element);
 associated_matrix  : matrix;
END_ENTITY;

ENTITY discontinuous_fiber_assembly
  SUBTYPE OF (stock_material);
END_ENTITY;

ENTITY draped_orientation_angle
  SUBTYPE OF (ply_orientation_angle);
END_ENTITY;

ENTITY edge_zone_shape
  SUBTYPE OF (zone_structural_makeup_shape_representation);
 boundary  : boundary_class_zone;
END_ENTITY;

ENTITY effectivity
  SUPERTYPE OF (ONEOF (sequence_effectivity,
                       lot_effectivity,
                       date_effectivity));
 affected_assemblies  : SET [1:?] OF assembly;
 approved_by          : OPTIONAL approval;
 configuration_item   : product_configuration;
END_ENTITY;

ENTITY element
  SUPERTYPE OF (ONEOF (volume_element,
                       surface_element,
                       substructure_element,
                       curve_element,
                       point_element,
                       directionally_explicit_element,
                       explicit_element));
 description             : OPTIONAL element_description;
 element_identification  : IDENTIFIER;
 model_ref               : fea_model;
 node_list               : LIST [1:?] OF node;
END_ENTITY;

ENTITY element_description;
 description  : TEXT;
END_ENTITY;

ENTITY element_field_variable_definition
  SUBTYPE OF (fe_analysis_state_definition);
END_ENTITY;

ENTITY element_nodal_freedom_actions
  SUBTYPE OF (fe_analysis_state_definition);
END_ENTITY;

ENTITY element_property_geometric_relationship;
 geometry_ref  : geometry_element;
 property_ref  : element_property_select;
END_ENTITY;
```

```
ENTITY element_shape_aspect;
END_ENTITY;

ENTITY element_shape_relationship;
 element_aspect     : element_shape_aspect;
 element_reference  : element;
 role               : TEXT;
 shape              : geometry_element;
END_ENTITY;

ENTITY environment;
END_ENTITY;

ENTITY explicit_element
  SUBTYPE OF (element);
 associated_matrix  : matrix;
END_ENTITY;

ENTITY face_based_sheet_representation
  SUBTYPE OF (composite_sheet_representation);
 face_surface  : SURFACE;
 inner_bounds  : OPTIONAL SET [1:?] OF LOOP;
 outer_bound   : LOOP;
END_ENTITY;

ENTITY faceted_boundary_representation
  SUBTYPE OF (geometric_model_representation);
END_ENTITY;

ENTITY fe_analysis;
 approved_by             : OPTIONAL approval;
 intended_analysis_code  : LABEL;
 model_ref               : fea_model;
END_ENTITY;

ENTITY fe_analysis_control_step
  SUPERTYPE OF (ONEOF (modes_and_frequencies_control_step,
                       static_control_step));
 analysis             : fe_analysis;
 identification       : IDENTIFIER;
 initial_input_state  : fe_analysis_state;
 sequence             : INTEGER;
END_ENTITY;

ENTITY fe_analysis_results;
 approved_by  : OPTIONAL approval;
END_ENTITY;

ENTITY fe_analysis_results_step
  SUPERTYPE OF (ONEOF (modes_and_frequencies_results_step,
                       static_results_step));
 control  : fe_analysis_control_step;
 result   : fe_analysis_results;
END_ENTITY;

ENTITY fe_analysis_state
  SUPERTYPE OF (ONEOF (calculated_state,
                       linearly_superimposed_state,
                       output_request_state,
                       specified_state));
```



```
 definitions  : SET [1:?] OF fe_analysis_state_definition;
END_ENTITY;

ENTITY fe_analysis_state_definition
  SUPERTYPE OF (ONEOF (analysis_message,
                       element_field_variable_definition,
                       element_nodal_freedom_actions,
                       linear_constraint_equation_value,
                       nodal_freedom_definitions,
                       single_point_constraint_values));
 ref  : definition_reference;
END_ENTITY;

ENTITY fea_material_definition;
 description              : TEXT;
 elements                 : SET [0:?] OF definition_element;
 material_identification  : IDENTIFIER;
END_ENTITY;

ENTITY fea_material_property
  SUBTYPE OF (material_property);
 property_use  : fea_material_definition;
END_ENTITY;

ENTITY fea_material_specification
  SUBTYPE OF (material_specification);
 specification_use  : fea_material_definition;
END_ENTITY;

ENTITY fea_model;
 analysis_type           : LABEL;
 approved_by             : OPTIONAL approval;
 creating_software       : LABEL;
 definition              : analysis_discipline_product_definition;
 description             : fea_model_description;
 fea_filename            : OPTIONAL FILE;
 identification          : IDENTIFIER;
 intended_analysis_code  : LABEL;
END_ENTITY;

ENTITY fea_model_description;
 description  : TEXT;
END_ENTITY;

ENTITY filament_assembly
  SUBTYPE OF (stock_material);
END_ENTITY;

ENTITY filament_laminate
  SUBTYPE OF (constituent_part);
 made_from  : filament_assembly;
 shape      : OPTIONAL filament_laminate_shape;
END_ENTITY;

ENTITY filament_laminate_shape;
 cross_section  : cross_section_shape_select;
END_ENTITY;

ENTITY FILE;
END_ENTITY;
```

```
ENTITY flat_pattern_ply_shape
  SUBTYPE OF (ply_shape);
 wrapup_origin_on_flat_pattern  : LOCATION;
 wrapup_origin_on_surface       : LOCATION;
END_ENTITY;

ENTITY geometric_model_representation
  SUPERTYPE OF (ONEOF (advanced_boundary_representation,
                       point_model,
                       manifold_surface_with_topology,
                       wireframe_with_topology,
                       faceted_boundary_representation,
                       non_topological_surface_and_wireframe,
                       constituent_shape_representation));
 elements  : SET [1:?] OF geometry_element;
 role      : TEXT;
END_ENTITY;

ENTITY geometric_sheet_representation
  SUBTYPE OF (composite_sheet_representation);
 basis_surface  : SURFACE;
 cutouts        : OPTIONAL SET [1:?] OF boundary_curve_representation;
 outer_edge     : boundary_curve_representation;
END_ENTITY;

ENTITY geometry_element;
END_ENTITY;

ENTITY graphical_representation
  SUBTYPE OF (analysis_report_representation);
 graphical_filename  : TEXT;
END_ENTITY;

ENTITY group;
 description           : TEXT;
 elements              : OPTIONAL SET [1:?] OF element;
 group_identification  : IDENTIFIER;
 nodes                 : OPTIONAL SET [1:?] OF node;
END_ENTITY;

ENTITY group_relationship;
 related_group   : group;
 relating_group  : group;
END_ENTITY;

ENTITY idealized_analysis_shape
  SUBTYPE OF (shape, analysis_shape);
 basis           : nominal_design_shape;
 defining_shape  : geometric_model_representation;
END_ENTITY;

ENTITY IDENTIFIER;
END_ENTITY;

ENTITY isotropic_material
  SUBTYPE OF (stock_material);
END_ENTITY;

ENTITY LABEL;
END_ENTITY;
```



```
ENTITY laid_orientation_angle
  SUBTYPE OF (ply_orientation_angle);
END_ENTITY;

ENTITY laid_ply_shape
  SUBTYPE OF (ply_shape);
END_ENTITY;

ENTITY laminate_table
  SUPERTYPE OF (ONEOF (zone_structural_makeup,
                       part_laminate_table));
 base_surface       : OPTIONAL surface_with_direction;
 basis              : reinforcement_orientation_basis;
 properties         : OPTIONAL fea_material_property;
 resulting_surface  : OPTIONAL SURFACE;
END_ENTITY;

ENTITY LENGTH_MEASURE;
END_ENTITY;

ENTITY linear_constraint_equation
  SUBTYPE OF (constraint);
 freedoms_and_coefficients_nodes : SET [1:?] OF
                                   constraint_definition_reference;
END_ENTITY;

ENTITY linear_constraint_equation_value
  SUBTYPE OF (fe_analysis_state_definition);
 equation  : linear_constraint_equation;
END_ENTITY;

ENTITY linearly_superimposed_state
  SUBTYPE OF (fe_analysis_state);
END_ENTITY;

ENTITY LOCATION;
END_ENTITY;

ENTITY LOOP;
END_ENTITY;

ENTITY lot_effectivity
  SUBTYPE OF (effectivity);
 lot_number                : NUMBER;
 lot_size                  : NUMBER;
 lot_size_unit_of_measure  : LABEL;
END_ENTITY;

ENTITY make_from;
 basis      : design_discipline_product_definition;
 resultant  : design_discipline_product_definition;
END_ENTITY;

ENTITY manifold_surface_with_topology
  SUBTYPE OF (geometric_model_representation);
END_ENTITY;

ENTITY mass_matrix
  SUBTYPE OF (matrix);
END_ENTITY;
```

```
ENTITY MASS_MEASURE;
END_ENTITY;

ENTITY material_direction;
 material_orientation  : DIRECTION;
END_ENTITY;

ENTITY material_property
  SUPERTYPE OF (fea_material_property);
 ambient         : environment;
 property_name   : LABEL;
 property_value  : OPTIONAL MEASURE_VALUE;
END_ENTITY;

ENTITY material_specification
  SUPERTYPE OF (fea_material_specification)
  SUBTYPE OF (specification);
END_ENTITY;

ENTITY matrix
  SUPERTYPE OF (ONEOF (mass_matrix,
                       stiffness_matrix,
                       damping_matrix));
END_ENTITY;

ENTITY MEASURE_VALUE;
END_ENTITY;

ENTITY modes_and_frequencies_control_step
  SUBTYPE OF (fe_analysis_control_step);
 final_input_state  : fe_analysis_state;
 frequency_range    : LIST [1:2] OF REAL;
 number_of_modes    : INTEGER;
END_ENTITY;

ENTITY modes_and_frequencies_results_step
  SUBTYPE OF (fe_analysis_results_step);
 resulting_states  : SET [1:?] OF calculated_state;
END_ENTITY;

ENTITY next_higher_assembly
  SUBTYPE OF (assembly);
 as_required           : LOGICAL;
 component_quantity    : NUMBER;
 reference_designator  : OPTIONAL LABEL;
 unit_of_measure       : LABEL;
END_ENTITY;

ENTITY nodal_degree_of_freedom_reduction
  SUBTYPE OF (constraint);
 node  : constraint_definition_reference;
END_ENTITY;

ENTITY nodal_freedom_definitions
  SUBTYPE OF (fe_analysis_state_definition);
END_ENTITY;

ENTITY nodal_results_coordinate_system;
END_ENTITY;
```



```
ENTITY node;
 description              : OPTIONAL node_description;
 location                 : geometry_element;
 model_ref                : fea_model;
 node_identification      : IDENTIFIER;
 results_coordinate_space  : OPTIONAL nodal_results_coordinate_system;
END_ENTITY;

ENTITY node_description;
 description  : TEXT;
END_ENTITY;

ENTITY node_shape
  SUBTYPE OF (shape, analysis_shape);
 defining_shape  : point_model;
END_ENTITY;

ENTITY node_shape_relationship;
 node_reference  : node;
 role            : TEXT;
 shape           : geometry_element;
END_ENTITY;

ENTITY nominal_design_shape
  SUBTYPE OF (shape);
 defining_shape  : geometric_model_representation;
 design_view     : design_discipline_product_definition;
END_ENTITY;

ENTITY non_topological_surface_and_wireframe
  SUBTYPE OF (geometric_model_representation);
END_ENTITY;

ENTITY output_request_state
  SUBTYPE OF (fe_analysis_state);
 steps  : SET [1:?] OF fe_analysis_control_step;
END_ENTITY;

ENTITY part;
 owner                 : person_organization;
 part_classification  : OPTIONAL LABEL;
 part_nomenclature    : LABEL;
 part_number          : IDENTIFIER;
 part_type            : LABEL;
 standard_part        : LOGICAL;
UNIQUE
 part_number;
END_ENTITY;

ENTITY part_laminate_table
  SUPERTYPE OF (ONEOF (composite_assembly_table,
                       ply_laminate_table))
  SUBTYPE OF (laminate_table);
END_ENTITY;

ENTITY part_version;
 approved_by       : OPTIONAL approval;
 contract_number   : OPTIONAL IDENTIFIER;
 creator           : person_organization;
 make_or_buy_code  : OPTIONAL LABEL;
```

```
 part_number       : part;
 release_status    : LABEL;
 revision_letter   : IDENTIFIER;
 security_code     : LABEL;
 weight            : OPTIONAL MASS_MEASURE;
END_ENTITY;

ENTITY PERCENTAGE;
END_ENTITY;

ENTITY percentage_laminate_table
  SUBTYPE OF (zone_structural_makeup);
 table_components   : SET [1:?] OF percentage_ply;
 total_thickness    : LENGTH_MEASURE;
END_ENTITY;

ENTITY percentage_ply;
 strength_orientation   : OPTIONAL ply_orientation_angle;
 material               : stock_material;
 makeup_and_properties  : OPTIONAL zone_structural_makeup;
 volume_percent         : PERCENTAGE;
END_ENTITY;

ENTITY person_organization;
 address                             : LABEL;
 organization                        : LABEL;
 person                              : LABEL;
 person_organization_identification  : IDENTIFIER;
END_ENTITY;

ENTITY ply
  SUBTYPE OF (constituent_part);
 constituents          : OPTIONAL LIST [1:?] OF ply_piece;
 material_type         : ply_stock_material;
 material_orientation  : ply_orientation_angle;
 ply_thickness         : LENGTH_MEASURE;
 shape                 : OPTIONAL ply_shape;
END_ENTITY;

ENTITY ply_laminate
  SUBTYPE OF (constituent_part);
 ply_table  : ply_laminate_table;
 shape      : OPTIONAL ply_laminate_shape;
END_ENTITY;

ENTITY ply_laminate_sequence_definition;
 plies_in_sequence  : SET [1:?] OF ply;
 properties         : OPTIONAL fea_material_property;
END_ENTITY;

ENTITY ply_laminate_table
  SUBTYPE OF (part_laminate_table);
 sequence  : LIST [2:?] OF ply_laminate_sequence_definition;
END_ENTITY;

ENTITY ply_orientation_angle
  SUPERTYPE OF (ONEOF (draped_orientation_angle,
                      laid_orientation_angle));
 angle_reference  : angle_select;
 basis            : reinforcement_orientation_basis;
```



```
END_ENTITY;

ENTITY ply_piece
  SUBTYPE OF (constituent_part);
 shape          : OPTIONAL ply_piece_shape;
 warp_surface  : LOGICAL;
END_ENTITY;

ENTITY ply_shape
  SUPERTYPE OF (ONEOF (projected_ply_shape,
                       laid_ply_shape,
                       flat_pattern_ply_shape));
 basis           : OPTIONAL ply_shape;
 defining_model  : composite_sheet_representation;
END_ENTITY;

ENTITY POINT;
END_ENTITY;

ENTITY point_and_vector;
 location  : POINT;
 vector    : LIST [1:2] OF DIRECTION;
END_ENTITY;

ENTITY point_element
  SUBTYPE OF (element);
 associated_matrix  : matrix;
END_ENTITY;

ENTITY point_model
  SUBTYPE OF (geometric_model_representation);
END_ENTITY;

ENTITY point_path;
 directions  : LIST [1:?] OF point_and_vector;
END_ENTITY;

ENTITY point_zone_shape
  SUBTYPE OF (zone_structural_makeup_shape_representation);
 location  : POINT;
END_ENTITY;

ENTITY process_specification
  SUBTYPE OF (specification);
END_ENTITY;

ENTITY processed_core
  SUBTYPE OF (constituent_part);
 added_material             : OPTIONAL LIST [0:?] OF ply;
 cell_direction             : ply_orientation_angle;
 made_from_processed_core  : OPTIONAL SET [1:?] OF processed_core;
 made_from_stock            : stock_core;
 shape                      : OPTIONAL core_shape;
END_ENTITY;
```

```
ENTITY product_configuration;
 approved_by          : OPTIONAL approval;
 item_identification  : IDENTIFIER;
 model_name           : product_model;
 parts_configured     : OPTIONAL SET [1:?] OF part;
 phase_of_product     : LABEL;
UNIQUE
 item_identification;
END_ENTITY;

ENTITY product_model;
 model_name  : LABEL;
END_ENTITY;

ENTITY projected_ply_shape
  SUPERTYPE OF (ONEOF (view_ply_shape,
                       surface_ply_shape))
  SUBTYPE OF (ply_shape);
 method                : projection_method;
 projection_direction  : OPTIONAL DIRECTION;
END_ENTITY;

ENTITY promissory_usage
  SUBTYPE OF (assembly);
END_ENTITY;

ENTITY reinforcement_orientation_basis;
 basis_location  : LOCATION;
 orientation     : DIRECTION;
END_ENTITY;

ENTITY retention_period;
 approved_by              : OPTIONAL approval;
 earliest_end_definition  : DATE;
 is_applied_to            : SET [1:?] OF retention_data_select;
 latest_end_definition    : DATE;
 retention_purpose        : OPTIONAL TEXT;
 start_definition         : DATE;
END_ENTITY;

ENTITY sequence_effectivity
  SUBTYPE OF (effectivity);
 component_quantity               : NUMBER;
 quantity_unit_of_measure         : LABEL;
 from_effectivity_identification  : NUMBER;
 to_effectivity_identification    : NUMBER;
END_ENTITY;

ENTITY shape
  SUPERTYPE OF (ONEOF (nominal_design_shape,
                       node_shape,
                       idealized_analysis_shape));
 role  : TEXT;
END_ENTITY;

ENTITY shape_aspect;
 characteristics  : SET [0:?] OF specification;
 geometry         : geometric_model_representation;
 parent_shape     : shape;
END_ENTITY;
```



```
ENTITY single_point_constraint
  SUBTYPE OF (constraint);
 required_node  : constraint_definition_reference;
END_ENTITY;

ENTITY single_point_constraint_values
  SUBTYPE OF (fe_analysis_state_definition);
 element  : single_point_constraint;
END_ENTITY;

ENTITY smeared_material
  SUBTYPE OF (zone_structural_makeup);
 total_thickness  : LENGTH_MEASURE;
END_ENTITY;

ENTITY specification
  SUPERTYPE OF (ONEOF (design_specification,
                      surface_finish_specification,
                      process_specification,
                      material_specification));
 specification_code    : TEXT;
 specification_source  : TEXT;
END_ENTITY;

ENTITY specified_state
  SUBTYPE OF (fe_analysis_state);
END_ENTITY;

ENTITY start_order
  SUBTYPE OF (work_order);
END_ENTITY;

ENTITY start_request
  SUBTYPE OF (work_request);
END_ENTITY;

ENTITY static_control_step
  SUBTYPE OF (fe_analysis_control_step);
 final_input_state  : fe_analysis_state;
END_ENTITY;

ENTITY static_results_step
  SUBTYPE OF (fe_analysis_results_step);
 resulting_state  : calculated_state;
END_ENTITY;

ENTITY stiffness_matrix
  SUBTYPE OF (matrix);
END_ENTITY;

ENTITY stock_core
  SUBTYPE OF (stock_material);
END_ENTITY;

ENTITY stock_material
  SUPERTYPE OF (ONEOF (anisotropic_material,
                       discontinuous_fiber_assembly,
                       filament_assembly,
                       isotropic_material,
```

```
                        stock_core));
 approved_by          : OPTIONAL approval;
 property             : material_property;
 reference_direction  : material_direction;
 of_part              : part_version;
 specified_material   : material_specification;
END_ENTITY;

ENTITY substitute_part;
 base        : assembly;
 substitute  : part;
END_ENTITY;

ENTITY substructure_element
  SUBTYPE OF (element);
 substructure_model_ref  : fea_model;
END_ENTITY;

ENTITY substructure_node_relationship;
 related_node   : node;
 relating_node  : node;
END_ENTITY;

ENTITY supplied_part_version;
 approved_by             : OPTIONAL approval;
 certification_required  : LOGICAL;
 is_identified_as        : part_version;
 produced_by             : supplier;
 supplier_part_number    : OPTIONAL IDENTIFIER;
END_ENTITY;

ENTITY supplier;
 identified_as            : person_organization;
 supplier_identification  : IDENTIFIER;
END_ENTITY;

ENTITY SURFACE;
END_ENTITY;

ENTITY surface_element
  SUBTYPE OF (element);
END_ENTITY;

ENTITY surface_finish_specification
  SUBTYPE OF (specification);
END_ENTITY;

ENTITY surface_ply_shape
  SUBTYPE OF (projected_ply_shape);
 offset        : LENGTH_MEASURE;
 surface_role  : LABEL;
END_ENTITY;

ENTITY surface_property;
 defined_elements  : SET [1:?] OF surface_element;
END_ENTITY;
```



```
ENTITY surface_thickness;
 property    : surface_property;
 thickness   : LENGTH_MEASURE;
END_ENTITY;

ENTITY surface_with_direction;
 defining_surface    : SURFACE;
 material_direction  : DIRECTION;
END_ENTITY;

ENTITY tabular_representation
  SUBTYPE OF (analysis_report_representation);
 tabular_filename  : TEXT;
END_ENTITY;

ENTITY TEXT;
END_ENTITY;

ENTITY thickness_laminate_table
  SUBTYPE OF (zone_structural_makeup);
 table_components  : LIST [2:?] OF thickness_laminate_table_component;
END_ENTITY;

ENTITY three_d_geometry_set
  SUBTYPE OF (constituent_shape_representation);
 basis_role         : LABEL;
 basis_surface      : SURFACE;
 defining_boundary  : SET [1:?] OF CURVE;
END_ENTITY;

ENTITY usage_constraint;
 constrains     : specification;
 usage_element  : TEXT;
 usage_value    : TEXT;
END_ENTITY;

ENTITY view_ply_shape
  SUBTYPE OF (projected_ply_shape);
END_ENTITY;

ENTITY volume_element
  SUBTYPE OF (element);
END_ENTITY;

ENTITY wireframe_with_topology
  SUBTYPE OF (geometric_model_representation);
END_ENTITY;

ENTITY work_order
  SUPERTYPE OF (ONEOF (start_order,
                       change_order));
 additional_data             : OPTIONAL FILE;
 analysis_data               : OPTIONAL FILE;
 approved_by                 : OPTIONAL approval;
 incorporates                : SET [1:?] OF work_request;
 versions                    : SET [1:?] OF design_or_analysis;
 work_order_identification  : IDENTIFIER;
UNIQUE
 work_order_identification;
END_ENTITY;
```

```
ENTITY work_request
  SUPERTYPE OF (ONEOF (change_request,
                       start_request));
 affective_parts               : OPTIONAL SET [1:?] OF design_or_analysis;
 approved_by                   : OPTIONAL approval;
 description                   : TEXT;
 recipients                    : SET [1:?] OF person_organization;
 reason                        : TEXT;
 request_date                  : DATE;
 status                        : LABEL;
 work_request_identification   : IDENTIFIER;
UNIQUE
 work_request_identification;
END_ENTITY;

ENTITY zone_structural_makeup
  SUPERTYPE OF (ONEOF (thickness_laminate_table,
                       percentage_laminate_table,
                       smeared_material,
                       (thickness_laminate_table AND
                        smeared_material),
                       (percentage_laminate_table AND
                        smeared_material)))
  SUBTYPE OF (laminate_table);
 boundary  : OPTIONAL zone_structural_makeup_shape_representation;
END_ENTITY;

ENTITY zone_structural_makeup_shape_representation
  SUPERTYPE OF (ONEOF (point_zone_shape,
                       edge_zone_shape));
END_ENTITY;

END_SCHEMA;
(*
```

Bibliography

The following are informative references:

1. *Guidelines for the Development and Approval of STEP Application Protocols*, Version 1.1, ISO TC184/SC4/WG4 N66, January 1993.

2. IDEF0 *Federal Information Processing Standards Publication 183, Integration Definition for Function Modeling (IDEF0)*, FIPS PUB 183, National Institute of Standards and Technology, December 1993.

3. Marks' Standard Handbook for Mechanical Engineers (Avallone and Baumeister - Editors), Ninth Edition, McGraw-Hill, 1987.

Index

ICS 25.040.40

Price based on 966 pages