INTERNATIONAL STANDARD

ISO 10303-204

First edition
2002-08-15

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

Part 204: Application protocol: Mechanical design using boundary representation

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

Partie 204: Protocole d'application: Conception mécanique utilisant une représentation délimitée

Reference number
ISO 10303-204:2002(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.

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

Attention is drawn to the possibility that some of the elements of this 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 10303-204was prepared by Technical Committee ISO TC184/SC4. *Industrial automation systems and integration,* Subcommittee SC4 *Industrial data.*

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

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

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

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

Annexes A, B, C, D and E form an integral 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 is suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases and as a basis for archiving.

This part of ISO 10303 is a member of the application protocol series.

This Part of ISO 10303 specifies an application protocol (AP) for mechanical design using boundary representation solid models. A boundary representation solid model provides a complete descripton of the shape of a solid object by describing precisely the geometry and topology of all its internal and external boundaries.

This application protocol defines the context, scope, and information requirements for mechanical design using boundary representation models and specifies the integrated resources necessary to satisfy these requirements.

Application protocols provide the basis for developing implementations of ISO 10303. Application protocols provide the basis for developing 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. 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 definitions and the EXPRESS provided in the integrated resources for constructs used in the AIM may include select list items and subtypes not imported into the AIM. The expanded listing given in Annex A contains the complete EXPRESS of 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.

This Part of ISO 10303 contains the definition of conforming boundary representation solid models and the mechanisms to transfer them via an exchange structure as defined in Part ISO 10303-21. The exchange of such models, with associated visual presentation information is required during the initial desgn of a mechanical product and when detailed designs of components are communicated to suppliers and sub-contractors. In this Part B-reps are characterised by the fact that they can represent models with only planar surfaces (faceted B-rep), models with only analytical surfaces (elementary B-rep) and models with sculptured surfaces and curves (advanced B-rep). The application reference environment in which these B-rep models are used is the generation and exchange of volume-based data in the Computer-aided Mechanical design process. This application places fundamental requirements on the model exchange

and the neutral representation of models. The transfer and archiving of B-rep models at different stages of the design and engineering process requires the following to be maintained:

— the completeness of the models when mapped between application systems;

— the correctness of semantics of the representation;

— the accuracy of the geometric relationships between entity instances which form part of a B-rep model; in particular all vertices shall lie on the edges using them and all edge_curves shall lie on each face using this edge as part of the boundary.

Three different classes of implementation are specified in clause 6.

This application protocol was developed as one component of a series of Mechanical Design application protocols and is complemented by ISO 10303-205 Mechanical design using surface models, see (1). These Parts share a common application environment and have a similar scope for the representation of mechanical parts. The significant differences among these Parts of ISO 10303 is in the manner in which the shape of a mechanical part is represented. In this Part the representation is as a manifold solid boundary representation model. In ISO 10303-205 the shape of the part is represented by a surface model in which all surfaces and bounding curves are fully represented. Figure 1 gives a pictorial representation of the scope of this AP.

NOTE 1 In figure 1 the term scope refers to the intended scope of the information models in this Part of ISO 10303. These information models may be useful as part of an information model for applications shown as 'out of scope' in this diagram.

Figure 2 contains the data planning model that gives a high level description of the requirements for this application protocol, as well as the relationships between the basic data objects.

NOTE 2 A dashed line in figure 2 is used to denote an optional association.

The planning model illustrates that a product may be either a part or an assembly. The shape of a part or assembly is represented by a shape model which takes the form of one, or more, B-reps. Each B-rep is either a faceted B-rep, an elementary B-rep, or an advanced B-rep. Names can be associated with products, parts or shape models. Visual properties may optionally be attached to B-rep models.

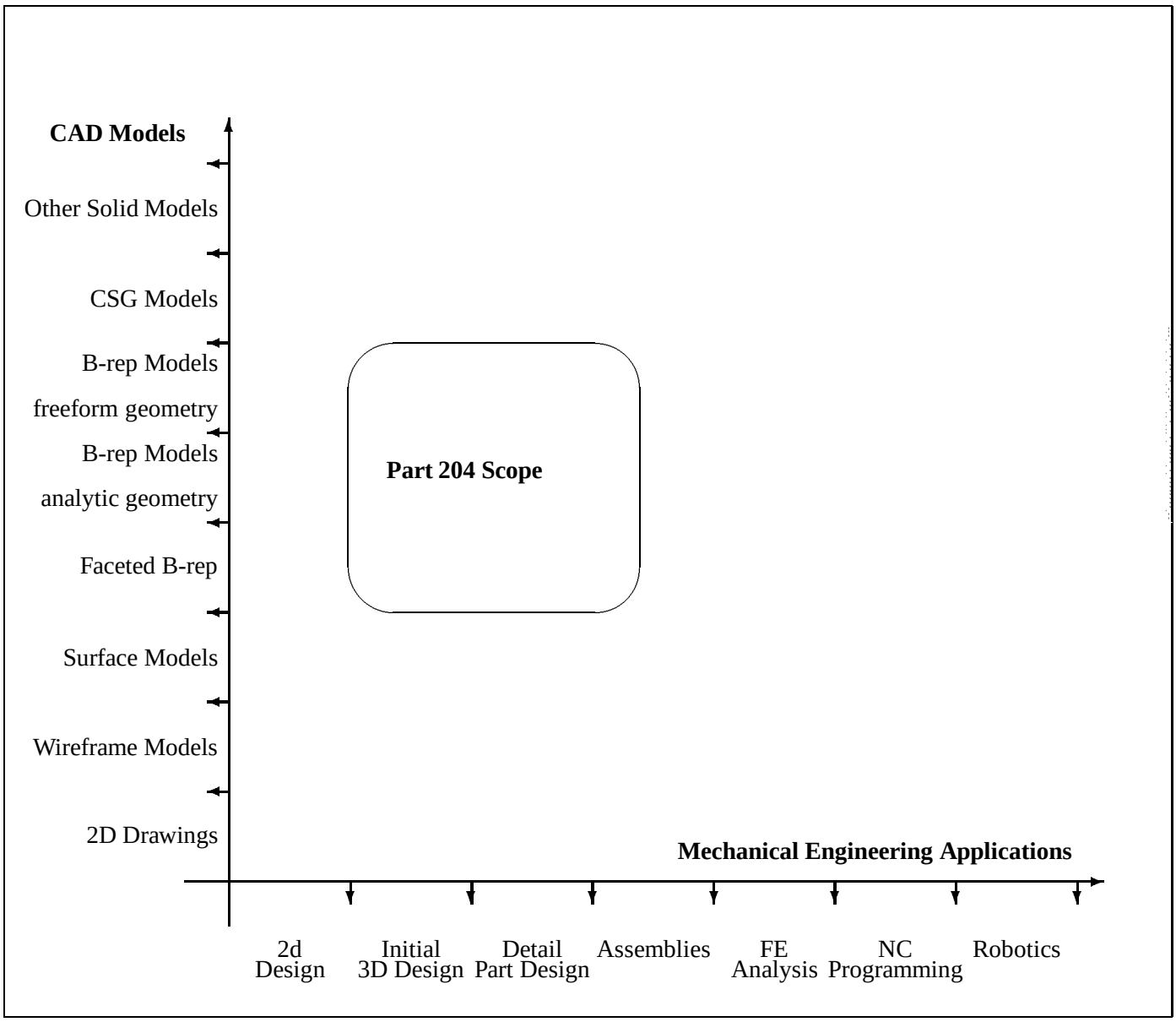


Figure 1 – The scope of this part of ISO 10303 in the contexts of CAD models and mechanical engineering applications

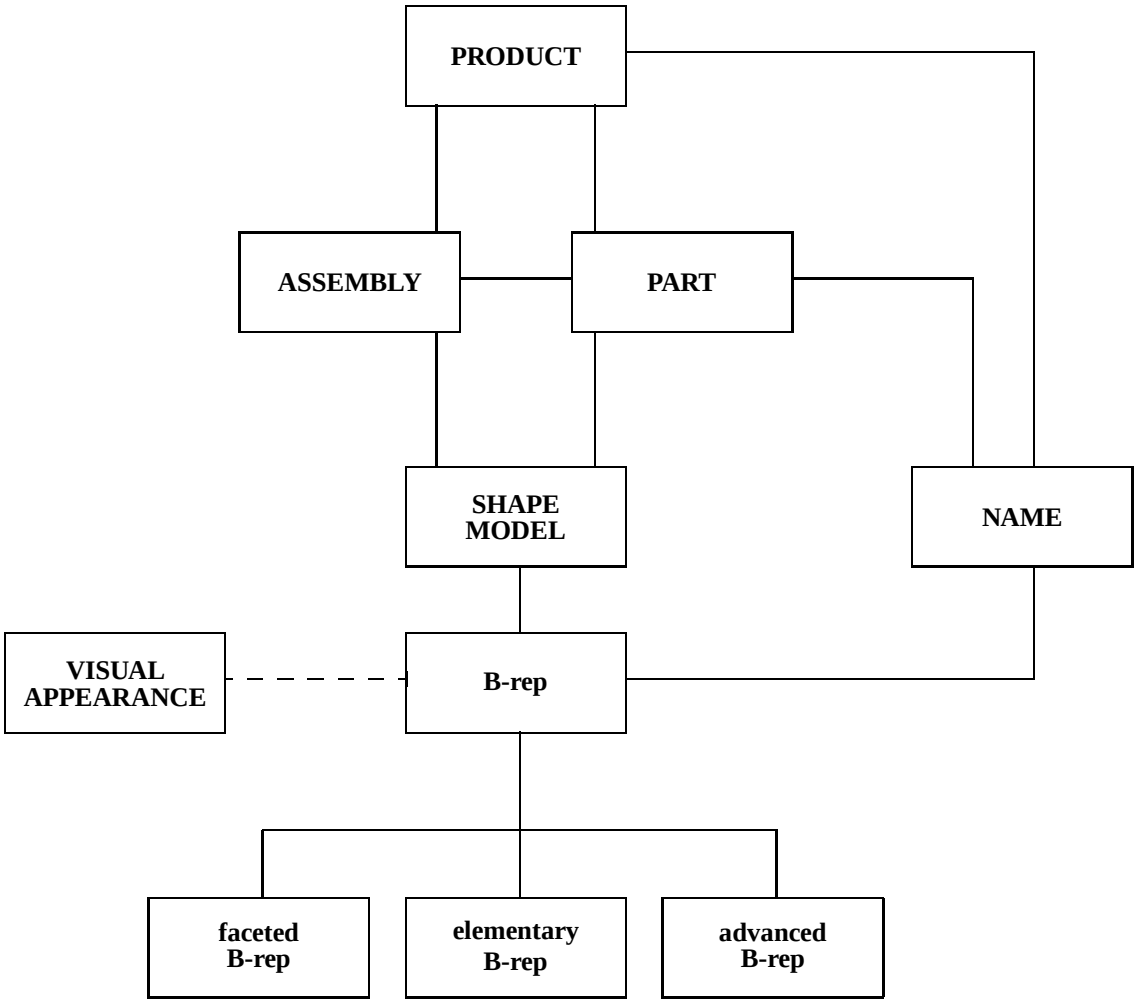


Figure 2 – Data planning model

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

Part 204: Application protocol: Mechanical design using boundary representation

1 Scope

This part of ISO 10303 specifies the use of the integrated resources necessary for the scope and information requirements for the use and exchange of boundary representation solid models in the mechanical engineering design context.

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

This document describes an application reference environment for the generation and exchange of volume-based design data in the computer-aided mechanical design process, together with appropriate data models and a physical file implementation form. The information model supports all geometric and topological aspects of a complete description of the shape and size of an object. It was originally developed for applications in mechanical engineering design using the CAD modelling technique boundary representation (B-rep) solid modelling and may be appropriate for other application areas using this technique.

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

— Three types of B-rep model that are used to represent shape:

  a) faceted B-rep model;

  b) B-rep model with elementary surfaces;

  c) B-rep model with sculptured surfaces;

— curve and surface geometry;

— curves defined in parameter space (pcurves);

— manifold topology;

— product identification information;

— the association of simple presentation attributes such as line-style, line-width, colour with an entire B-rep model, or, with geometric or topological elements of a B-rep model;

— preservation of user-defined names of objects;

— units and measures associated with geometric elements;

— assemblies of parts and sub-assemblies.

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

— Other types of shape representation:

  a) wireframe models;

  b) surface models;

  c) geometrically trimmed curves and surfaces;

  d) constructive solid geometry models;

  e) compound B-rep models.

— Geometric and topological data:

  a) 2D geometry, other than for the definition of pcurves;

  b) self-intersecting geometry;

  c) non-manifold topology.

— Dimensioning;

— Tolerances;

— Manufacturing information;

— Advanced presentation features such as multiple views, character fonts and symbols.

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 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: Implementation methods: Clear text encoding of the exchange structure*

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

ISO 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 10303-46:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 46: Integrated generic resources: Visual presentation.*

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

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

ISO 10303-513:2000, *Industrial automation systems and integration — Product data representation and exchange — Part 513: Application interpreted construct: Elementary boundary representation*

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

ISO 10303-517:2000, *Industrial automation systems and integration — Product data representation and exchange — Part 517: Application interpreted construct: Mechanical design geometric presentation*

ISO 10303-518[1)], *Industrial automation systems and integration — Product data representation and exchange — Part 518: Application interpreted construct: Mechanical design shaded presentation*

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

[1)] To be published.

3 Terms, definitions, and abbreviations

3.1 Terms defined in ISO 10303-1

For the purposes of this part of ISO 10303, the following terms defined in ISO 10303-1 apply.

— abstract test suite;

— application;

— application activity model (AAM);

— application context;

— application interpreted model (AIM);

— application object;

— application protocol (AP);

— application reference model (ARM);

— assembly;

— component;

— conformance class;

— conformance requirement;

— conformance testing;

— context;

— data;

— data exchange;

— implementation method;

— interpretation;

— integrated resource;

— model;

— PICS proforma;

— presentation;

— product;

— product data;

— resource construct;

— structure;

— unit of functionality (UoF);

3.2 Terms defined in ISO 10303-42

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

— arcwise connected;

— boundary;

— boundary representation solid model;

— bounds;

— curve;

— euler equations;

— inside;

— interior;

— parameter space;

— surface;

— topological sense.

3.3 Terms defined in ISO 10303-44

For the purposes of this part of ISO 10303, the following terms defined in ISO 10303-44 apply.

— bill_of_material structure;

— component;

— constituent;

— form, fit and function.

3.4 Other definitions

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

3.4.1
advanced B-rep
general boundary representation model which may have any geometric form for the faces and edges. In particular B-splines may be used to define the face and edge geometry.

3.4.2
computer-aided design
way of designing mechanical and other products utilizing computerized tools.

3.4.3
elementary B-rep
boundary representation model in which all surfaces are planar, cylindrical, conical, spherical or toroidal.

3.4.4
elementary geometry
geometry composed of lines, polylines, conics and elementary_surfaces.

3.4.5
faceted B-rep
simplified boundary representation model with planar faces and implicitly defined edges and vertices.

3.4.6
functional level
indicator used to distinguish geometric complexity of a B-rep model or other representation.

3.4.7
genus
topological property used to classify solids. The genus of a solid is an abstraction for the number of through holes in the solid.

3.4.8
manifold solid
arcwise connected solid such that, the interior of any infinitessimally small sphere, centred at any point on the boundary of the solid, is divided into precisely 2 regions, inside and outside the solid respectively.

3.4.9

normal direction
unit vector perpendicular to a surface and pointing away from the material.

3.4.10
orientation
mathematical sense of curves, loops or edges.

3.4.11
product version
identifier for a variant of a product.

3.5 Abbreviations

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

B-rep	Boundary representation solid
CAE	Computer Aided Engineering
CIM	Computer Integrated Manufacturing
CSG	Constructive Solid Geometry
FEA	Finite Element Analysis
ID	Identification
IDEF0	ICAM definition language 0
NIAM	Nijssen's Information Analysis Method
NC	Numerical Control

4 Information requirements

This clause specifies the information required for mechanical design using boundary representation.

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

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

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

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

These requirements apply to system developers developing conforming implementations and to users of this application protocol to exchange physical files containing B-rep model data. An implementation claiming to conform to this application protocol shall ensure that the structure and constraints defined by these information requirements are satisfied when physical files are exchanged.

Functional Levels

The information requirements for mechanical design using boundary representation models are presented in terms of three distinct levels of functionality. The goal is to classify different implementations into levels distinguished by the complexity of the shape being represented.

The shape of each part described in this AP is composed of geometry and topology. The topology structure provides the connectivity and trimming information for the unbounded geometry of the part. In this Application Protocol the use of a topological entity requires that all associated geometry be defined. In order to classify different levels of design-shape complexity the criterion used is complexity of surface geometry. Level 1 has simple surface geometry for each face of the model, and much of the topological information is implicit. Both level 2 and level 3 provide for a complete explicit representation of the topology of the part in which all vertices, edges, loops and faces are included. The only distinction between level 2, and level 3 is in the complexity of the geometric curves and surfaces which are associated with the topological data. There is no distinction in topology structures between level 3 and level 2.

In this part of ISO 10303 three levels of complexity are defined.

B-rep level 1: Level 1 geometric complexity is for faceted B-rep models with planar surfaces as the bounding surfaces. Only points and planar polygons which can be implicitly represented by their vertex points are necessary for this representation.

At level 1 much of the topological information is implicit. Edge and vertex information is not given and the shells consist of faces bounded by polygons. The face geometry is implicitly planar. The complete part shape is represented by the polyhedron.

EXAMPLE 1 box shapes;

EXAMPLE 2 faceted shape approximating a model of more complex shape.

Level 1 models can either be an exact model of a simple part or a simplified model of a more complex part which is suitable for a selected range of applications such as stereolithography, or finite element analysis.

Level 1 models can be represented in a more compact form than models from level 2 or level 3: edges and curves are not required to be explicitly defined, since these are always straight lines; the connecting points are sufficient for their definition.

EXAMPLE 3 Applications of these models:

a) in rapid prototype manufacturing;

b) for visualization purposes;

c) for collision checks of parts;

d) for kinematic studies;

e) for robot programming and simulations.

B-rep level 2: Level 2 of geometric complexity is for models with elementary surfaces. In this level the geometry needed to represent the curves and surfaces of objects is elementary analytic geometry. The surfaces included at this level are the plane, sphere, cylinder, cone, and torus. The curves are lines and conics. Both curves and surfaces are unbounded, and the bounding information is contained in the topology data. At this level the complete part shape is represented by an elementary B-rep model.

EXAMPLE 4 Application examples:

— milled parts suitable for $2\frac{1}{2}$D manufacturing;

— turned parts.

EXAMPLE 5 Part examples:

— bolts and screws (excluding the thread detail);

— piston of a simple piston-engine;

— motor housings.

NOTE 4 For the same part there can exist different model representations, which correspond to different application requirements.

NOTE 5 For a part there might exist a representation of level 1 and of level 2.

B-rep level 3: Level 3 of geometric complexity is for B-rep models with advanced surface descriptions.

This level will be used for modelling of parts whose geometric shape is representable with elementary , or sculptured surfaces, or swept surfaces with linear or rotational extrusions, or any combination of these. The generator curves for the extrusion can be analytic or free-form curves. The sculptured surfaces or free-form curves will be B-spline based. Level 3 includes more general forms of twisted curve and sculptured surface in addition to all those included in level 2.

EXAMPLE 6 Application examples:

— parts which require 3 to 5 axis NC machining for their manufacturing;

— dies for moulding;

— dies for forming;

— ergonomically formed consumer products.

EXAMPLE 7 Part examples:

— plastic housing of a telephone;

— car surface parts like fenders;

— housing block of a combustion engine.

NOTE 6 Level 3 is a superset of level 2 in the sense that all entities supported at level 2 are also supported at level 3.

NOTE 7 Level 3 models contain surfaces that may have any shape.

4.1 Units of functionality

This clause specifies the units of functionality for the mechanical design using boundary representation application protocol. This part of ISO 10303 specifies the following units of functionality:

— faceted_B-rep;

— elementary_B-rep;

— advanced_B-rep;

— name_preservation;

— product_structure;

— visual_presentation_for_B-rep.

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

4.1.1 faceted_B-rep

The faceted B-rep UoF provides the ability to define a boundary representation model all of whose faces are planar.

The following application objects are used by the faceted_B-rep UoF.

— B-rep;

— Direction;

— Face;

— Faceted_B-rep;

— Geometric_element;

— Global_unit;

— Location;

— Loop;

— Plane;

— Point;

— Poly_loop;

— Shell;

— Surface;

— Topological_element;

— Void.

4.1.2 elementary_B-rep

The elementary_B-rep UoF provides for the definition of a boundary representation model composed of shells having topologically bounded elementary surfaces as faces.

The following application objects are used by the elementary_B-rep UoF.

— B-rep;

— Bounded_curve;

— Circle;

— Conic;

— Conical_surface;

— Curve;

— Cylindrical_surface;

— Direction;

— Edge;

— Elementary_B-rep;

— Elementary_surface;

— Ellipse;

— Face;

— Geometric_element;

— Global_unit;

— Hyperbola;

— Line;

— Location;

— Loop;

— Parabola;

— Plane;

— Point;

— Polyline;

— Shell;

— Spherical_surface;

— Surface;

— Topological_element;

— Toroidal_surface;

— Unbounded_curve;

— Vertex;

— Void.

4.1.3 advanced_B-rep

The advanced_B-rep UoF provides the ability to define a boundary representation model with topologically bounded surfaces as faces and sculptured geometry.

The following application objects are used by the advanced_B-rep UoF.

— Advanced_B-rep;

— B-rep;

— Bounded_curve;

— Circle;

— Conic;

— Conical_surface;

— Curve;

— Cylindrical_surface;

— Degenerate_toroidal_surface;

— Direction;

— Edge;

— Elementary_surface;

— Ellipse;

— Face;

— Geometric_element;

— Global_unit;

— Hyperbola;

— Line;

— Location;

— Loop;

— Parabola;

— Pcurve

— Plane;

— Point;

— Polyline;

— Sculptured_surface;

— Shell;

— Spherical_surface;

— Surface;

— Surface_curve;

— Surface_of_extrusion;

— Surface_of_revolution;

— Swept_surface;

— Topological_element;

— Toroidal_surface;

— Twisted_curve;

— Unbounded_curve;

— Vertex;

— Void.

4.1.4 name_preservation

The name_preservation UoF provides the ability to associate and preserve user defined names with models or model components. This allows for the preservation of entity names that were defined by a user by means of a computer-aided application. The user-defined

name of an item is used as an alias to any implementation-dependent identifiers.

The following application objects are used by the name_preservation UoF.

— Name.

4.1.5 product_structure

The product_structure UoF provides the ability to define a product as an assembly of parts or of sub-assemblies. In this AP each part is defined as a B-rep model. Products are composed of individual parts and of collections of parts which form so called assemblies. Assemblies may consist of sub-assemblies and of individual parts. Individual parts are represented by specific geometric shape descriptions as B-rep models. Assemblies have specific geometric relationships with one another and to individual parts. This UoF includes the structures for the identification of mechanical parts assemblies and the structure that links the shape of the parts and assemblies to their identification.

These relationships are given by the following properties:

— a reference from an assembly to another assembly or to a part;

— a geometric relationship, which may be described with a transformation matrix, which allows translation, rotation, and, if required, mirroring and scaling;

— the assemblies and parts have names which a user may require to be unique within one product assembly.

The following detailed requirements are met by this UoF:

— the structural description of assemblies and parts with references to shapes;

— the assembly structure which is independent of the referenced shape of the product;

— the versioning of parts;

— the identification of multiple definitions of parts;

— the specification of the relationship between parts which comprise an assembly;

— the specification of the relationship between a part and its shape representation;

— the specification of the relationship of a part shape to the shape representation of an assembly.

The following application objects are used by the product_structure UoF.

— Assembly;

— Part;

— Product;

— Shape_representation;

— Transformation.

4.1.6 visual_presentation_for_B-rep

The visual_presentation_for_B-rep UoF consists of application objects for the association of elementary viewing information with geometric or topological components of B-rep models. This enables the user to specify the visual appearance of points, curves, and surfaces and to assign pre-defined colour, line-style (dashed, dotted and similar) and line-width to any geometric or topological entity.

The following application objects are used by the visual_presentation_for_B-rep UoF.

— 3D_projection;

— Curve_appearance;

— Light_source;

— Point_appearance;

— Presentation_appearance;

— Screen_image;

— Surface_appearance.

4.1.7 Relationship of units of functionality to functional levels

NOTE Table 1 shows the relationship between the units of functionality in this AP and the functional levels used in the definition of conformance classes.

B-rep level 1	**B-rep level 2**	**B-rep level 3**
name preservation		
product structure		
visual presentation for B-rep		
faceted B-rep	elementary B-rep	advanced B-rep

Table 1 – Use of units of functionality within functional levels

4.2 Application objects

This clause specifies the application objects for the mechanical design using boundary representation application protocol. Each application object is an atomic element which embodies a unique application concept and contains attributes specifying the data elements of the object. The application objects and their definitions are given below.

4.2.1 3D_projection

A 3D_projection is a is a type of Presentation_appearance (see 4.2.36) that is a 2-dimensional picture of a 3-dimensional shape. The picture is the image of a mapping defined by a camera model.

4.2.2 Advanced_B-rep

An Advanced_B-rep is a type of B-rep (see 4.2.4) which may include elementary geometry, sculptured geometry, and swept geometry.

4.2.3 Assembly

An Assembly is a collection of assemblies or parts which are connected together. It has a name assigned, which may describe the functionality, and uses transformations to position its sub-assemblies or components in space.

The data associated with an Assembly are the following:

— components;

— coordinate_system;

— user_defined_name.

4.2.3.1 components

The components specifies the constituents of the assembly; these may be individual parts or other assemblies.

4.2.3.2 coordinate_system

The coordinate_system specifies the Cartesian coordinate system used to define the geometry of the Assembly. This is the underlying global rectangular Cartesian coordinate system to which all geometry refers. A coordinate_system is identified with the context of the shape representation for an Assembly.

4.2.3.3 user_defined_name

The user_defined_name specifies a user-defined identifier for the Assembly. It consists of one or more words and may describe the functionality of the Assembly.

4.2.4 B-rep

A B-rep is a manifold boundary representation which defines a volume in terms of topology and geometry. The geometry is used to describe the shape of the object in terms of surfaces and edge-curves. The topology bounds the geometry to a finite extent and combines all bounded elements to define a manifold solid. The material side of the solid is defined by normal vectors to the faces; this normal shall point away from the material. Each B-rep is either an Advanced_B-rep (see 4.2.2), an Elementary_B-rep (see 4.2.15), or a Faceted_B-rep (see 4.2.19).

The data associated with a B-rep are the following:

— boundary;

— voids.

4.2.4.1 boundary

The boundary specifies the closed Shell (see 4.2.41) that is the outer boundary of the B-rep.

4.2.4.2 voids

The voids specifies a list of Voids (see 4.2.55) inside the B-rep. The voids need not be specified for a particular B-rep if it has a completely solid interior.

4.2.5 Bounded_curve

A Bounded_curve is a type of Curve (see 4.2.9) with finite extent and identifiable end points. Each Bounded_curve is either a Polyline (see 4.2.34) or a Twisted_curve (see 4.2.52).

4.2.6 Circle

A Circle is a type of Conic (see 4.2.7) generated by intersecting a conical surface with a plane perpendicular to the axis of the conical surface.

The data associated with a Circle are the following:

— radius;

— location.

4.2.6.1 radius

The radius specifies the distance of all points on the Circle from the centre.

4.2.6.2 location

The location specifies the centre point of a Circle and the normal direction to the plane of the Circle.

4.2.7 Conic

A Conic is a type of Unbounded_curve (see 4.2.53), it is a planar curve which may be produced as the intersection of a plane and a conical surface, where the axis of the Conic does not lie in the plane. Each Conic is either a Circle (see 4.2.6), an Ellipse (see 4.2.17), an Hyperbola (see 4.2.22), or a Parabola (see 4.2.28).

4.2.8 Conical_surface

A Conical_surface is a type of Elementary_surface (see 4.2.16) constructed by moving a line, which is fixed in one point, the vertex point, along a closed circle.

4.2.9 Curve

A Curve is a type of Geometric_element (see 4.2.20), it is a one-dimensional manifold in a space of dimension 2 or 3. Informally, a Curve can be envisioned as the path of a point moving in its coordinate space. Each Curve is either a Bounded_curve (see 4.2.5), a Pcurve (see 4.2.30), a Surface_curve (see 4.2.45), or an Unbounded_curve (see 4.2.53).

NOTE see figure G.9 in the NIAM diagrams.

4.2.10 Curve_appearance

Curve_appearance is a type of Presentation_appearance (see 4.2.36) that specifies the required appearance of a Curve (see 4.2.9) when it is visualised.

The data associated with a Curve_appearance are the following:

— colour;

— curve_font;

— curve_width.

4.2.10.1 colour

The colour defines the colour of the Curve when viewed with a colour display. It is specified by intensity values for red, green and blue. The colour need not be specified for a particular Curve, in which case the default option is black.

4.2.10.2 curve_font

The curve_font describes the lengths of the visible and invisible segments of a curve when visualised. The curve_font need not be specified for a particular Curve, in which case the default option is fully_visible.

4.2.10.3 curve_width

The curve_width specifies the apparent width of the curve when visualised. The curve_width need not be specified for a particular Curve, in which case the default option is that of the graphics system.

4.2.11 Cylindrical_surface

A Cylindrical_surface is a type of Elementary_surface (see 4.2.16) constructed by the parallel movement of a line along a closed circle, where the line is perpendicular to the plane of the circle.

The data associated with a Cylindrical_surface are the following:

— radius;

— axis.

4.2.11.1 radius

The radius defines the distance of points on the cylindrical_surface from the axis.

4.2.11.2 axis

The axis is the central axis of the cylindrical_surface, it is defined by the axis direction, and a point on the axis.

4.2.12 Degenerate_toroidal_surface

A Degenerate_toroidal_surface is a type of Toroidal_surface (see 4.2.50) in which the minor_radius is greater than the major_radius. The resulting surface is degenerate and self-intersecting. The attribute select_outer specifies which portion of this surface is selected to produce a well defined surface.

The data associated with a Degenerate_toroidal_surface are the following:

— select_outer.

4.2.12.1 select_outer

The select_outer specifies whether or not the outer portion of the degenerate surface is selected. When select_outer is true an apple shaped surface is defined; if false the Degenerate_toroidal_surface is lemon shaped.

4.2.13 Direction

A Direction is a type of Geometric_element (see 4.2.20), it is a unit vector in 3 dimensional space.

4.2.14 Edge

An Edge is a type of Topological_element (see 4.2.49) that is the bounded intersection curve of two surfaces. It is defined by the curve geometry and two vertices.

4.2.15 Elementary_B-rep

The Elementary_B-rep is a type of B-rep (see 4.2.4) which consists of only elementary geometric elements, all Faces (see 4.2.18) and Edges (see 4.2.14) being associated with elementary geometry.

4.2.16 Elementary_surface

An Elementary_surface is a type of Surface (see 4.2.43) which can be described by simple analytic equations. Each Elementary_surface is either a Conical_surface (see 4.2.8), a Cylindrical_surface (see 4.2.11), a Plane (see 4.2.31), a Spherical_surface (see 4.2.42), or a Toroidal_surface (see 4.2.50).

NOTE see figure G.8 in the NIAM diagrams

4.2.17 Ellipse

An Ellipse is a type of Conic (see 4.2.7) generated by intersecting a conical surface with a plane whose normal is at a small angle to the axis of the conical surface.

The data associated with an Ellipse are the following:

— location;

— major_radius;

— minor_radius.

4.2.17.1 location

The location specifies the position and orientation of the Ellipse. It is defined by the centre point, the normal direction to its plane, and the direction of one of the principal axes of the Ellipse.

4.2.17.2 major_radius

The major_radius specifies the half length of the longer principal axis of the Ellipse. It is equal to the maximum distance of any point on the ellipse from the centre.

4.2.17.3 minor_radius

The minor_radius specifies the half length of the shorter principal axis of the Ellipse. It is equal to the minimum distance of any point on the Ellipse from the centre.

4.2.18 Face

A Face is a type of Topological_element (see 4.2.49) that is a bounded portion of a Surface (see 4.2.43). It consists of the surface geometry, at least one surrounding loop, and possible inner loops, which can be regarded as holes in the surface. If the Face is used in a Faceted_B-rep (see 4.2.19), the loops are simplified and defined as Poly_loops (see 4.2.35).

The data associated with a Face are the following:

— face_geometry;

— bounds.

4.2.18.1 face_geometry

The face_geometry specifies the Surface (see 4.2.43) of which the Face is a bounded portion.

4.2.18.2 bounds

The bounds are the loops which define the boundaries of the Face. At most one of these may be an outer bound.

4.2.19 Faceted_B-rep

A Faceted_B-rep is a type of B-rep (see 4.2.4) with only planar surfaces and straight lines as geometric elements. The topology is simplified by using Poly_loops (see 4.2.35) which list all points defining the corners of the faces of the solid.

4.2.20 Geometric_element

A Geometric_element is part of the geometric description of a B-rep. Each Geometric_element is either a Point (see 4.2.32), a Curve (see 4.2.9), a Direction (see 4.2.13), a Location (see 4.2.25), or a Surface (see 4.2.43).

4.2.21 Global_unit

A Global_unit is a unit that is valid for all length measures or for all angle measures within the context of a shape model.

EXAMPLE 1 millimetre could be a global unit for length measures.

4.2.22 Hyperbola

An Hyperbola is a type of Conic (see 4.2.7) that is an open conic section defined by its radius, imaginary radius, the centre point, and the normal direction to its plane.

NOTE An Hyperbola may be generated by intersecting a conical surface with a plane whose normal is at a large angle to the axis of the conical surface.

4.2.23 Light_source

A Light_source is a type of Presentation_appearance (see 4.2.36) that specifies the way in which the components of a B-rep model are illuminated for viewing. There may be more than one Light_source associated with a particular view. The data associated with a Light_source are the following:

— colour;

— position;

— direction.

4.2.23.1 colour

The colour specifies the red, green and blue intensity values of the Light_source.

4.2.23.2 position

The position specifies the location of the Light_source. The position need not be specified for some types of Light_source.

4.2.23.3 direction

The direction specifies the directional properties of the Light_source. The direction need not be specified for some types of Light_source.

4.2.24 Line

A Line is a type of Unbounded_curve (see 4.2.53) which is defined by a point and a direction. The positive direction of the Line is that of the direction vector.

4.2.25 Location

A Location is a type of Geometric_element (see 4.2.20), it is the placement, or position and orientation, of a geometric element in the coordinate space. It is defined by a Point and 2 Directions (see 4.2.32 and 4.2.13).

4.2.26 Loop

A Loop is a type of Topological_element (see 4.2.49) that defines the boundary of a surface. The Loop has to be closed, and self-intersection is not allowed. A Loop consists of an ordered collection of at least one Edge (see 4.2.14) (in the case of edge_loop), or of a single Vertex (see 4.2.54) (in the case of vertex_loop), or of an ordered collection of points (in the case of a Poly_loop) (see 4.2.35).

4.2.27 Name

A Name is a user-defined identifier for an object. Any entity of this part of ISO 10303 that may be part of a B-rep model can be assigned a Name. Presentation_appearance (see 4.2.36) entities shall not have Names.

4.2.28 Parabola

A Parabola is a type of Conic (see 4.2.7) generated by intersecting a conical surface with a plane whose normal is parallel to a generating line of the conical surface. A parabola is defined by its focal length, (i.e., the distance between focal point and vertex point), vertex point, and the direction of the normal to its plane.

4.2.29 Part

A Part is a mechanical component which can be represented by a B-rep solid model.

The data associated with a Part are the following:

— user_defined_name.

The user_defined_name specifies one or more words chosen by the user to identify and describe the functionality of the Part.

4.2.30 Pcurve

A Pcurve is a type of Curve (see 4.2.9), and is a 3D curve which is defined in the 2D parametric space of a Surface. The data associated with a Pcurve are:

— basis_surface;

— curve_2d.

4.2.30.1 basis_surface

The basis_surface specifies the Surface (see 4.2.43) which provides the parameter space for the definition of the curve_2d. The Pcurve itself lies on this Surface.

4.2.30.2 curve_2d

The curve_2d specifies the geometry of the Pcurve as a 2D curve definition which uses the parameter space of the basis_surface as its coordinate system.

4.2.31 Plane

A Plane is a type of Elementary_surface (see 4.2.16) which is unbounded and has a constant normal.

The data associated with a Plane are the following:

— location.

4.2.31.1 location

The location specifies the position and orientation of the Plane, it is defined by a point on the surface and the direction of the normal to the Plane.

4.2.32 Point

A Point is a type of Geometric_element (see 4.2.20), it is a precise location in real space. A Point is defined by its Cartesian coordinates in 3-dimensional space.

4.2.33 Point_appearance

A Point_appearance is a type of Presentation_appearance (see 4.2.36) that specifies the visual appearance of a Point (see 4.2.32). The data associated with a Point_appearance are the following:

— colour;

— marker;

— marker_size.

4.2.33.1 colour

The colour specifies the colour of the Point when viewed with a colour display. It is specified by intensity values for red, green, and blue. The colour need not be specified for a particular Point, in which case the default option is black.

4.2.33.2 marker

The marker specifies the form of the symbol used to display a Point. The marker need not be specified for a particular Point, in which case the default option is an addition sign $(+)$.

4.2.33.3 marker_size

The marker_size specifies the size of the selected marker type. The marker_size need not be specified for a particular Point, in which case the default value for this size is 1 mm.

4.2.34 Polyline

A Polyline is a type of Bounded_curve (see 4.2.5) which consists of n-1 linear segments. It is defined by a list of n points.

4.2.35 Poly_loop

A Poly_loop is a type of Loop (see 4.2.26) used in the Faceted_B-rep (see 4.2.19). It is represented by an ordered coplanar collection of points forming the vertices of the loop. The loop is composed of straight line segments each joining a point in the collection to the succeeding point in the collection. The closing segment is from the last to the first point in the collection. The direction of the loop is in the direction of the line segments.

4.2.36 Presentation_appearance

A Presentation_appearance is a specification of the visual appearance which is relevant for the visualization of geometric models. The visual properties which may be specified include curve-style and

curve-width for curves; colour for B-rep models, surfaces, curves, points. Presentation_appearance is an abstract concept which is not itself instantiated but may be assigned to geometric elements, or to topological elements. Each Presentation_appearance is either a 3D_projection (see 4.2.1), a Curve_appearance (see 4.2.10), a Light_source (see 4.2.23), a Point_appearance (see 4.2.33), or a Surface_appearance (see 4.2.44).

4.2.37 Product

A Product is a physical manufactured object. In the context of this part of ISO 10303 the shape of a Product is represented by one or more B-rep models (see 4.2.4) corresponding to the constituent parts of the Product. These may be collected together as an Assembly (see 4.2.3).

The data associated with a Product are the following:

— coordinate_system;

— user_defined_name;

— version_and_id.

4.2.37.1 coordinate_system

The coordinate_system specifies the Cartesian coordinate system used to define the precise geometry of the Product.

4.2.37.2 user_defined_name

The user_defined_name specifies the name selected by the user for reference purposes. It consists of one or more words and may describe the product functionality.

4.2.37.3 version_and_id

The version_and_id specifies terms including version number and identifier which uniquely identify an instance of a Product.

4.2.38 Sculptured_surface

A Sculptured_surface is a type of Surface (see 4.2.43) which is a general bi-parametric surface of polynomial or rational form.

4.2.39 Screen_image

A Screen_image is a collection of 2-dimensional images defined by Presentation_appearance instances which is intended to be displayed simultaneously. There is a maximum of one Screen_image present in any instance of a model which conforms to this part of ISO 10303.

4.2.40 Shape_representation

A Shape_representation is a representation of the precise size and shape of a mechanical product or component. A Shape_representation is made up of one or more B-rep models (see 4.2.4).

4.2.41 Shell

A Shell is a type of Topological_element (see 4.2.49). In the context of this part of ISO 10303 a Shell is always closed. A Shell is a collection of one or more Faces (see 4.2.18) which bounds a region in the 3-dimensional space. The topological normal of the Shell is defined as being directed from the finite to the infinite region. A Void (see 4.2.55) in a B-rep is represented by an interior shell which builds a "hole" inside an outer shell. For an interior shell the topological normal will point into the solid material.

NOTE see figure G.6 in the NIAM diagrams.

4.2.42 Spherical_surface

A Spherical_surface is a type of Elementary_surface (see 4.2.16) that is ball-shaped. It is defined by its radius and its centre point.

The data associated with a Spherical_surface are:

— radius;

— centre.

4.2.42.1 radius

The radius specifies the distance from any point on the Spherical_surface to the centre of the sphere.

4.2.42.2 centre

The centre specifies the fixed point which is equidistant from all points on the Spherical_surface.

4.2.43 Surface

A Surface is a type of Geometric_element (see 4.2.20). A Surface can be regarded as being generated by a continuously changing curve moving in space. Each Surface is either an Elementary_surface (see 4.2.16), a Swept_surface (see 4.2.48), or a Sculptured_surface (see 4.2.38). The extent of a Surface may be infinite.

NOTE see figure G.7 in the NIAM diagrams.

4.2.44 Surface_appearance

A Surface_appearance is a type of Presentation_appearance (see 4.2.36) that provides the specification of the visual appearance of a Surface (see 4.2.43) when displayed. The data associated with a Surface_appearance are the following:

— colour;

— grid_indicator;

— shading_method.

4.2.44.1 colour

The colour specifies the colour of the Surface when viewed with a colour display. It is specified by intensity values for red, green, and blue. The colour need not be specified for a particular Surface, in which case the default option is black.

4.2.44.2 grid_indicator

The grid_indicator specifies the way in which a surface is to be displayed. This includes the selection of the curves which are used to display the Surface. The value of the grid_indicator may be one or more of the following options:

— boundary curves;

— silhouette curves;

— segmentation curves;

— control grid;

— parameter lines;

— shading.

The grid_indicator need not be specified for a particular Surface, in which case the default option is silhouette curves.

4.2.44.3 shading_method

The shading_method specifies the algorithm used for the shading of a surface. The shading_method need not be specified for a particular Surface, in which case the default option is normal shading. Normal shading is based upon interpolated values of the surface normals.

4.2.45 Surface_curve

A Surface_curve is a type of Curve (see 4.2.9) which lies on a surface. It is defined in three dimensional space and in the parametric space of the surface. The data associated with a Surface_curve are the following:

— curve_3d;

— associated_geometry.

4.2.45.1 curve_3d

The curve_3d specifies a representation of the Surface_curve as a curve in the same three dimensional coordinate system as the surface.

4.2.45.2 associated_geometry

The associated_geometry specifies a set of one or two pcurves which provide two dimensional representations of the surface_curve in the parameter space of the surface, or surfaces, on which it lies. The surfaces are derived from this associated_geometry.

4.2.46 Surface_of_extrusion

A Surface_of_extrusion is a type of Swept_surface (see 4.2.48) constructed by sweeping a Curve (see 4.2.9) in a given fixed direction.

The data associated with a Surface_of_extrusion are the following:

— swept_curve;

— extrusion_direction.

4.2.46.1 swept_curve

The swept_curve specifies the curve which is swept in order to generate the Surface_of_extrusion. Any cross section of the surface perpendicular to the extrusion_direction has geometry identical to this curve.

4.2.46.2 extrusion_direction

The extruded_direction specifies the direction in which the swept_curve is extruded to generate the surface.

4.2.47 Surface_of_revolution

A Surface_of_revolution is a type of Swept_surface (see 4.2.48) constructed by rotating a Curve (see 4.2.9) a complete revolution about an axis.

The data associated with a Surface_of_revolution are the following:

— revolved_curve;

— axis.

4.2.47.1 revolved_curve

The revolved_curve specifies the curve which is rotated about the axis in order to generate the Surface_of_revolution.

4.2.47.2 axis

The axis specifies the axis about which the revolved_curve is revolved to generate the surface. It is the central axis of the surface which is generated.

4.2.48 Swept_surface

A Swept_surface is a type of Surface (see 4.2.43) constructed by sweeping a curve along another curve. Each Swept_surface is either a Surface_of_extrusion (see 4.2.46) or a Surface_of_revolution (see 4.2.47).

4.2.49 Topological_element

A Topological_element is part of the topological definition of a B-rep (see 4.2.4). Each Topological_element is either an Edge (see 4.2.14), a Face (see 4.2.18), a Loop (see 4.2.26), a Shell (see 4.2.41) or a Vertex (see 4.2.54).

4.2.50 Toroidal_surface

A Toroidal_surface is a type of Elementary_surface (see 4.2.16) constructed by rotating a circle around an axis lying outside the circle (or inside in the special case of a Degenerate_toroidal_surface), and in the plane of the circle. It is defined by a centre point, an axis, and two radii.

Each Toroidal_surface may be a Degenerate_toroidal_surface. (see 4.2.12)

The data associated with a Toroidal_surface are the following:

— centre_point;

— axis;

— major_radius;

— minor_radius.

4.2.50.1 centre_point

The centre_point specifies the point at the centre of the Toroidal_surface.

4.2.50.2 axis

The axis specifies the direction in which the swept_curve is extruded to generate the surface.

4.2.50.3 major_radius

The major_radius specifies the distance from the centre of the generating circle of the Toroidal_surface to the axis.

4.2.50.4 minor_radius

The minor_radius specifies the radius of the circle which is rotated about the axis to generate the Toroidal_surface.

Except for the special Degenerate_toroidal_surface type the minor_radius shall be less than the major_radius.

4.2.51 Transformation

A Transformation is a geometric operation composed of translation, rotation, mirroring, and uniform scaling. Transformations may be used to locate the components of an assembly.

4.2.52 Twisted_curve

A Twisted_curve is a type of Bounded_curve (see 4.2.5), in the context of this part of ISO 10303, is a general parametric 3-dimensional curve which is represented by a B-spline curve.

4.2.53 Unbounded_curve

An Unbounded_curve is a type of Curve (see 4.2.9) with infinite extent. An Unbounded_curve has a simple analytic equation to describe it. Each Unbounded_curve is either a Line (see 4.2.24) or a Conic (see 4.2.7).

4.2.54 Vertex

A Vertex is a type of Topological_element (see 4.2.49) that is a topological point which forms part of the boundary of an Edge (see 4.2.14), or of a Face (see 4.2.18).

4.2.55 Void

A Void is a hollow region inside a solid model. A void is represented by a Shell with a normal pointing into the empty space inside. This normal opposes the sense of the topological normal to the defining shell.

4.3 Application assertions

This clause specifies the application assertions for the mechanical design using boundary representation application protocol. The application assertions define the required relationships between the application objects defined in the previous clause, the cardinality of the relationships and any integrity rules required.

4.3.1 3D_projection to Screen_image

A 3D_projection is in one Screen_image. A Screen_image contains one or more 3D_projections.

4.3.2 3D_projection to B-rep

A 3D_projection presents zero, one or more B-rep models. A B-rep model is associated with zero, one, or more 3D_projections.

4.3.3 Assembly to Assembly

An Assembly contains zero, one or more Assemblies, in the role of sub-assemblies with associated transformation. An Assembly is part of zero, one or many Assemblies.

4.3.4 Assembly to Product

An Assembly is part of one or more Products. A Product contains zero, one, or more Assemblies.

4.3.5 B-rep to Shell

A B-rep consists of one or more Shells. Where two or more closed Shells are used to define a B-rep one of these shall be the outer shell and contain all the others. Each Shell is used to define one B-rep.

4.3.6 Curve to Edge

Each Curve in the exchange structure is used to define the geometry of zero, one, or more Edges. Each Edge has its geometry defined by one Curve.

4.3.7 Curve_appearance to Curve

A curve_appearance is associated with zero, one, or many Curves. A Curve is represented as zero or one Curve_appearances.

4.3.8 Curve_appearance to Edge

A curve_appearance is associated with zero, one, or many Edges. An Edge is represented as zero or one curve_appearances.

4.3.9 Edge to B-rep

Each Edge referenced by a B-rep model is used precisely twice with opposing orientations in the definition of that B-rep model. A B-rep has zero, one or many explicit Edges.

4.3.10 Edge to Edge

An Edge shall not intersect any other Edge except at a shared Vertex.

4.3.11 Edge to Vertex

An Edge is bounded by precisely two Vertices. A Vertex is used to bound one or more Edges.

4.3.12 Face to Face

A Face shall not intersect any other Face except along a common edge.

4.3.13 Face to Loop

A Face is bounded by one or more Loops. A Loop bounds one or more Faces.

4.3.14 Face to Surface

A Face shall have its geometry defined by precisely one Surface. In a faceted B-rep model the geometry of each Face shall be defined by precisely one Plane. Each Surface in the exchange structure shall be used to define the geometry of one or more Faces.

4.3.15 Loop to Edge

A Loop of type edge_loop shall be defined by one or more Edges. Each Edge shall be used in the definition of one or two Loops in a B-rep model.

4.3.16 Name to Geometric_element

A Geometric_element has zero or one user defined Name. A Name is associated with zero or one Geometric_elements.

4.3.17 Name to Topological_element

A Topological_element has zero or one user defined Name. A Name is associated with zero or one Topological_elements.

4.3.18 Part to Assembly

A Part may belong to one or more Assemblies. An Assembly contains one or more Parts. Each Part in the Assembly is located by one Transformation. An Assembly may not exist without a Part.

4.3.19 Part to Product

A Part is a direct component of zero, one or many Products; it may, also, be indirectly included in a product as part of an assembly. A Product consists of zero, one, or many Parts.

4.3.20 Part to Shape_representation

A Part shall have its shape defined by one or more B-rep models in a Shape_representation.

4.3.21 Point_appearance to Point

A Point_appearance may be associated with zero, one, or many Points. A Point is represented as zero or one Point_appearances.

4.3.22 Presentation_appearance to B-rep

A Presentation_appearance presents zero, one, or many B-reps. A B-rep model is presented by zero, one or many Presentation_appearances.

4.3.23 Presentation_appearance to Shell

A Presentation_appearance presents zero, one, or many Shells. A Shell is presented by zero, one or many Presentation_appearances.

4.3.24 Shape_representation to B-rep

Each B-rep model in the exchange structure shall be part of precisely one Shape_representation corresponding to one of the three levels of functionality defined in this document. The simplest possible form of representation compatible with the information requirements shall be used.

4.3.25 Shell to Face

A Shell is defined by one or more Faces. A Face is used to define one or more Shells. Within a given B-rep model a Face is associated with precisely one Shell.

4.3.26 Surface_appearance to Face

A Surface_appearance presents zero, one, or many Faces. A Face is presented by zero or one Surface_appearances.

4.3.27 Surface_appearance to Surface

A Surface_appearance presents zero, one, or many Surfaces. A Surface is presented by zero or one Surface_appearances.

4.3.28 Topological_element to Presentation_appearance

A Topological_element is associated with zero, one, or more, presentation_appearances to define its display properties. A Presentation_appearance presents zero, one or more Topological_elements.

4.3.29 Transformation to Assembly

A Transformation is used to locate zero or one Assembly, as a sub-assembly, in an Assembly. When an assembly has the role of a sub-assembly it shall be located by a Transformation. An Assembly may be located by zero or one transformations.

4.3.30 Transformation to Part

A Transformation may be used to locate zero or one Parts in an Assembly. Each Part in an Assembly is located by one Transformation.

4.3.31 Vertex to Point

Each Vertex has its geometry defined by precisely one Point. A Point is used to define the geometry of zero or one Vertex.

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. The contents of these five columns are:

— 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 application interpreted construct (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 ARM element, 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 is specified.

For the expression of reference paths the following notational conventions apply:

a) `[]` : multiple AIM elements or sections of the reference path are required to satisfy an information requirement;

b) `()` : multiple AIM elements or sections of the reference path are identified as alternatives within the mapping to satisfy an information requirement;

c) `{}` : enclosed section contrains the reference path to satisfy an information requirement;

d) → : attribute references the entity or select type given in the following row;

e) ←: entity or select type is referenced by the attribute in the following row;

f) `[i]` : attribute is an aggregation of which a single member is given in the following row;

g) `[n]` : attribute is an aggregation of which member `n` is given in the following row;

h) ⇒: entity is a supertype of the entity given in the following row;

i) ⇐: entity is a subtype of the entity given in the following row;

j) = : the string, select, or enumeration type is constrained to a choice or value.

k) `/` : the line continuations for strings that wrap.

This Part of ISO 10303 makes use of the following application interpreted constructs (AIC):

— part 511 = aic_topologically_bounded_surface : topologically bounded surface AIC;

— part 512 = aic_faceted_brep : faceted B-rep AIC;

— part 513 = aic_elementary_brep : elementary B-rep AIC;

— part 514 = aic_advanced_brep : advanced B-rep AIC (uses AIC1);

— part 517 = aic_mechanical_design_geometric_presentation: AIC for simple visual presentation of geometric models used in mechanical design;

— part 518 = aic_mechanical_design_shaded_representation : mechanical design shaded presentation AIC.

NOTE 1 The geometric AIC parts 511 and 514 are interdependent and their relationships via ‘USE FROM’ statements are illusrated in figure 3.

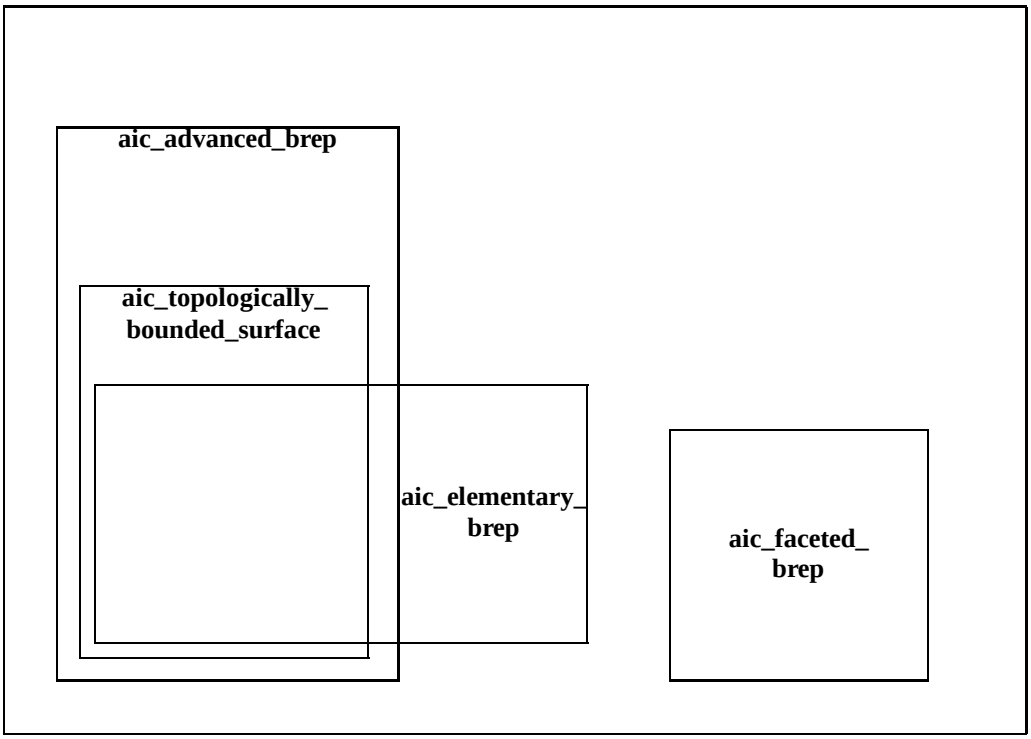


Figure 3 – Relationships between geometric AICs

Table 2 – Mapping table for advanced_B-rep UoF

Application element	AIM element	Source	Rules	Reference Path
ADVANCED_B-REP	advanced_brep_shape_representation	514 41	1	advanced_brep_shape_representation ⇐ shape_representation
B-REP	manifold_solid_brep OR brep_with_voids	514 514	2,9,12	
B-rep to shell				(manifold_solid_brep.outer → closed_shell) (brep_with_voids.voids[i] → oriented_closed_shell ⇐ closed_shell)
BOUNDED_CURVE	bounded_curve	511	9,12	
CIRCLE	circle	511	2,9,12	
radius	circle.radius			
location	axis2_placement_3d			circle ⇐ conic conic.position → axis2_placement = axis2_placement_3d
CONIC	conic	511	2,9,12	
CONICAL_SURFACE	conical_surface	513	2,9,12	
CURVE	curve	511	9,12	
curve to edge				curve ← edge_curve.edge_geometry edge_curve

Table 2 – Mapping table for advanced_B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
CYLINDRICAL_SURFACE	cylindrical_surface	513	2,9,12	
radius	cylindrical_surface.radius			
axis	axis2_placement_3d.axis			cylindrical_surface ⇐ elementary_surface elementary_surface.position → axis2_placement_3d axis2_placement_3d.axis
DEGENERATE_TOROIDAL_SURFACE	degenerate_toroidal_surface	511		
select_outer	degenerate_toroidal_surface.select_outer			
DIRECTION	direction	511	2,9,12	
EDGE	edge_curve	511 42	16,20,24	edge_curve ⇐ edge
edge to B-rep				(manifold_solid_brep.outer → closed_shell) (brep_with_voids.voids[i] → oriented_closed_shell ⇐ closed_shell) closed_shell.cfs_faces[i] → face face.bounds[i] → face_bound face_bound.bound → loop ⇒ edge_loop ⇐ path path.edge_list[i] → oriented_edge oriented_edge.edge_element → edge

Table 2 – Mapping table for advanced_B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
edge to curve				edge_curve.edge_geometry → curve
edge to edge				edge_curve ⇐ edge (edge.edge_start) (edge.edge_end) → vertex ⇒ { vertex_point = vertex_point } ⇐ vertex ← (edge.edge_start) (edge.edge_end) edge
edge to vertex				(edge.edge_start)(edge.edge_end) → vertex ⇒ vertex_point
ELEMENTARY_SURFACE	elementary_surface	511	9,12	
ELLIPSE	ellipse	511	2,9,12	
major_radius	ellipse.semi_axis_1 OR ellipse.semi_axis_2			
minor_radius	ellipse.semi_axis_2 OR ellipse.semi_axis_1			
location	axis2_placement_3d			ellipse ⇐ conic conic.position → axis2_placement = axis2_placement_3d
FACE	face_surface	511	2,9,12	
face_geometry	face_surface.face_geometry			
bounds	face.bounds			face_surface ⇐ face face.bounds

Table 2 – Mapping table for advanced_B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
face to face				face.bounds[i] → face_bound face_bound.bound → loop ⇒ edge_loop ⇐ path path.edge_list[i] → oriented_edge oriented_edge.edge_element → {edge = edge} ← oriented_edge.edge_element oriented_edge ← path.edge_list[i] path ⇒ edge_loop ⇐ loop ← face_bound.bound face_bound ← face.bounds[i]
face to loop				face_surface ⇐ face face.bounds[i] → face_bound face_bound.bound → loop ⇒
face to surface				face_surface face_surface.face_geometry → surface
GLOBAL_UNIT	global_unit_assigned_context.units	41	17,18,19	
HYPERBOLA	hyperbola	511	2,9,12	
LINE	line	511	2,9,12	

Table 2 – Mapping table for advanced_B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
LOCATION	axis2_placement_3d	511	2	
LOOP	loop	511	16	
loop to edge				edge_loop ⇐ path path.edge_list[i] → oriented_edge ⇐ edge
PARABOLA	parabola	511	2,9,12	
PCURVE	pcurve	511		
basis_surface	pcurve.basis_surface			
curve_2d	representation.items[1]		24, 25	pcurve pcurve.reference_to_curve → definitional_representation ⇐ representation representation.items[1]
PLANE	plane	511	2,9,12	
location	elementary_surface.position			plane ⇐ elementary_surface elementary_surface.position
POINT	cartesian_point	511	2,9,12	
POLYLINE	polyline	511	2,9,12	
SCULPTURED_SURFACE	b_spline_surface	511	2,9,12	
SHELL	closed_shell	514	16	
shell to face				closed_shell ⇐ connected_face_set connected_face_set.cfs_faces → face
SPHERICAL_SURFACE	spherical_surface	511	2,9,12	
radius	spherical_surface.radius			

Table 2 – Mapping table for advanced_B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
centre	placement.location			spherical_surface ⇐ elementary_surface elementary_surface.position → axis2_placement_3d ⇐ placement placement.location
SURFACE	surface	511		
SURFACE_CURVE	surface_curve	511		
curve_3d	surface_curve.curve_3d			
associated_geometry	surface_curve.associated_geometry			
SURFACE_OF_EXTRUSION	surface_of_linear_extrusion	511	2,9,12	
extrusion_direction	surface_of_linear_extrusion. extrusion_axis			
swept_curve	swept_surface.swept_curve			surface_of_linear_extrusion ⇐ swept_surface swept_surface.swept_curve
SURFACE_OF_REVOLUTION	surface_of_revolution	511	2,9,12	
axis	surface_of_revolutionion. axis_position			
revolved_curve	swept_surface.swept_curve			surface_of_revolution ⇐ swept_surface swept_surface.swept_curve
SWEPT_SURFACE	swept_surface	511	9	
TOROIDAL_SURFACE	toroidal_surface	511	2,9,12	
major_radius	toroidal_surface.major_radius			
minor_radius	toroidal_surface.minor_radius			

Table 2 – Mapping table for advanced_B-rep UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
centre_point	placement.location			toroidal_surface ⇐ elementary_surface elementary_surface.position → axis2_placement_3d ⇐ placement placement.location
axis	axis2_placemnt_3d.axis			toroidal_surface ⇐ elementary_surface elementary_surface.position → axis2_placement_3d axis2_placement_3d.axis
TWISTED_CURVE	b_spline_curve	511	2,9,12	
UNBOUNDED_CURVE	conic OR line	511 511		
VERTEX	vertex_point	511	2,16,24	vertex_point
vertex to point				vertex_point.vertex_geometry → cartesian_point
VOID	brep_with_voids.voids[i]	514		

Table 3 – Mapping table for elementary_B-Rep UoF

Application element	AIM element	Source	Rules	Reference Path
B-REP	manifold_solid_brep OR brep_with_voids	513 513	2,20,23,24	
B-rep to shell				(manifold_solid_brep.outer → closed_shell) (brep_with_voids.voids[i] → oriented_closed_shell ⇐ closed_shell)
BOUNDED_CURVE	polyline	513	2,20,24	
CIRCLE	circle	513	2,20,24	
CONIC	conic	513	2,20,24	
CONICAL_SURFACE	conical_surface	513	2,20,24	
CURVE	curve	513	9,12	
curve to edge				curve ← edge_curve.edge_geometry edge_curve
CYLINDRICAL_SURFACE	cylindrical_surface	513	2,20,24	
DIRECTION	direction	513	2,20,24	
EDGE	edge_curve	513 42	16,20,24	edge_curve ⇐ edge

Table 3 – Mapping table for elementary_B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
edge to B-rep				(manifold_solid_brep.outer → closed_shell) (brep_with_voids.voids[i] → oriented_closed_shell ⇐ closed_shell) closed_shell.cfs_faces[i] → face face.bounds[i] → face_bound face_bound.bound → loop ⇒ edge_loop ⇐ path path.edge_list[i] → oriented_edge oriented_edge.edge_element → edge
edge to curve				edge_curve.edge_geometry → curve
edge to edge				edge_curve ⇐ edge (edge.edge_start) (edge.edge_end) → {vertex_point = vertex_point } ← (edge.edge_start) (edge.edge_end) edge
edge to vertex				(edge.edge_start)(edge.edge_end) → vertex_point
ELEMENTARY_B-REP	elementary_brep_shape_representation	513 41	1 14	elementary_brep_shape_representation ⇐ shape_representation
ELEMENTARY_SURFACE	elementary_surface	513	9	
ELLIPSE	ellipse	513	2,20,24	

Table 3 – Mapping table for elementary_B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
FACE	face_surface	513	16,20,24	
face to face				face.bounds[i] → face_bound
				face_bound.bound → loop ⇒ edge_loop ⇐ path path.edge_list[i] → oriented_edge oriented_edge.edge_element → {edge = edge} ← oriented_edge.edge_element oriented_edge ← path.edge_list[i] path ⇒ edge_loop ⇐ loop ← face_bound.bound face_bound ← face.bounds[i]
face to loop				face_surface ⇐ face face.bounds[i] → face_bound face_bound.bound → loop ⇒
face to surface				face_surface face_surface.face_geometry → surface ⇒ elementary_surface

Table 3 – Mapping table for elementary_B-rep UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
GLOBAL_UNIT	global_unit_assigned_context.units	41	17,18,19	
HYPERBOLA	hyperbola	513	2,20,24	
LINE	line	513	2,20,24	
LOCATION	axis2_placement_3d	513	2,9	
LOOP	loop	513	16	
loop to edge				edge_loop ⇐ path path.edge_list[i] → oriented_edge ⇐ edge
PARABOLA	parabola	513	2,20,24	
PLANE	plane	513	2,20,24	
POINT	cartesian_point	513	2,20,24	
POLYLINE	polyline	513	2,20,24	
SHELL	closed_shell	513	16	
shell to face				closed_shell ⇐ connected_face_set connected_face_set.cfs_faces → face
SPHERICAL_SURFACE	spherical_surface	513	2,20,24	
SURFACE	elementary_surface	513	9,12	
TOROIDAL_SURFACE	toroidal_surface	513	2,20,24	
UNBOUNDED_CURVE	conic OR line	511 511		
VERTEX	vertex_point	513	16,20,24	vertex_point
vertex to point				vertex_point.vertex_geometry → cartesian_point
VOID	brep_with_voids.voids[i]	513		

Table 4 – Mapping table for faceted_B-Rep UoF

Application element	AIM element	Source	Rules	Reference Path
B-REP	faceted_brep OR (faceted_brep AND brep_with_voids)	512 512	2,20,23,24	
B-rep to shell	manifold_solid_brep.outer OR brep_with_voids.voids[i]	512 512		faceted_brep ⇐ manifold_solid_brep manifold_solid_brep.outer → closed_shell faceted_brep AND brep_with_voids brep_with_voids.voids[i] → closed_shell
DIRECTION	direction	512	2,20,24	
FACE	face_surface	512	2,20,24	
face to loop	face.bounds[i]	512		face_surface ⇐ face face.bounds → loop
face to surface	face_surface.face_geometry	512		face_surface face_surface.face_geometry → surface ⇒ elementary_surface ⇒ plane
FACETED_B-REP	faceted_brep_shape_representation	512	1	
GLOBAL_UNIT	global_unit_assigned_context.units	41	17,18,19	
LOOP	poly_loop	512	16,20,24	poly_loop

Table 4 – Mapping table for faceted_B-rep UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
PLANE	plane	512	2,20,24	
POINT	cartesian_point	512	2,20,24	
POLY_LOOP	poly_loop	512	16,20,24	
SHELL	closed_shell	512	16	
SURFACE	plane	512	2,20,24	plane
VOID	brep_with_voids.voids[i]	512		

Table 5 – Mapping table for name_preservation UoF

Application element	AIM element	Source	Rules	Reference Path
NAME	representation.name OR representation_item.name	43 43		
name to geometric_element				representation_item.name representation_item ⇒ geometric_representation_item
name to topological_element				representation_item.name representation_item ⇒ topological_representation_item

Table 6 – Mapping table for product_structure UoF

Application element	AIM element	Source	Rules	Reference Path
ASSEMBLY	product_definition_relationship. relating_product_definition	44	13	{assembly_component_usage ⇐ product_definition_relationship } product_definition_relationship.relating_product_definition
coordinate_system	shape_representation.context_ of_items	43		
user_defined_name	product.name	41	10, 11 12	product_definition_relationship.relating_product_definition → product_definition product_definition.formation → product_definition_formation product_definition_formation.of_product → product product.name
assembly to assembly				product_definition_relationship.relating_product_definition {product_definition_relationship ⇒ product_definition_usage ⇒ assembly_component_usage } product_definition_relationship.relating_product_definition

Table 6 – Mapping table for product_structure UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
assembly to coordinate_system				product_definition_relationship.relating_ / product_definition → product_definition ← product_definition_shape.definition product_definition_shape = shape_definition ← shape_definition_representation.representation_of shape_definition_representation shape_definition_representation.representation_ / model → shape_representation ⇐ representation representation.context_of_items
assembly to product				product_definition_relationship.relating_ / product_definition product_definition → product_definition.formation
PART	product_definition_shape	41		
user_defined_name	product_definition_shape.name	41		
part to assembly				product_definition_shape product_definition_shape.definition → characterised_product_definition = product_definition ← product_definition_relationship.related_product_ / definition product_definition_relationship product_definition_relationship.relating_product_ / definition
part to product				product_definition_shape product_definition_shape.definition → characterised_product_definition = product_definition product_definition.formation ← product_definition_formation

Table 6 – Mapping table for product_structure UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
part to shape_representation				product_definition_shape = shape_definition ← shape_definition_representation. / representation_of shape_definition_representation
PRODUCT	product_definition_formation	41	21,22	
coordinate_system	shape_representation.context_of_items	43		
user_defined_name	product.name	41	9	product_definition_formation.of_product → product product.name
version_and_id	product_definition_formation.id AND product.id	41 41		
SHAPE_REPRESENTATION	shape_representation	41	1	
global_units	global_unit_assigned_context.units	41	3	shape_representation ⇐ representation representation.context_of_items → representation_context ⇒ global_unit_assigned_context global_unit_assigned_context.units
shape_representation to brep	elementary_brep_shape_rep. OR advanced_brep_shape_rep. OR faceted_brep_shape_rep.	513 514 512	1 1 1	elementary_brep_shape_rep. ⇒ shape_representation advanced_brep_shape_rep. ⇒ shape_representation faceted_brep_shape_rep. ⇒ shape_representation

Table 6 – Mapping table for product_structure UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
TRANSFORMATION	mapped_item	43	3	
scaling_factor	cartesian_transformation_operator.scale	514		mapped_item.mapping_target → representation_item ⇒ geometric_representation_item ⇒ cartesian_transformation_operator cartesian_transformation_operator.scale
transformation to assembly				mapped_item ⇐ representation_item ← representation.items[i] representation ⇒ shape_representation ← shape_definition_representation.representation_of shape_definition = product_definition_shape product_definition_shape.definition → characterised_product_definition = product_definition_relationship product_definition_relationship.related_ product_definition
transformation to part				mapped_item ⇐ representation_item ← representation.items[i] representation ⇒ shape_representation ← shape_definition_representation.representation_of shape_definition = product_definition_shape

Table 7 – Mapping table for visual_presentation_for_B-rep UoF

Application element	AIM element	Source	Rules	Reference Path
3D_PROJECTION	camera_model_d3	517, 518		
3D_projection to B-rep				camera_model_d3 ⇐ camera_model_d3 ⇐ camera_model ← camera_usage.projection (DER) camera_usage ⇐ representation_map representation_map.mapped_representation → representation ⇒ shape_representation ⇒ (advanced_brep_shape_representation) (elementary_brep_shape_representation) (faceted_brep_shape_representation)
3D_projection to screen_image				camera_model_d3 ⇐ camera_model ← camera_usage.projection (DER) camera_usage ← camera_image.source (DER) camera_image ⇐ geometric_representation_item ⇐ representation_item ← representation.items[i] {representation ⇒ presentation_representation ⇒} presentation_area ⇒ (mechanical_design_geometric_presentation_area) (mechanical_design_shaded_presentation_area)

Table 7 – Mapping table for visual_presentation_for_B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
CURVE_APPEARANCE	curve_style	517	8,15	
colour	colour_rgb	517	7	curve_style curve_style.curve_colour → colour ⇒ colour_specification ⇒ colour_rgb
curve_font	curve_style_font	517	7, 15	curve_style curve_style.curve_font → curve_font_or_scaled_ curve_font_select = curve_style_font_select = curve_style_font
curve_width	curve_style.curve_width	517		curve_style curve_style.curve_width
curve_appearance to curve				curve_style = presentation_style_select ← presentation_style_assignment.styles[i] presentation_style_assignment ← styled_item.styles[i] styled_item styled_item.item → representation_item ⇒ geometric_representation_item ⇒ curve

Table 7 – Mapping table for visual_presentation_for_B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
curve_appearance to edge				curve_style = presentation_style_select ← presentation_style_assignment.styles[i] presentation_style_assignment ← styled_item.styles[i] styled_item styled_item.item → representation_item ⇒ topological_representation_item ⇒ edge
LIGHT_SOURCE	light_source	518		
colour	light_source.light_colour			
position	cartesian_point			light_source ⇐ (light_source_spot light_source_spot.position →) (light_source_positional light_source_positional.position →) cartesian_point
direction	direction			light_source ⇐ (light_source_spot light_source_spot.orientation →) (light_source_directional light_source_directional.orientation →) direction

Table 7 – Mapping table for visual_presentation_for_B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
POINT_APPEARANCE	point_style	517	5,6	
colour	colour_rgb	517		point_style point_style.marker_colour → colour ⇒ colour_specification ⇒ colour_rgb
marker	marker_type	517		point_style point_style.marker → marker_select = marker_type
marker_size	point_style.marker_size	517		point_style point_style.marker_size
point_appearance to point				point_style = presentation_style_select ← presentation_style_assignment.styles[i] presentation_style_assignment ← styled_item.styles[i] styled_item ⇒ representation_dependent_styled_item representation_dependent_styled_item.item → representation_item ⇒ geometric_representation_item ⇒ point

Table 7 – Mapping table for visual_presentation_for_B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
PRESENTATION_APPEARANCE	presentation_style_assignment. styles[i]	517	5	
presentation_appearance to B-rep				presentation_style_assignment. styles[i] presentation_style_assignment ← styled_item.styles[i] styled_item {⇒ representation_dependent_styled_item} styled_item.item → representation_item ← representation.items[i] representation ⇒ shape_representation ⇒ (advanced_brep_shape_representation) (elementary_brep_shape_representation) (faceted_brep_shape_representation)
presentation_appearance to shell				presentation_style_assignment. styles[i] presentation_style_assignment ← styled_item.styles[i] styled_item {⇒ representation_dependent_styled_item} styled_item.item → representation_item ⇒ topological_representation_item ⇒ connected_face_set

Table 7 – Mapping table for visual_presentation_for B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
visual_presentation_for_B-rep UoF continued				
SCREEN_IMAGE	(mechanical_design_geometric_presentation_area) (mechanical_design_shaded_presentation_area)	517 518		
SURFACE_APPEARANCE	surface_style_usage	517	8,15	
colour	colour_rgb	517		surface_style_usage surface_style_usage.surface_side_style_select = surface_style_rendering surface_style_rendering.surface_colour → colour ⇒ colour_specification ⇒ colour_rgb
grid_indicator	surface_style_control_grid	517		surface_style_usage surface_style_usage.surface_side_style_select = (surface_style_boundary) (surface_style_control_grid) (surface_style_parameter_line) (surface_style_segmentation_curve) (surface_style_silhouette_curve)
shading_method	surface_style_rendering.rendering_method	518		surface_style_usage surface_style_usage.surface_side_style_select = surface_style_rendering surface_style_rendering.rendering_method

Table 7 – Mapping table for visual_presentation_for_B-rep UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
surface_appearance to face				surface_style_usage ≡ presentation_style_select ← presentation_style_assignment.styles[i] presentation_style_assignment ← styled_item.styles[i] styled_item styled_item.item → representation_item ⇒ topological_representation_item ⇒ face
surface_appearance to surface				surface_style_usage ≡ presentation_style_select ← presentation_style_assignment.styles[i] presentation_style_assignment ← styled_item.styles[i] styled_item {⇒ representation_dependent_styled_item} styled_item.item → representation_item ⇒ geometric_representation_item ⇒ surface

Rules applied to table

The following rules are referenced in the preceding tables:

1. **advanced_or_elementary_or_faceted**.
2. **dependent_instantiation_of_geometry**.
3. **dependent_intantiation_of_mapped_item**.
4. **dependent_instantiation_of_mechanical_design_geometric_p_r**.
5. **dependent_instantiation_of_mechanical_design_shaded_p_r**.
6. **dependent_instantiation_of_named_unit**.
7. **dependent_instantiation_of_pre_defined_item**.
8. **dependent_instantiation_of_presentation_style**.
9. **dependent_instantiation_of_product_context**.
10. **dependent_instantiation_of_product_definition**.
11. **dependent_instantiation_of_product_definition_context**.
12. **dependent_instantiation_of_product_definition_formation**.
13. **dependent_instantiation_of_product_definition_relationship**.
14. **dependent_instantiation_of_shape_representation**.
15. **dependent_instantiation_of_styled_item**.
16. **dependent_instantiation_of_topology**.
17. **global_units_in_geometric_representation_context**.
18. **global_units_required**.
19. **illegal_complex_named_units**.
20. **no_complex_subtypes**.
21. **product_context_mechanical**.
22. **product_definition_context_design**.
23. **shape_representation_required**.
24. **three_dimensional_geometry**.
25. **pcurve WR2.**
26. **pcurve WR3.**

5.2 AIM EXPRESS short listing

This clause specifies the EXPRESS schema that uses elements from the integrated resources (and the AICs) and contains the types, entity specializations, rules, and functions that are specific to this part of ISO 10303. This clause also specifies modifications to the text for constructs that are imported from the integrated resources (and the 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 part_204_brep_product_schema;
USE FROM aic_advanced_brep;   -- ISO 10303-514
USE FROM aic_elementary_brep; -- ISO 10303-513
USE FROM aic_faceted_brep;  -- ISO 10303-512
USE FROM aic_mechanical_design_geometric_presentation; -- ISO 10303-517
USE FROM aic_mechanical_design_shaded_presentation; -- ISO 10303-518
USE FROM application_context_schema -- ISO 10303-41
     (product_context,
      product_definition_context);
USE FROM external_reference_schema(pre_defined_item); -- ISO 10303-41
USE FROM geometry_schema -- ISO 10303-42
    (geometric_representation_context,
     geometric_representation_item);
REFERENCE FROM geometry_schema(dimension_of); -- ISO 10303-42
USE FROM measure_schema  -- ISO 10303-41
     (conversion_based_unit,
      global_unit_assigned_context,
      named_unit,
      si_unit);
USE FROM presentation_appearance_schema -- ISO 10303-46
    (presentation_style_assignment);
USE FROM product_definition_schema -- ISO 10303-41
    (product_definition,
     product_definition_formation,
     product_definition_relationship);
USE FROM product_property_definition_schema -- ISO 10303-41
    (product_definition_shape);
USE FROM product_property_representation_schema -- ISO 10303-41
     (shape_definition_representation,
      shape_representation);
USE FROM product_structure_schema(assembly_component_usage);-- ISO 10303-46
USE FROM representation_schema -- ISO 10303-43
    (representation_context,
     representation_item);
REFERENCE FROM representation_schema(using_representations);-- ISO 10303-43
USE FROM topology_schema(topological_representation_item); -- ISO 10303-42
```



```
(*
```

NOTE 1 The schemas referenced above can be found in the following Parts of ISO 10303:

aic_advanced_brep	ISO 10303-514
aic_elementary_brep	ISO 10303-513
aic_faceted_brep	ISO 10303-512
aic_mechanical_design_geometric_presentation	ISO 10303-517
aic_mechanical_design_shaded_presentation	ISO 10303-518
application_context_schema	ISO 10303-41
external_reference_schema	ISO 10303-41
geometry_schema	ISO 10303-42
measure_schema	ISO 10303-41
presentation_appearance_schema	ISO 10303-46
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
representation_schema	ISO 10303-43
topology_schema	ISO 10303-42

NOTE 2 One further AIC is used indirectly in this AP, the **aic_advanced_brep** uses the **aic_topologically_-bounded_surface**.

Definitions of types and entities unique to this AP are given below.

5.2.1 Mechanical design using boundary representation entities

5.2.1.1 design_context

A **design_context** is a **product_definition_context** that defines a life cycle stage of design as the frame of reference for **product_definition** entities.

NOTE 1 A further refinement within the context opf design may be made using the **application_context_element** entity.

EXPRESS specification:

```
*)
 ENTITY design_context
  SUBTYPE OF (product_definition_context);
 WHERE
    WR1 : SELF.life_cycle_stage = 'design';
 END_ENTITY;
(*
```


Formal propositions:

WR1: The **life_cycle_stage** of the **design_context** entity shall contain the value ‘design’.

5.2.1.2 mechanical_context

A **mechanical_context** is a **product_context** that is applicable to those products that are mechanical.

NOTE 1 The use of this entity defines a viewpoint for the context of a product. It is not intended to be a classification or categorisation of a type of product. The definition of the **product_context** to be mechanical using this entity is specifying the way that the product acts within the data exchange.

EXPRESS specification:

```
*)

 ENTITY mechanical_context
   SUBTYPE OF (product_context);
   WHERE
     WR1 : SELF.discipline_type = 'mechanical';
 END_ENTITY;
(*
```

Formal propositions:

WR1: The **discipline_type** of the **mechanical_context** entity shall contain the value ‘mechanical’.

5.2.1.3 Mechanical design using boundary representation imported entity modifications

5.2.1.3.1 geometric_representation_context

The base definition of the **geometric_representation_context** entity is given in ISO 10303-42. The following modifications apply to this part of ISO 10303.

Associated global rules:
The following global rule defined in this Part of ISO 10303 applies to the **geometric_representation_-context** entity:

global_units_in_geometric_representation_context, see 5.2.2.18.

5.2.1.3.2 geometric_representation_item

The base definition of the **geometric_representation_item** entity is given in ISO 10303-42. The following modifications apply to this part of ISO 10303.

Associated global rules:
The following global rules defined in this Part of ISO 10303 apply to the **geometric_representation_item** entity:

dependent_instantiation_of_geometry, see 5.2.2.3.

no_complex_subtypes, see 5.2.2.21.
three_dimensional_geometry, see 5.2.2.26.

5.2.1.3.3 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.

Assoiated global rules:
The following global rule defined in this Part of ISO 10303 applies to the **global_unit_assigned_context** entity:

global_units_required, see 5.2.2.19.

5.2.1.3.4 mapped_item

The base definition of the **mapped_item** entity is given in ISO 10303-43. The following modifications apply to this Part of ISO 10303.

Associated global rules:
The following global rule defined in this Part of ISO 10303 applies to the **mapped_item** entity:

dependent_instantiation_of_mapped_item, see 5.2.2.4.

NOTE 1 If a **cartesian_transformation_operator_3d** is included as **mapped_item.mapping_target** with an **axis2_placement_3d** corresponding to the original coordinate system as **mapped_representation.mapping_origin** then the resulting **mapped_item** is a transformed copy of the **representation** which is the **representation_map.mapped_representation**. The precise definition of the transformation, including translation, rotation, scaling and, if appropriate, mirroring, is given by the transformation operator.

5.2.1.3.5 mechanical_design_geometric_presentation_representation

The base definition of the **mechanical_design_geometric_presentation_representation** entity is given in ISO 10303-517. The following modifications apply to this Part of ISO 10303.

Associated global rules:
The following global rule defined in this Part of ISO 10303 applies to the **mechanical_design_geometric_-presentation_representation** entity:

dependent_instantiation_of_mechanical_design_geometric_p_r, see 5.2.2.5.

5.2.1.3.6 mechanical_design_shaded_presentation_representation

The base definition of the **mechanical_design_shaded_presentation_representation** entity is given in ISO 10303-518. The following modifications apply to this Part of ISO 10303.

Associated global rules:
The following global rule defined in this Part of ISO 10303 applies to the **mechanical_design_shaded_-presentation_representation** entity:

dependent_instantiation_of_mechanical_design_shaded_p_r, see 5.2.2.6.

5.2.1.3.7 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 rules:
The following global rules defined in this Part of ISO 10303 apply to the **named_unit** entity:

dependent_instantiation_of_named_unit, see 5.2.2.7.
illegal_complex_named_units, see 5.2.2.20.

5.2.1.3.8 pre_defined_item

The base definition of the **pre_defined_item** entity is given in ISO 10303-41. The following modifications apply to this Part of ISO 10303.

Associated global rules:
The following global rule defined in this Part of ISO 10303 applies to the **pre_defined_item** entity:

dependent_instantiation_of_pre_defined_item, see 5.2.2.8.

5.2.1.3.9 presentation_style_assignment

The base definition of the **presentation_style_assignment** entity is given in ISO 10303-46. The following modifications apply to this Part of ISO 10303. **Associated global rules:**
The following global rule defined in this Part of ISO 10303 applies to the **presentation_style_assignment** entity:

dependent_instantiation_of_presentation_style, see 5.2.2.9.

5.2.1.3.10 product_context

The base definition of the **product_context** entity is given in ISO 10303-41. The following modifications apply to this Part of ISO 10303.

Associated global rules:
The following global rules defined in this Part of ISO 10303 apply to the **product_context** entity:

product_context_mechanical, see 5.2.2.22.
dependent_instantiation_of_product_context, see 5.2.2.10

5.2.1.3.11 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 rule defined in this Part of ISO 10303 applies to the **product_definition** entity:

dependent_instantiation_of_product_definition, see 5.2.2.11.

5.2.1.3.12 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 rules:
The following global rules defined in this Part of ISO 10303 apply to the **product_definition_context** entity:

product_definition_context_design, see 5.2.2.23.
dependent_instantiation_of_product_definition, see 5.2.2.12.

5.2.1.3.13 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 rule defined in this Part of ISO 10303 applies to the **product_definition_formation** entity:

dependent_instantiation_of_product_definition_formation, see 5.2.2.13.

5.2.1.3.14 product_definition_relationship

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 rule defined in this Part of ISO 10303 applies to the **product_definition_relationship** entity:

dependent_instantiation_of_product_definition_relationship, see 5.2.2.14.

5.2.1.3.15 shape_representation

The base definition of the **shape_representation** entity is given in ISO 10303-41. The following modifications apply to this Part of ISO 10303.

Associated global rules:
The following global rule defined in this Part of ISO 10303 applies to the **shape_representation** entity:

advanced_or_elementary_or_faceted, see 5.2.2.1.

5.2.1.3.16 solid_model

The base definition of the **solid_model** entity is given in ISO 10303-42. The following modifications apply to this part of ISO 10303.

The definition of **solid_model** is modified as follows:

A **solid_model** is a complete representation of the nominal shape of a mechanical product such that all points in the interior are connected. Any point can be classified as being inside, outside, or on the boundary of the solid.

In the context of this part of ISO 10303 a **solid_model** is a **manifold_solid_brep**.

5.2.1.3.17 styled_item

The base definition of the **styled_item** entity is given in ISO 10303-46. The following modifications apply to this Part of ISO 10303.

Associated global rules:
The following global rule defined in this Part of ISO 10303 applies to the **styled_item** entity:

dependent_instantiation_of_styled_item, see 5.2.2.16.

5.2.1.3.18 topological_representation_item

The base definition of the **topological_representation_item** entity is given in ISO 10303-42. The following modifications apply to this Part of ISO 10303.

Associated global rules:
The following global rule defined in this Part of ISO 10303 applies to the **topological_representation_item** entity:

dependent_instantiation_of_topology, see 5.2.2.17.

5.2.2 Mechanical design using boundary representation rule definitions

5.2.2.1 advanced_or_elementary_or_faceted

The **advanced_or_elementary_or_faceted** rule ensures that any instance of a **shape_representation** conforms to the specification of one of:

— an **advanced_brep_shape_representation**;

— an **elementary_brep_shape_representation**;

— a **faceted_brep_shape_representation**.

EXPRESS specification:

```
*)
RULE
advanced_or_elementary_or_faceted FOR(shape_representation);
WHERE
   WR1: SIZEOF (QUERY (sr <* shape_representation |
                NOT( SIZEOF(
```



```
       ['PART_204_BREP_PRODUCT_SCHEMA.ADVANCED_BREP_SHAPE_REPRESENTATION',
        'PART_204_BREP_PRODUCT_SCHEMA.ELEMENTARY_BREP_SHAPE_REPRESENTATION',
        'PART_204_BREP_PRODUCT_SCHEMA.FACETED_BREP_SHAPE_REPRESENTATION']
           * TYPEOF (sr)) =1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

shape_representation: identifies the set of all instances of **shape_representation** entities to which the rule is applied.

Formal propositions:

WR1: Each instance of **shape_representation** shall be either an **advanced_brep_shape_representation**, an **elementary_brep_shape_representation**, or a **faceted_brep_shape_representation**.

5.2.2.2 compatible_dimension rule

The rule **compatible_dimension** ensures that:

a) all **geometric_representation_item**s are geometrically founded in one or more **geometric_representation_context** coordinate spaces;

b) when **geometric_representation_item**s are geometrically founded together in a coordinate space, they have the same coordinate space **dimension_count** by ensuring that each matches the **dimension_count** of the coordinate space in which it is geometrically founded.

NOTE Two-dimensional **geometric_representation_item**s that are geometrically founded in a **geometric_representation_context** are only geometrically founded in **geometric_representation_context**s with a **coordinate_space_dimension** of 2.
All **geometric_representation_items** founded in such a context are two-dimensional. All other values of **dimension_count** behave similarly.

EXPRESS specification:

```
 *)
RULE compatible_dimension FOR
  (cartesian_point,
  direction,
  representation_context,
  geometric_representation_context);
WHERE

  -- ensure that the count of coordinates of each cartesian_point
```

```
  -- matches the coordinate_space_dimension of each geometric_context in
  -- which it is geometrically founded
  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;

  -- ensure that the count of direction_ratios of each direction
  -- matches the coordinate_space_dimension of each geometric_context in
  -- which it is geometrically founded
  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;
(*
```

Formal propositions:

WR1: There shall be no **cartesian_point** that has a number of coordinates that differs from the **coordinate_space_dimension** of the **geometric_representation_context**s in which it is geometrically founded.

WR2: There shall be no **direction** that has a number of **direction_ratios** that differs from the **coordinate_space_dimension** of the **geometric_representation_context**s in which it is geometrically founded.

NOTE A check of only **cartesian_point**s and **direction**s is sufficient for all **geometric_representation_item**s because:

a) All **geometric_representation_item**s appear in trees of **representation_item**s descending from the **items** attribute of entity **representation**. See WR1 of entity **representation_item** in ISO 10303-43.

b) Each **geometric_representation_item** gains its position and orientation information only by being, or referring to, a **cartesian_point** or **direction** entity in such a tree. In many cases this reference is made via an **axis_placement**.

c) No other use of any **geometric_representation_item** is allowed that would associate it with a coordinate space or otherwise assign a **dimension_count**.

5.2.2.3 dependent_instantiation_of_geometry

This rule ensures that any instance of a **geometric_representation_item** which is defined is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
```



```
RULE dependent_instantiation_of_geometry FOR (geometric_representation_item);
   WHERE
      WR1 : SIZEOF ( QUERY (gri <* geometric_representation_item |
                  NOT (SIZEOF (USEDIN (gri,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

geometric_representation_item: identifies the set of all instances of **geometric_representation_item** subtypes.

Formal propositions:

WR1: Any instance of **geometric_representation_item** which occurs in this application protocol shall be used in the definition of some other entity.

5.2.2.4 dependent_instantiation_of_mapped_item

This rule ensures that any instance of a **mapped_item** which is defined is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_mapped_item FOR (mapped_item);
   WHERE
      WR1 : SIZEOF ( QUERY (mi <* mapped_item |
                  NOT (SIZEOF (USEDIN (mi,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

mapped_item: identifies the set of all instances of **mapped_item** subtypes.

Formal propositions:

WR1: Any instance of **mapped_item** which occurs in this application protocol shall be used in the definition of some other entity.

5.2.2.5 dependent_instantiation_of_mechanical_design_-geometric_p_r

This rule ensures that any instance of a **mechanical_design_geometric_presentation_representation** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_mechanical_design_geometric_p_r
   FOR (mechanical_design_geometric_presentation_representation);
WHERE
  WR1 : SIZEOF ( QUERY (mdpr <*
        mechanical_design_geometric_presentation_representation |
        NOT (SIZEOF (USEDIN (mdpr,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

mechanical_design_geometric_presentation_representation: identifies the set of all instances of **mechanical_design_geometric_presentation_representation**.

Formal propositions:

WR1: Any instance of a **mechanical_design_geometric_presentation_representation** shall be used in the definition of some other entity.

5.2.2.6 dependent_instantiation_of_mechanical_design_shaded_p_r

This rule ensures that any instance of a **mechanical_design_shaded_presentation_representation** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_mechanical_design_shaded_p_r
   FOR (mechanical_design_shaded_presentation_representation);
WHERE
  WR1 : SIZEOF ( QUERY (mdpr <*
        mechanical_design_shaded_presentation_representation |
        NOT (SIZEOF (USEDIN (mdpr,'')) > 0 ))) = 0;
```



```
END_RULE;
(*
```

Argument definitions:

mechanical_design_shaded_presentation_representation: identifies the set of all instances of **mechanical_design_shaded_presentation_representation**.

Formal propositions:

WR1: Any instance of a **mechanical_design_shaded_presentation_representation** shall be used in the definition of some other entity.

5.2.2.7 dependent_instantiation_of_named_unit

This rule ensures that any instance of a **named_unit** which is defined is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_named_unit FOR (named_unit);
   WHERE
      WR1 : SIZEOF ( QUERY (nmu <* named_unit |
                  NOT (SIZEOF (USEDIN (nmu,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

named_unit: identifies the set of all instances of **named_unit** subtypes.

Formal propositions:

WR1: Any instance of **named_unit** which occurs in this application protocol shall be used in the definition of some other entity.

5.2.2.8 dependent_instantiation_of_pre_defined_item

This rule ensures that any instance of a **pre_defined_item** which is defined is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_pre_defined_item FOR
                  (pre_defined_item);
   WHERE
      WR1 : SIZEOF ( QUERY (pdi <* pre_defined_item |
                NOT (SIZEOF (USEDIN (pdi,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

pre_defined_item: identifies the set of all instances of **pre_defined_item** subtypes.

Formal propositions:

WR1: Any instance of **pre_defined_item** which occurs in this application protocol shall be used in the definition of some other entity.

5.2.2.9 dependent_instantiation_of_presentation_style

This rule ensures that any instance of a **presentation_style_assignment** which is defined is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_presentation_style FOR
                  (presentation_style_assignment);
   WHERE
      WR1 : SIZEOF ( QUERY (psa <* presentation_style_assignment |
                NOT (SIZEOF (USEDIN (psa,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

presentation_style_assignment: identifies the set of all instances of **presentation_style_assignment** subtypes.

Formal propositions:

WR1: Any instance of **presentation_style_assignment** which occurs in this application protocol shall be used in the definition of some other entity.

5.2.2.10 dependent_instantiation_of_product_context

This rule ensures that any instance of a **product_context** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_product_context FOR (product_context);
WHERE
  WR1 : SIZEOF ( QUERY (pc <* product_context |
        NOT (SIZEOF (USEDIN (pc,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

product_context: identifies the set of all instances of **product_context**.

Formal propositions:

WR1: Any instance of a **product_context** shall be used in the definition of some other entity.

5.2.2.11 dependent_instantiation_of_product_definition

This rule ensures that any instance of a **product_definition** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_product_definition FOR (product_definition);
WHERE
  WR1 : SIZEOF ( QUERY (pd <* product_definition |
        NOT (SIZEOF (USEDIN (pd,'')) > 0 ))) = 0;
END_RULE;
(*
```


Argument definitions:

product_definition: identifies the set of all instances of **product_definition**.

Formal propositions:

WR1: Any instance of a **product_definition** shall be used in the definition of some other entity.

5.2.2.12 dependent_instantiation_of_product_definition_context

This rule ensures that any instance of a **product_definition_context** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_product_definition_context FOR
     (product_definition_context);
WHERE
  WR1 : SIZEOF ( QUERY (pdc <* product_definition_context |
        NOT (SIZEOF (USEDIN (pdc,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

product_definition_context: identifies the set of all instances of **product_definition_context**.

Formal propositions:

WR1: Any instance of a **product_definition_context** shall be used in the definition of some other entity.

5.2.2.13 dependent_instantiation_of_product_definition_formation

This rule ensures that any instance of a **product_definition_formation** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_product_definition_formation
```



```
            FOR (product_definition_formation);
WHERE
  WR1 : SIZEOF ( QUERY (pdf <* product_definition_formation |
        NOT (SIZEOF (USEDIN (pdf,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

product_definition_formation: identifies the set of all instances of **product_definition_formation**.

Formal propositions:

WR1: Any instance of a **product_definition_formation** shall be used in the definition of some other entity.

5.2.2.14 dependent_instantiation_of_product_definition_relationship

This rule ensures that any instance of a **product_definition_relationship** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_product_definition_relationship FOR
     (product_definition_relationship);
WHERE
  WR1 : SIZEOF ( QUERY (pdr <* product_definition_relationship |
        NOT (SIZEOF (USEDIN (pdr,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

product_definition_relationship: identifies the set of all instances of **product_definition_relationship**.

Formal propositions:

WR1: Any instance of a **product_definition_relationship** shall be used in the definition of some other entity.

5.2.2.15 dependent_instantiation_of_shape_representation

This rule ensures that any instance of a **shape_representation** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_shape_representation
                                FOR (shape_representation);
WHERE
  WR1 : SIZEOF ( QUERY (sr <* shape_representation |
        NOT (SIZEOF (USEDIN (sr,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

shape_representation: identifies the set of all instances of **shape_representation**.

Formal propositions:

WR1: Any instance of a **shape_representation** shall be used in the definition of some other entity.

5.2.2.16 dependent_instantiation_of_styled_item

This rule ensures that any instance of a **styled_item** which is defined is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_styled_item FOR (styled_item);
   WHERE
      WR1 : SIZEOF ( QUERY (si <* styled_item |
                  NOT (SIZEOF (USEDIN (si,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

styled_item: identifies the set of all instances of **styled_item** subtypes.

Formal propositions:

WR1: Any instance of **styled_item** which occurs in this application protocol shall be used in the definition of some other entity.

5.2.2.17 dependent_instantiation_of_topology

This rule ensures that any instance of a **topological_representation_item** which is defined is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_topology
             FOR (topological_representation_item);
   WHERE
      WR1 : SIZEOF ( QUERY (tri <* topological_representation_item |
                  NOT (SIZEOF (USEDIN (tri,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

topological_representation_item: identifies the set of all instances of **topological_representation_item** subtypes.

Formal propositions:

WR1: Any instance of **topological_representation_item** which occurs in this application protocol shall be used in the definition of some other entity.

5.2.2.18 global_units_in_geometric_representation_context

The **global_units_in_geometric_representation_context** rule requires that each **geometric_representation_context** is also a **global_unit_assigned_context**. This implies that in case of cartesian geometry all units shall be defined as global units that are valid within specfic contexts.

EXPRESS specification:

```
*)
RULE global_units_in_geometric_representation_context FOR
      (geometric_representation_context);
```



```
WHERE
  WR1 :  SIZEOF (QUERY (grc <* geometric_representation_context |
         NOT ('PART_204_BREP_PRODUCT_SCHEMA.' +
         'GLOBAL_UNIT_ASSIGNED_CONTEXT' IN TYPEOF (grc)))) = 0 ;
END_RULE;
(*
```

Argument definitions:

geometric_representation_context: identifies the set of all instances of **geometric_representation_context** entities to which the rule is applied.

Formal propositions:

WR1: The instance of a **geometric_representation_context** shall always also be of type **global_unit_assigned_context**. That means that cartesian geometry shall always be assigned default units, whereas parametric geometry that does reference **parameter_value** such as **pcurve** need not be assigned units.

5.2.2.19 global_units_required

The **global_units_required** rule specifies the units that shall be defined for **global_unit_assigned_context**. Every **global_unit_assigned_context** shall have a maximum of 2 elements in its set of units; these shall include a unit of length and may also include a unit of plane angle measure.

EXPRESS specification:

```
*)
RULE global_units_required 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 |
                    'PART_204_BREP_PRODUCT_SCHEMA.LENGTH_UNIT' IN TYPEOF
                    (u))) =1 ))) = 0;
      WR3:  SIZEOF ( QUERY (guac <* global_unit_assigned_context |
                  NOT (SIZEOF (QUERY( u <* guac.units |
         NOT (( 'PART_204_BREP_PRODUCT_SCHEMA.LENGTH_UNIT' IN TYPEOF(u))
                  OR ( 'PART_204_BREP_PRODUCT_SCHEMA.PLANE_ANGLE_UNIT' IN
                        TYPEOF (u))))) = 0))) = 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 any instance of **global_unit_assigned_context** which occurs in this application protocol the set of units shall contain 2 or less elements.

WR2: For any instance of **global_unit_assigned_context** which occurs in this application protocol the set of units shall contain a **length_unit**.

WR3: For any instance of **global_unit_assigned_context** which occurs in this application protocol the set of units shall contain only **length_unit**s or **plane_angle_unit**s.

Informal propositions:

IP1: Any entity instance containing a **length_measure** as part of its definition shall only occur in a **global_unit_assigned_context**.

IP2: Any entity instance containing a **plane_angle_measure** as part of its definition shall only occur in a **global_unit_assigned_context** for which a **plane_angle_unit** is defined.

5.2.2.20 illegal_complex_named_units

The following rule ensures that a **named_unit** can not simultaneously be a **length_unit** and a **plane_angle_unit** and that a **named_unit** can not simultaneously be a **length_unit** and a **conversion_based_unit**.

EXPRESS specification:

```
*)
RULE illegal_complex_named_units FOR (named_unit);
WHERE
  WR1: SIZEOF (QUERY (nu <* named_unit |
       NOT (SIZEOF (TYPEOF(nu) *
       ['PART_204_BREP_PRODUCT_SCHEMA.LENGTH_UNIT',
        'PART_204_BREP_PRODUCT_SCHEMA.PLANE_ANGLE_UNIT']) < 2 )
       AND
       NOT (SIZEOF (TYPEOF(nu) *
       ['PART_204_BREP_PRODUCT_SCHEMA.LENGTH_UNIT',
        'PART_204_BREP_PRODUCT_SCHEMA.CONVERSION_BASED_UNIT']) < 2 )
       )) = 0;
END_RULE;
(*
```


Argument definitions:

named_unit: identifies the set of all instances of **named_unit** entities to which the rule is applied.

Formal propositions:

WR1: The TYPEOF function shall not return two or more of the subtypes in the two lists in this rule for any **named_unit**. I.e. a **named_unit** shall not be used as a combination of **length_unit** and **plane_angle_unit**. And a **length_unit** shall not be based on conversion, but shall be a pure **si_unit**. Only **plane_angle_unit**s may deviate from **si_unit**s.

5.2.2.21 no_complex_subtypes

The following rule is equivalent to a ONEOF declaration and ensures that no subtype of **geometric_representation_item** can simultaneously be of more than one of the subtypes listed in the rule.

EXPRESS specification:

```
*)
RULE no_complex_subtypes FOR(geometric_representation_item);
  WHERE
   WR1: SIZEOF (QUERY (gri <* geometric_representation_item |
   NOT (SIZEOF (TYPEOF(gri) *
            ['PART_204_BREP_PRODUCT_SCHEMA.CAMERA_IMAGE',
             'PART_204_BREP_PRODUCT_SCHEMA.CAMERA_MODEL',
     'PART_204_BREP_PRODUCT_SCHEMA.CARTESIAN_TRANSFORMATION_OPERATOR',
              'PART_204_BREP_PRODUCT_SCHEMA.CURVE',
               'PART_204_BREP_PRODUCT_SCHEMA.DIRECTION',
          'PART_204_BREP_PRODUCT_SCHEMA.EDGE_CURVE',
          'PART_204_BREP_PRODUCT_SCHEMA.FACE_SURFACE',
           'PART_204_BREP_PRODUCT_SCHEMA.PLACEMENT',
            'PART_204_BREP_PRODUCT_SCHEMA.POINT',
          'PART_204_BREP_PRODUCT_SCHEMA.POLY_LOOP',
            'PART_204_BREP_PRODUCT_SCHEMA.SOLID_MODEL',
             'PART_204_BREP_PRODUCT_SCHEMA.SURFACE',
              'PART_204_BREP_PRODUCT_SCHEMA.VECTOR',
          'PART_204_BREP_PRODUCT_SCHEMA.VERTEX_POINT']) < 2 ))) = 0;
END_RULE;
(*
```

Argument definitions:

geometric_representation_item: identifies the set of all instances of **geometric_representation_item** entities to which the rule is applied.

Formal propositions:

WR1: The TYPEOF function shall not return two or more of the subtypes listed in this rule for any **geometric_representation_item** used in this AP.

5.2.2.22 product_context_mechanical

In the context of this AP the only valid subtype of **product_context** is **mechanical_context**.

EXPRESS specification:

```
*)
RULE product_context_mechanical FOR (product_context);
   WHERE
      WR1 : SIZEOF ( QUERY (pc <* product_context |
                  NOT ('PART_204_BREP_PRODUCT_SCHEMA.MECHANICAL_CONTEXT'
            IN TYPEOF(pc)))) = 0;
END_RULE;
(*
```

Argument definitions:

product_context: identifies the set of all instances of **product_context** entities.

Formal propositions:

WR1: Any instance of **product_context** which occurs in this application protocol shall be of the **mechanical_context** subtype.

5.2.2.23 product_definition_context_design

In the context of this AP the only valid **product_definition_context**s are of the special subtype **design_context**.

EXPRESS specification:

```
*)
RULE product_definition_context_design FOR (product_definition_context);
   WHERE
      WR1 : SIZEOF ( QUERY (pdc <* product_definition_context |
                  NOT ('PART_204_BREP_PRODUCT_SCHEMA.DESIGN_CONTEXT'
                    IN TYPEOF(pdc)))) = 0;
```

```
END_RULE;
(*
```

Argument definitions:

product_definition_context: identifies the set of all instances of **product_definition_context** entities.

Formal propositions:

WR1: Any instance of **product_definition_context** which occurs in this application protocol shall be of subtype **design_context**.

5.2.2.24 shape_representation_required

This rule ensures that any instance of **product_definition_shape** is associated, via the **shape_definition_representation** entity, with at least one **shape_representation**.

EXPRESS specification:

```
*)
RULE shape_representation_required FOR (product_definition_shape);
   WHERE
      WR1 : SIZEOF ( QUERY (pds <* product_definition_shape |
              NOT ( SIZEOF(USEDIN
            (pds,'PART_204_BREP_PRODUCT_SCHEMA.
                    SHAPE_DEFINITION_REPRESENTATION\
                    PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) > 0
                 ))) = 0;
END_RULE;  -- shape_representation_required
(*
```

Argument definitions:

product_definition_shape: identifies the set of all instances of **product_definition_shape** entities.

Formal propositions:

WR1: Any instance of **product_definition_shape** which occurs in this application protocol shall be used at least once as a **definition** attribute of a **shape_definition_representation**. This ensures that the **product_definition_shape** is associated with a **manifold_solid_brep** via a **shape_representation**.

5.2.2.25 single_curve_style

This rule ensures that any instance of **curve** is associated, via the **styled_item** entity, with no more than one **curve_style**.

EXPRESS specification:

```
*)
RULE single_curve_style FOR (curve);
   WHERE
      WR1 : SIZEOF ( QUERY (crv <* curve |
                 SIZEOF(USEDIN
            (crv,'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM.ITEM')) > 1
                 )) = 0;
END_RULE;  -- single_curve_style
(*
```

Argument definitions:

curve: identifies the set of all instances of **curve** entities.

Formal propositions:

WR1: Any instance of **curve** which occurs in this application protocol shall be used at most once as an **item** attribute of a styled_item. This ensures that a maximum of one set of presentation styles is associated with any **curve** instance.

5.2.2.26 three_dimensional_geometry

The following rule ensures that all geometry used in the definition of a B-rep product is three-dimensional or is one of the specialised entities used in the presentation of a two-dimensional image or the definition of a **pcurve**. All geometric representation items have a derived integer attribute **dim** which gives the dimension of the space in which they are defined. This rule verifies that two-dimensional geometry is not used in the definition of geometric models except for defining curves in parameter space.

EXPRESS specification:

```
 *)
 RULE three_dimensional_geometry FOR(geometric_representation_item);
   WHERE
    WR1: SIZEOF(QUERY(gri <* geometric_representation_item |
         (dimension_of(gri) = 2) AND NOT ( (SIZEOF (
            ['PART_204_BREP_PRODUCT_SCHEMA.PLANAR_BOX',
```



```
            'PART_204_BREP_PRODUCT_SCHEMA.AXIS2_PLACEMENT_2D'] *
              TYPEOF(gri)) = 1 )
              OR
              (SIZEOF(QUERY(rep <* using_representations(gri) |
              'PART_204_BREP_PRODUCT_SCHEMA.DEFINITIONAL_REPRESENTATION'
           IN TYPEOF(rep) )) > 0 )
              ))) = 0;
 END_RULE;
(*
```

Argument definitions:

geometric_representation_item: identifies the set of all instances of **geometric_representation_item** entities to which the rule is applied.

Formal propositions:

WR1: Each instance of **geometric_representation_item** occurring in this AP shall have a dimension of 3 or be of type **planar_box** or of type **annotation_text**, or shall be used too define, via the **definional_representation** entity, a **reference_to_curve** attribute of a **pcurve**.

EXPRESS specification:

```
*)
END_SCHEMA; --end part 204 brep product schema;
(*
```


6 Conformance requirements

Conformance to this part of ISO 10303 includes satisfying the requirements stated in this part, the requirements of the implementation method(s) supported, and the relevant requirements of the normative references.

An implementation shall support at least one of the following implementation methods: ISO 10302-21, ISO 10303-22.

Requirements with respect to implementation methods-specific requirements 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 the following conformance classes.

Class 1: B-rep level 1: The definition of a mechanical engineering product model where the shape is represented by one or more faceted_brep_models.

Class 2: B-rep level 2: The definition of a mechanical engineering product model where the shape is represented by one or more elementary_brep_models.

Class 3: B-rep level 3: The definition of a mechanical engineering product model where the shape is represented by one or more advanced_brep_models.

Support for a particular conformance class requires support for all the options specified in that class.

NOTE 1 ISO 10303-304 defines the abstract test suite and test purposes to be used in the assessment of conformance.

General requirements for all classes are:

a) The information requirements and relationships of the ARM shall be preserved in the implementation. This includes support for all valid combinations of entities and their attributes. No ‘substitution’ of entities shall be permitted. Consequently all construct assertions from clause 4 shall be maintained.

b) All AIM entities, types, and their associated constraints shall be supported. Treatment of options and default values shall conform to the AIM.

c) All AIM entities, types, and their associated constraints shall be read and processed by a postprocessor.

d) Entities not belonging to the AIM shall be excluded from a preprocessor implementation. Entities not specified in the AIM shall not be included in a conforming exchange structure.

e) An implementation shall satisfy all general requirements of implementation methods (given in the appropriate 20-series class). This includes the preservation of the AIM mapped onto the implementation form and conformance to the syntax of the implementation form.

Table 8 – Units of functionality within conformance classes

	Class 1	Class 2	Class 3
name preservation	O	O	O
product structure	R	R	R
visual presentation for B-rep	O	O	O
faceted B-rep	R		
elementary B-rep		R	
advanced B-rep			R

NOTE 2 In Table 8, O denotes an optional capability which may be supported by the pre-processor and shall be supported by all post-processors. R denotes a requirement on both pre- and post-processor.

6.1 Conformance class 1: B-rep level 1 (CC1)

The scope of CC1 is:

— Shape representatation using **faceted_brep**s;

— Geometry of planes;

— Geometric transformations;

— Poly_loops with implicit edges;

— Topology of faces and shells;

— Association of shape models with products;

— Preservation of names of products and of geometric elements;

— Product structure including assemblies.

Optionally visual presentation properties may be associated with faceted shape models.

6.2 Conformance class 2: B-rep level 2 (CC2)

The scope of CC2 is:

— Shape representatation using B-reps with elementary geometry;

— Geometry of curves and surfaces;

— Geometric transformations;

— Topology of edges, faces and shells;

— Association of shape models with products;

— Preservation of names of products and of geometric elements;

— Product structure including assemblies.

Optionally visual presentation properties may be associated with elementary B-rep models.

6.3 Conformance class 3: B-rep level 3 (CC3)

The scope of CC3 is:

— Shape representatation using adnanced B-reps;

— Geometry of curves and surfaces including B-spline geometry;

— Geometric transformations;

— Topology of edges, faces and shells;

— Association of shape models with products;

— Preservation of names of products and of geometric elements;

— Product structure including assemblies.

Optionally visual presentation properties may be associated with advanced B-rep models.

NOTE Table 9 shows the usage of significant AIM entities within conformance classes. In this table O denotes an optional requirement.

Table 9 – AIM entities within conformance classes

	Class 1	Class 2	Class 3
advanced_brep_shape_representation			R
elementary_brep_shape_representation		R	
faceted_brep_shape_representation	R		
advanced_face			R
face_surface	R	R	
cartesian_point	R	R	R
vertex_point		R	R
poly_loop	R		
line		R	R
polyline		R	R
edge_curve		R	R
conic subtypes		R	R
pcurve			R
b_spline_curve			R
plane	R	R	R
elementary_surface subtypes		R	R
b_spline_surface			R
cartesian_transformation_operator	R	R	R
mapped_item	R	R	R
product_definition	R	R	R
product_definition_relationship	R	R	R
assembly_component_usage	R	R	R
geometric_representation_context	R	R	R
global_unit_assigned_context	R	R	R
camera_model_d3	O	O	O
presentation_style_assignment	O	O	O
light_source	O	O	O
curve_appearance		O	O
surface_appearance	O	O	O

Annex A
(normative)

AIM EXPRESS expanded listing

This Annex includes all long forms related to the different AICs used in this document.

A.1 AIM EXPRESS listing

```
*)
SCHEMA part_204_brep_product_schema; -- from part 204 FDIS WG3 N1025 July 2001.
CONSTANT
 dummy_gri : geometric_representation_item := representation_item('')||
                  geometric_representation_item();

   dummy_tri : topological_representation_item := representation_item('')||
                  topological_representation_item();
END_CONSTANT;

(* types below are introduced only to solve compiler problems arising
   in topology reverse functions, etc., they are not to be used explicitly:
  shell, transition_code, list_of_reversible_topology_item
  geometric_set_select, trimming_select
*)
TYPE area_or_view = SELECT
  (presentation_area,
   presentation_view);
END_TYPE;

TYPE axis2_placement = SELECT
   (axis2_placement_2d,
    axis2_placement_3d);
END_TYPE;

TYPE BOOLEAN_OPERAND = SELECT (
   SOLID_MODEL);
END_TYPE;

TYPE b_spline_curve_form = ENUMERATION OF
   (polyline_form,
    circular_arc,
    elliptic_arc,
    parabolic_arc,
    hyperbolic_arc,
    unspecified);
END_TYPE;

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;

TYPE central_or_parallel = ENUMERATION OF
  (central,
   parallel);
END_TYPE;

TYPE CHARACTERIZED_DEFINITION = SELECT (
   SHAPE_DEFINITION);
END_TYPE;

TYPE CURVE_FONT_OR_SCALED_CURVE_FONT_SELECT = SELECT (
   CURVE_STYLE_FONT_SELECT);
END_TYPE;

TYPE curve_on_surface = SELECT
   (pcurve,
    surface_curve,
    composite_curve_on_surface);
END_TYPE;

TYPE curve_or_render = SELECT
  (curve_style,
   curve_style_rendering);
END_TYPE;

TYPE CURVE_STYLE_FONT_SELECT = SELECT (
   CURVE_STYLE_FONT,
   PRE_DEFINED_CURVE_FONT);
END_TYPE;

TYPE dimension_count = INTEGER;
WHERE
  WR1: SELF > 0;
END_TYPE;

TYPE direction_count_select = SELECT
  (u_direction_count,
   v_direction_count);
END_TYPE;

TYPE FILL_STYLE_SELECT = SELECT (
```



```
   FILL_AREA_STYLE_COLOUR);
END_TYPE;

TYPE founded_item_select = SELECT
  (founded_item,
   representation_item);
END_TYPE;

TYPE geometric_set_select = SELECT
   (point,
    curve,
    surface);
 END_TYPE;

TYPE identifier = STRING;
END_TYPE;

TYPE INVISIBLE_ITEM = SELECT (
   STYLED_ITEM,
   REPRESENTATION);
END_TYPE;

TYPE knot_type = ENUMERATION OF
   (uniform_knots,
    unspecified,
    quasi_uniform_knots,
    piecewise_bezier_knots);
 END_TYPE;

TYPE label = STRING;
END_TYPE;

TYPE layered_item = SELECT
  (presentation_representation,
   representation_item);
END_TYPE;

TYPE length_measure = REAL;
END_TYPE;

TYPE list_of_reversible_topology_item =
                                LIST [0:?] of reversible_topology_item;
END_TYPE;

TYPE MARKER_SELECT = SELECT (
   MARKER_TYPE);
END_TYPE;

TYPE marker_type = ENUMERATION OF
  (dot,
   x,
   plus,
```

```
   asterisk,
   ring,
   square,
   triangle);
END_TYPE;

TYPE MEASURE_VALUE = SELECT (
   LENGTH_MEASURE,
   PLANE_ANGLE_MEASURE,
   RATIO_MEASURE,
   PARAMETER_VALUE,
   POSITIVE_LENGTH_MEASURE,
   POSITIVE_PLANE_ANGLE_MEASURE,
   POSITIVE_RATIO_MEASURE);
END_TYPE;

TYPE parameter_value = REAL;
END_TYPE;

TYPE pcurve_or_surface = SELECT
   (pcurve,
    surface);
 END_TYPE;

TYPE plane_angle_measure = REAL;
END_TYPE;

TYPE positive_length_measure = length_measure;
WHERE
  WR1: SELF > 0;
END_TYPE;

TYPE positive_plane_angle_measure = plane_angle_measure;
WHERE
  WR1: SELF > 0;
END_TYPE;

TYPE positive_ratio_measure = ratio_measure;
WHERE
  WR1: SELF > 0;
END_TYPE;

TYPE preferred_surface_curve_representation = ENUMERATION OF
   (curve_3d,
    pcurve_s1,
    pcurve_s2);
 END_TYPE;

TYPE presentation_representation_select = SELECT
  (presentation_representation,
   presentation_set);
END_TYPE;
```



```
TYPE presentation_size_assignment_select = SELECT
  (presentation_view,
   presentation_area,
   area_in_set);
END_TYPE;

TYPE PRESENTATION_STYLE_SELECT = SELECT (
   POINT_STYLE,
   CURVE_STYLE,
   SURFACE_STYLE_USAGE,
   FILL_AREA_STYLE);
END_TYPE;

TYPE ratio_measure = REAL;
END_TYPE;

TYPE rendering_properties_select = SELECT
  (surface_style_reflectance_ambient,
   surface_style_transparent);
END_TYPE;

TYPE reversible_topology = SELECT
           (reversible_topology_item,
            list_of_reversible_topology_item,
            set_of_reversible_topology_item);
END_TYPE;

TYPE reversible_topology_item = SELECT
   (edge,
    path,
    face,
    face_bound,
    closed_shell,
    open_shell);
END_TYPE;

TYPE set_of_reversible_topology_item =
                      SET [0:?] of reversible_topology_item;
 END_TYPE;

TYPE shading_curve_method = ENUMERATION OF
  (constant_colour,
   linear_colour);
END_TYPE;

TYPE shading_surface_method = ENUMERATION OF
  (constant_shading,
   colour_shading,
   dot_shading,
   normal_shading);
END_TYPE;
```



```
TYPE SHAPE_DEFINITION = SELECT (
   PRODUCT_DEFINITION_SHAPE);
END_TYPE;

TYPE SHELL = SELECT (
   OPEN_SHELL,
   CLOSED_SHELL);
END_TYPE;

TYPE si_prefix = ENUMERATION OF
  (exa,
   peta,
   tera,
   giga,
   mega,
   kilo,
   hecto,
   deca,
   deci,
   centi,
   milli,
   micro,
   nano,
   pico,
   femto,
   atto);
END_TYPE;

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;

TYPE SIZE_SELECT = SELECT (
   POSITIVE_LENGTH_MEASURE,
   MEASURE_WITH_UNIT);
END_TYPE;

TYPE STYLE_CONTEXT_SELECT = SELECT (
   REPRESENTATION,
   REPRESENTATION_ITEM,
   PRESENTATION_SET);
END_TYPE;

TYPE surface_side = ENUMERATION OF
  (positive,
   negative,
   both);
END_TYPE;

TYPE SURFACE_SIDE_STYLE_SELECT = SELECT (
   SURFACE_SIDE_STYLE);
END_TYPE;

TYPE surface_style_element_select = SELECT
  (surface_style_fill_area,
   surface_style_boundary,
   surface_style_silhouette,
   surface_style_segmentation_curve,
   surface_style_control_grid,
   surface_style_parameter_line,
   surface_style_rendering);
END_TYPE;

TYPE text = STRING;
END_TYPE;

TYPE TRANSFORMATION = SELECT (
   FUNCTIONALLY_DEFINED_TRANSFORMATION);
END_TYPE;

TYPE transition_code = ENUMERATION OF
   (discontinuous,
    continuous,
    cont_same_gradient,
    cont_same_gradient_same_curvature);
END_TYPE;
```

```
TYPE trimming_select = SELECT
   (cartesian_point,
    parameter_value);
END_TYPE;
TYPE u_direction_count = INTEGER;
WHERE
  WR1: SELF > 1;
END_TYPE;

TYPE UNIT = SELECT (
   NAMED_UNIT);
END_TYPE;

TYPE v_direction_count = INTEGER;
WHERE
  WR1: SELF > 1;
END_TYPE;

TYPE vector_or_direction = SELECT
   (vector,
    direction);
END_TYPE;



(* entities below are introduced only to solve compiler problems arising
   in topology reverse functions, etc., they are not to be used explicitly:
   composite_curve, composite_curve_segment,
  open_shell,  oriented_face,  oriented_path, oriented_open_shell;
*)
 ENTITY advanced_brep_shape_representation
   SUBTYPE OF (shape_representation);
   WHERE
     WR1: SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
              'PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP',
              'PART_204_BREP_PRODUCT_SCHEMA.FACETED_BREP',
              'PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM',
              'PART_204_BREP_PRODUCT_SCHEMA.AXIS2_PLACEMENT_3D'] *
                       TYPEOF(it)) = 1)) ))  = 0;
     WR2: SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
              'PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP',
              'PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'] *
                                    TYPEOF(it)) = 1) )) > 0;
     WR3: SIZEOF(QUERY ( msb <* QUERY ( it <* SELF.items |
              ('PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP' IN
                        TYPEOF(it)) ) |
             ( NOT (SIZEOF(QUERY ( csh <* msb_shells(msb) |
                              (NOT (SIZEOF(QUERY ( fcs <* csh\
              connected_face_set.cfs_faces | (NOT (
           'PART_204_BREP_PRODUCT_SCHEMA.ADVANCED_FACE'
                       IN TYPEOF(fcs))) )) = 0)) ))
```



```
            = 0)) )) = 0;
    WR4: SIZEOF(QUERY ( msb <* QUERY ( it <* items |
           ( 'PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP' IN
                           TYPEOF(it)) ) |
           (  'PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_CLOSED_SHELL' IN
                TYPEOF(msb\manifold_solid_brep.outer)) )) = 0;
    WR5: SIZEOF(QUERY ( brv <* QUERY ( it <* items |
          ( 'PART_204_BREP_PRODUCT_SCHEMA.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 |
           ( 'PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM' IN TYPEOF(it)) ) |
           (NOT
       ( 'PART_204_BREP_PRODUCT_SCHEMA.ADVANCED_BREP_SHAPE_REPRESENTATION'
         IN TYPEOF(mi\mapped_item.mapping_source.mapped_representation)))
             )) = 0;
END_ENTITY;

ENTITY advanced_face
  SUBTYPE OF (face_surface);
WHERE
  WR1 : SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.ELEMENTARY_SURFACE',
          'PART_204_BREP_PRODUCT_SCHEMA.B_SPLINE_SURFACE',
          'PART_204_BREP_PRODUCT_SCHEMA.SWEPT_SURFACE'] *
            TYPEOF(face_geometry)) = 1;
  WR2 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
         'PART_204_BREP_PRODUCT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
          NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
              NOT('PART_204_BREP_PRODUCT_SCHEMA.EDGE_CURVE' IN
              TYPEOF(oe\oriented_edge.edge_element)))) = 0))) = 0;
  WR3 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
         'PART_204_BREP_PRODUCT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
            NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
            NOT (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.LINE',
                          'PART_204_BREP_PRODUCT_SCHEMA.CONIC',
                          'PART_204_BREP_PRODUCT_SCHEMA.POLYLINE',
                          'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_CURVE',
                          'PART_204_BREP_PRODUCT_SCHEMA.B_SPLINE_CURVE'] *
              TYPEOF(oe.edge_element\edge_curve.edge_geometry)) = 1 )
                )) = 0))) = 0;
  WR4 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
         'PART_204_BREP_PRODUCT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
            NOT(SIZEOF(QUERY (oe <* elp_fbnds.bound\path.edge_list |
             NOT((((('PART_204_BREP_PRODUCT_SCHEMA.VERTEX_POINT' IN
                 TYPEOF(oe\edge.edge_start)) AND
            ( 'PART_204_BREP_PRODUCT_SCHEMA.CARTESIAN_POINT' IN
             TYPEOF(oe\edge.edge_start\vertex_point.vertex_geometry)))) AND
                 (('PART_204_BREP_PRODUCT_SCHEMA.VERTEX_POINT' IN
                    TYPEOF(oe\edge.edge_end)) AND
                   ( 'PART_204_BREP_PRODUCT_SCHEMA.CARTESIAN_POINT' IN
```

```
                    TYPEOF(oe\edge.edge_end\vertex_point.vertex_geometry)))
             ))) = 0))) = 0;
  WR5 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
       'PART_204_BREP_PRODUCT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
            'PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_PATH' IN
             TYPEOF(elp_fbnds.bound))) = 0;
  WR6 : (NOT ('PART_204_BREP_PRODUCT_SCHEMA.SWEPT_SURFACE' IN
                                          TYPEOF(face_geometry))) OR
             (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.LINE',
                       'PART_204_BREP_PRODUCT_SCHEMA.CONIC',
                       'PART_204_BREP_PRODUCT_SCHEMA.POLYLINE',
                       'PART_204_BREP_PRODUCT_SCHEMA.B_SPLINE_CURVE'] *
                  TYPEOF(face_geometry\swept_surface.swept_curve)) = 1);
  WR7 : SIZEOF(QUERY (vlp_fbnds <* QUERY (bnds <* bounds |
      'PART_204_BREP_PRODUCT_SCHEMA.VERTEX_LOOP' IN TYPEOF(bnds.bound)) |
          NOT(('PART_204_BREP_PRODUCT_SCHEMA.VERTEX_POINT' IN
          TYPEOF(vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex)) AND
                  ('PART_204_BREP_PRODUCT_SCHEMA.CARTESIAN_POINT' IN
                     TYPEOF(vlp_fbnds\face_bound.bound\vertex_loop.
                     loop_vertex\vertex_point.vertex_geometry))
             ))) = 0;
  WR8 : SIZEOF (QUERY (bnd <* bounds |
         NOT (SIZEOF(['PART_204_BREP_PRODUCT_SCHEMA.EDGE_LOOP',
         'PART_204_BREP_PRODUCT_SCHEMA.VERTEX_LOOP'] * TYPEOF(bnd.bound))
                                 = 1))) = 0;
  WR9 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
        'PART_204_BREP_PRODUCT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
            NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
            ('PART_204_BREP_PRODUCT_SCHEMA.SURFACE_CURVE' IN
            TYPEOF(oe\oriented_edge.edge_element\edge_curve.edge_geometry))
            AND (NOT (SIZEOF (QUERY (sc_ag <*
            oe.edge_element\edge_curve.edge_geometry\
            surface_curve.associated_geometry |
            NOT ('PART_204_BREP_PRODUCT_SCHEMA.PCURVE' IN
            TYPEOF(sc_ag)))) = 0)))) = 0))) = 0;
  WR10 : ((NOT ('PART_204_BREP_PRODUCT_SCHEMA.SWEPT_SURFACE' IN
         TYPEOF(face_geometry))) OR
         ((NOT ('PART_204_BREP_PRODUCT_SCHEMA.POLYLINE' IN
         TYPEOF(face_geometry\swept_surface.swept_curve))) OR
         (SIZEOF(face_geometry\swept_surface.swept_curve\polyline.points)
         >= 3))) AND
         (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* bounds |
         'PART_204_BREP_PRODUCT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
         NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
         ('PART_204_BREP_PRODUCT_SCHEMA.POLYLINE' IN
        TYPEOF(oe\oriented_edge.edge_element\edge_curve.edge_geometry)) AND
         (NOT (SIZEOF (oe\oriented_edge.edge_element\
            edge_curve.edge_geometry\polyline.points) >= 3)))) = 0))) = 0);
 END_ENTITY;

ENTITY application_context;
  application          : text;
```



```
INVERSE
  context_elements : SET [1:?] OF  application_context_element
                                FOR frame_of_reference;
END_ENTITY;

ENTITY application_context_element
  SUPERTYPE OF (ONEOF (
              PRODUCT_CONTEXT,
              PRODUCT_DEFINITION_CONTEXT));
  name                 : label;
  frame_of_reference   : application_context;
END_ENTITY;

ENTITY area_in_set;
  area   : presentation_area;
  in_set : presentation_set;
END_ENTITY;

ENTITY assembly_component_usage
  SUBTYPE OF (product_definition_usage);
  reference_designator : OPTIONAL identifier;
END_ENTITY;

ENTITY axis1_placement
 SUBTYPE OF (placement);
   axis     : OPTIONAL direction;
 DERIVE
   z : direction := NVL(normalise(axis), dummy_gri ||
                                 direction([0.0,0.0,1.0]));
 WHERE
   WR1: SELF\geometric_representation_item.dim  = 3;
END_ENTITY;

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;

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 > 0.0);
 END_ENTITY;

ENTITY background_colour
  SUBTYPE OF (colour);
  presentation : area_or_view;
UNIQUE
  UR1:  presentation;
END_ENTITY;

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: ('PART_204_BREP_PRODUCT_SCHEMA.UNIFORM_CURVE'
         IN TYPEOF(self)) OR
        ('PART_204_BREP_PRODUCT_SCHEMA.QUASI_UNIFORM_CURVE'
            IN TYPEOF(self)) OR
        ('PART_204_BREP_PRODUCT_SCHEMA.BEZIER_CURVE' IN TYPEOF(self)) OR
        ('PART_204_BREP_PRODUCT_SCHEMA.B_SPLINE_CURVE_WITH_KNOTS'
            IN TYPEOF(self));
END_ENTITY;

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;

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: ('PART_204_BREP_PRODUCT_SCHEMA.UNIFORM_SURFACE' IN TYPEOF(SELF)) OR
        ('PART_204_BREP_PRODUCT_SCHEMA.QUASI_UNIFORM_SURFACE' IN
              TYPEOF(SELF)) OR
        ('PART_204_BREP_PRODUCT_SCHEMA.BEZIER_SURFACE' IN TYPEOF(SELF)) OR
        ('PART_204_BREP_PRODUCT_SCHEMA.B_SPLINE_SURFACE_WITH_KNOTS'
            IN TYPEOF(SELF));
END_ENTITY;

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;

ENTITY bezier_curve
   SUBTYPE OF (b_spline_curve);
END_ENTITY;

ENTITY bezier_surface
   SUBTYPE OF (b_spline_surface);
 END_ENTITY;

ENTITY bounded_curve
   SUPERTYPE OF (ONEOF (
              POLYLINE,
              B_SPLINE_CURVE,
              COMPOSITE_CURVE))
  SUBTYPE OF (curve);
END_ENTITY;

ENTITY bounded_surface
   SUPERTYPE OF (ONEOF (
              B_SPLINE_SURFACE))
  SUBTYPE OF (surface);
END_ENTITY;

ENTITY brep_with_voids
   SUBTYPE OF (manifold_solid_brep);
   voids : SET [1:?] OF oriented_closed_shell;
END_ENTITY;


ENTITY camera_image
  SUBTYPE OF (mapped_item);
WHERE
  WR1: 'PART_204_BREP_PRODUCT_SCHEMA.CAMERA_USAGE'
       IN TYPEOF (SELF\mapped_item.mapping_source);
  WR2: 'PART_204_BREP_PRODUCT_SCHEMA.PLANAR_BOX'
       IN TYPEOF (SELF\mapped_item.mapping_target);
  WR3: 'PART_204_BREP_PRODUCT_SCHEMA.GEOMETRIC_REPRESENTATION_ITEM'

       IN TYPEOF (SELF);
END_ENTITY;

ENTITY camera_image_3d_with_scale
  SUBTYPE OF (camera_image);
DERIVE
  scale: positive_ratio_measure := ((SELF\mapped_item.mapping_target\
         planar_extent.size_in_x) / (SELF\mapped_item.mapping_source.
         mapping_origin\camera_model_d3.perspective_of_volume.view_window.
```



```
        size_in_x));
WHERE
  WR1: ('PART_204_BREP_PRODUCT_SCHEMA.CAMERA_MODEL_D3'
       IN TYPEOF (SELF\mapped_item.mapping_source.mapping_origin));
  WR2: aspect_ratio(SELF\mapped_item.mapping_target) =
       aspect_ratio(SELF\mapped_item.mapping_source.mapping_origin\
       camera_model_d3.perspective_of_volume.view_window);
  WR3: SELF\mapped_item.mapping_source.mapping_origin\camera_model_d3.
       perspective_of_volume.front_plane_clipping
       AND
       SELF\mapped_item.mapping_source.mapping_origin\camera_model_d3.
       perspective_of_volume.view_volume_sides_clipping;
  WR4: (SELF\mapped_item.mapping_target\planar_extent.size_in_x > 0)
       AND
       (SELF\mapped_item.mapping_target\planar_extent.size_in_y > 0);
  WR5: (SELF\mapped_item.mapping_source.mapping_origin\camera_model_d3.
       perspective_of_volume.view_window.size_in_x > 0)
       AND
       (SELF\mapped_item.mapping_source.mapping_origin\camera_model_d3.
       perspective_of_volume.view_window.size_in_y > 0);
  WR6: ('PART_204_BREP_PRODUCT_SCHEMA.' +
       'AXIS2_PLACEMENT_2D' IN TYPEOF (SELF\mapped_item.
       mapping_target\planar_box.placement))
       AND NOT ('PART_204_BREP_PRODUCT_SCHEMA.' +

       'AXIS2_PLACEMENT_3D' IN TYPEOF (SELF\mapped_item.
       mapping_target\planar_box.placement));
END_ENTITY;

ENTITY camera_model
  SUPERTYPE OF (ONEOF (
              CAMERA_MODEL_D3))
  SUBTYPE OF (geometric_representation_item);
WHERE
  WR1: (SIZEOF (USEDIN (SELF, 'PART_204_BREP_PRODUCT_SCHEMA.' +
                               'ITEM_DEFINED_TRANSFORMATION.' +
                               'TRANSFORM_ITEM_1')) +
        SIZEOF (USEDIN (SELF, 'PART_204_BREP_PRODUCT_SCHEMA.' +
                               'REPRESENTATION_MAP.MAPPING_ORIGIN'))
       ) > 0;
  WR2: SIZEOF(USEDIN(SELF,'PART_204_BREP_PRODUCT_SCHEMA.'+

                          'STYLED_ITEM.ITEM')) = 0;
END_ENTITY;

ENTITY camera_model_d3
  SUBTYPE OF (camera_model);
  view_reference_system : axis2_placement_3d;
  perspective_of_volume : view_volume;
WHERE
  WR1: (dot_product (SELF.view_reference_system.p[3],
         SELF.perspective_of_volume.view_window.placement.p[3]) = 1.0)
```

```
        AND
      (SELF.view_reference_system.location.coordinates[3] =
       SELF.perspective_of_volume.view_window.
            placement.location.coordinates[3]);
  WR2: SELF\geometric_representation_item.dim = 3;
END_ENTITY;

ENTITY camera_model_d3_with_hlhsr
  SUBTYPE OF (camera_model_d3);
  hidden_line_surface_removal : BOOLEAN;
END_ENTITY;

ENTITY camera_model_with_light_sources
  SUBTYPE OF (camera_model_d3);
  sources : SET [1:?] OF light_source;
END_ENTITY;

ENTITY camera_usage
  SUBTYPE OF (representation_map);
WHERE
  WR1: NOT ('PART_204_BREP_PRODUCT_SCHEMA.PRESENTATION_REPRESENTATION'
       IN TYPEOF(SELF\representation_map.mapped_representation));
  WR2: 'PART_204_BREP_PRODUCT_SCHEMA.CAMERA_MODEL'

       IN TYPEOF (SELF\representation_map.mapping_origin);
END_ENTITY;

ENTITY cartesian_point
 SUBTYPE OF (point);
   coordinates  : LIST [1:3] OF length_measure;
END_ENTITY;

ENTITY cartesian_transformation_operator
   SUPERTYPE OF (ONEOF (
              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, 1.0);
 WHERE
   WR1: scl > 0.0;
END_ENTITY;

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;
 ENTITY circle
   SUBTYPE OF (conic);
   radius   : positive_length_measure;
 END_ENTITY;

ENTITY closed_shell
   SUBTYPE OF (connected_face_set);
END_ENTITY;

ENTITY colour;
END_ENTITY;

ENTITY colour_rgb
  SUBTYPE OF (colour_specification);
  red   : REAL;
  green : REAL;
  blue  : REAL;
WHERE
  WR1: {0.0 <= red <= 1.0};
  WR2: {0.0 <= green <= 1.0};
  WR3: {0.0 <= blue <= 1.0};
END_ENTITY;

ENTITY colour_specification
  SUBTYPE OF (colour);
  name : label;
END_ENTITY;

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;

ENTITY composite_curve_on_surface
  SUBTYPE OF (composite_curve);
 DERIVE
   basis_surface : SET[0:2] OF surface :=
```

```
               get_basis_surface(SELF);
 WHERE
   WR1: SIZEOF(basis_surface) > 0;
   WR2: constraints_composite_curve_on_surface(SELF);
 END_ENTITY;

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 : ('PART_204_BREP_PRODUCT_SCHEMA.BOUNDED_CURVE' IN
                                TYPEOF(parent_curve));
END_ENTITY;

ENTITY  conical_surface
   SUBTYPE OF (elementary_surface);
   radius     : length_measure;
   semi_angle : plane_angle_measure;
 WHERE
   WR1: radius >= 0.0;
 END_ENTITY;

ENTITY conic
   SUPERTYPE OF (ONEOF (
               CIRCLE,
               ELLIPSE,
               HYPERBOLA,
               PARABOLA))
  SUBTYPE OF (curve);
   position: axis2_placement;
END_ENTITY;

ENTITY connected_face_set
   SUPERTYPE OF (ONEOF (
               CLOSED_SHELL,
               OPEN_SHELL))
  SUBTYPE OF (topological_representation_item);
   cfs_faces : SET [1:?] OF face;
END_ENTITY;

ENTITY conversion_based_unit
  SUBTYPE OF (named_unit);
  name              : label;
  conversion_factor : measure_with_unit;
END_ENTITY;

ENTITY curve
   SUPERTYPE OF (ONEOF (
```



```
              LINE,
              CONIC,
              PCURVE,
              SURFACE_CURVE))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY;


ENTITY curve_style;
  name         : label;
  curve_font   : curve_font_or_scaled_curve_font_select;
  curve_width  : size_select;
  curve_colour : colour;
END_ENTITY;

ENTITY curve_style_font;
  name         : label;
  pattern_list : LIST [1:?] OF curve_style_font_pattern;
END_ENTITY;

ENTITY curve_style_font_pattern;
  visible_segment_length   : positive_length_measure;
  invisible_segment_length : positive_length_measure;
END_ENTITY;

ENTITY curve_style_rendering;
  rendering_method     : shading_curve_method;
  rendering_properties : surface_rendering_properties;
END_ENTITY;

ENTITY  cylindrical_surface
   SUBTYPE OF (elementary_surface);
   radius : positive_length_measure;
 END_ENTITY;

ENTITY definitional_representation
  SUBTYPE OF ( representation );
WHERE
  WR1: 'PART_204_BREP_PRODUCT_SCHEMA.PARAMETRIC_REPRESENTATION_CONTEXT'
       IN TYPEOF( SELF\representation.context_of_items );
END_ENTITY;

ENTITY degenerate_toroidal_surface
   SUBTYPE OF (toroidal_surface);
   select_outer : BOOLEAN;
 WHERE
  WR1: major_radius <   minor_radius;
 END_ENTITY;

 ENTITY design_context
  SUBTYPE OF (product_definition_context);
 WHERE
```

```
    WR1 : SELF.life_cycle_stage = 'design';
 END_ENTITY;


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;

 ENTITY direction
   SUBTYPE OF (geometric_representation_item);
   direction_ratios : LIST [2:3] OF REAL;
 WHERE
   WR1: SIZEOF(QUERY(tmp <* direction_ratios | tmp <> 0.0)) > 0;
 END_ENTITY;

ENTITY draughting_pre_defined_colour
  SUBTYPE OF (pre_defined_colour);
WHERE
  WR1: SELF.name IN
      ['red',
       'green',
       'blue',
       'yellow',
       'magenta',
       'cyan',
       'black',
       'white'];
END_ENTITY;

ENTITY draughting_pre_defined_curve_font
  SUBTYPE OF (pre_defined_curve_font);
WHERE
  WR1: SELF.name IN
       ['continuous',
        'chain',
        'chain double dash',
        'dashed',
        'dotted'];
END_ENTITY;

ENTITY edge
   SUPERTYPE OF (ONEOF (
              EDGE_CURVE,
              ORIENTED_EDGE))
  SUBTYPE OF (topological_representation_item);
   edge_start : vertex;
```



```
   edge_end   : vertex;
END_ENTITY;

ENTITY edge_curve
   SUBTYPE OF(edge,geometric_representation_item);
   edge_geometry : curve;
   same_sense    : BOOLEAN;
END_ENTITY;

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;

ENTITY elementary_brep_shape_representation
 SUBTYPE OF (shape_representation);
 WHERE
   WR1 : SIZEOF (QUERY (it <* SELF.items |
        NOT (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP',
                'PART_204_BREP_PRODUCT_SCHEMA.FACETED_BREP',
                'PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM',
                'PART_204_BREP_PRODUCT_SCHEMA.AXIS2_PLACEMENT_3D'] *
                   TYPEOF(it)) = 1))) = 0;
   WR2 : SIZEOF (QUERY (it <* SELF.items |
         SIZEOF(['PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP',
     'PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'] * TYPEOF(it)) =1 )) > 0;
   WR3 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
      'PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
           NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
            NOT (SIZEOF (QUERY(fcs <* csh.cfs_faces |
                NOT('PART_204_BREP_PRODUCT_SCHEMA.FACE_SURFACE' IN
                              TYPEOF(fcs)))) = 0
                 ))) = 0
                   ))) = 0;
   WR4 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
           'PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP'
                     IN TYPEOF(it)) |
           NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
            NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
              NOT(('PART_204_BREP_PRODUCT_SCHEMA.ELEMENTARY_SURFACE' IN
                   TYPEOF(fcs\face_surface.face_geometry))
             ))) = 0
                 ))) = 0
                   ))) = 0;
   WR5 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
           'PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP'
                      IN TYPEOF(it)) |
         NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
```

```
          NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
            NOT (SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
      'PART_204_BREP_PRODUCT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
             NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
                NOT('PART_204_BREP_PRODUCT_SCHEMA.EDGE_CURVE' IN
          TYPEOF(oe.edge_element)))) = 0
                 ))) = 0
               ))) = 0
             ))) = 0
           ))) = 0;
WR6 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
        'PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP'
                          IN TYPEOF(it)) |
       NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
        NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
         NOT (SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
   'PART_204_BREP_PRODUCT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
          NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
                NOT (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.LINE',
                      'PART_204_BREP_PRODUCT_SCHEMA.CONIC',
                      'PART_204_BREP_PRODUCT_SCHEMA.POLYLINE'] *
          TYPEOF(oe.edge_element\edge_curve.edge_geometry)) = 1 )
             )) = 0
             ))) = 0
               ))) = 0
                 ))) = 0
                  ))) = 0;
WR7 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
         'PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP'
                     IN TYPEOF(it)) |
      NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
       NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
        NOT (SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
  'PART_204_BREP_PRODUCT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
         NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
         NOT(('PART_204_BREP_PRODUCT_SCHEMA.VERTEX_POINT'
                  IN TYPEOF(oe.edge_start))
              AND ('PART_204_BREP_PRODUCT_SCHEMA.VERTEX_POINT' IN
               TYPEOF(oe.edge_end))
         ))) = 0
           ))) = 0
            ))) = 0
              ))) = 0
                ))) = 0;
WR8 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
        'PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP'
             IN TYPEOF(it)) |
      NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
       NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
        NOT (SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
  'PART_204_BREP_PRODUCT_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
         NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
```



```
                ('PART_204_BREP_PRODUCT_SCHEMA.POLYLINE' IN
            TYPEOF(oe.edge_element\edge_curve.edge_geometry)) AND
            (NOT (SIZEOF (oe\oriented_edge.edge_element\
                  edge_curve.edge_geometry\polyline.points) >= 3))
               )) = 0
               ))) = 0
                 ))) = 0
                  ))) = 0
                   ))) = 0;
   WR9 : SIZEOF (QUERY (msb <* QUERY (it <* items |
          'PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP'
                    IN TYPEOF(it)) |
          'PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_CLOSED_SHELL'
             IN TYPEOF(msb\manifold_solid_brep.outer))) = 0;
   WR10 : SIZEOF (QUERY (brv <* QUERY (it <* items |
          'PART_204_BREP_PRODUCT_SCHEMA.BREP_WITH_VOIDS'
                       IN TYPEOF(it)) |
          NOT (SIZEOF (QUERY (csh <* brv\brep_with_voids.voids |
           csh\oriented_closed_shell.orientation)) = 0))) = 0;
   WR11 : SIZEOF (QUERY (mi <*  QUERY (it <* items |
                 'PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'
                      IN TYPEOF(it)) |
          NOT (
     'PART_204_BREP_PRODUCT_SCHEMA.ELEMENTARY_BREP_SHAPE_REPRESENTATION'
             IN  TYPEOF(mi\mapped_item.mapping_source.
                          mapped_representation)))) = 0;
   WR12 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
        'PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP'
                       IN TYPEOF(it)) |
          NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
           NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
            NOT (SIZEOF(QUERY (vlp_fbnds <* QUERY (bnds <* fcs.bounds |
         'PART_204_BREP_PRODUCT_SCHEMA.VERTEX_LOOP'
                    IN TYPEOF(bnds.bound)) |
           NOT(('PART_204_BREP_PRODUCT_SCHEMA.VERTEX_POINT' IN
               TYPEOF(vlp_fbnds\face_bound.
                      bound\vertex_loop.loop_vertex)) AND
            ('PART_204_BREP_PRODUCT_SCHEMA.CARTESIAN_POINT' IN
               TYPEOF(vlp_fbnds\face_bound.bound\vertex_loop.
                     loop_vertex\vertex_point.vertex_geometry))
            ))) = 0))) = 0))) = 0))) =0;
END_ENTITY;

ENTITY elementary_surface
   SUPERTYPE OF (ONEOF (
              PLANE,
              CYLINDRICAL_SURFACE,
              CONICAL_SURFACE,
              SPHERICAL_SURFACE,
              TOROIDAL_SURFACE))
  SUBTYPE OF (surface);
   position : axis2_placement_3d;
```

```
END_ENTITY;

ENTITY ellipse
   SUBTYPE OF (conic);
   semi_axis_1 : positive_length_measure;
   semi_axis_2 : positive_length_measure;
 END_ENTITY;

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 |
        'PART_204_BREP_PRODUCT_SCHEMA.FACE_OUTER_BOUND' IN
                                  TYPEOF(temp))) <= 1;
END_ENTITY;

ENTITY face_bound
   SUBTYPE OF(topological_representation_item);
   bound       :  loop;
   orientation :  BOOLEAN;
END_ENTITY;

ENTITY face_outer_bound
    SUBTYPE OF (face_bound);
END_ENTITY;

ENTITY face_surface
   SUBTYPE OF(face,geometric_representation_item);
   face_geometry :  surface;
   same_sense    :  BOOLEAN;
END_ENTITY;

ENTITY faceted_brep
   SUBTYPE OF (manifold_solid_brep);
END_ENTITY;

ENTITY faceted_brep_shape_representation
   SUBTYPE OF (shape_representation);
 WHERE
   WR1 : SIZEOF (QUERY (it <* items |
      NOT (SIZEOF(['PART_204_BREP_PRODUCT_SCHEMA.FACETED_BREP',
            'PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM',
            'PART_204_BREP_PRODUCT_SCHEMA.AXIS2_PLACEMENT_3D'] *
             TYPEOF(it)) = 1))) = 0;
   WR2 : SIZEOF (QUERY (it <* items |
          SIZEOF(['PART_204_BREP_PRODUCT_SCHEMA.FACETED_BREP',
          'PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'] *
```



```
                    TYPEOF(it)) = 1)) > 0;
   WR3 : SIZEOF (QUERY (fbrep <* QUERY ( it <* items |
          'PART_204_BREP_PRODUCT_SCHEMA.FACETED_BREP'
                              IN TYPEOF(it)) |
          NOT (SIZEOF (QUERY (csh <* msb_shells(fbrep) |
   NOT (SIZEOF (QUERY (fcs <* csh\connected_face_set.cfs_faces |
          NOT (('PART_204_BREP_PRODUCT_SCHEMA.FACE_SURFACE'
                        IN TYPEOF (fcs)) AND
          (('PART_204_BREP_PRODUCT_SCHEMA.PLANE' IN TYPEOF
               (fcs\face_surface.face_geometry)) AND
          ('PART_204_BREP_PRODUCT_SCHEMA.CARTESIAN_POINT'
             IN TYPEOF (fcs\face_surface.face_geometry\
                  elementary_surface.position.location))))))
                      = 0))) = 0))) = 0;
   WR4 : SIZEOF (QUERY (fbrep <* QUERY ( it <* items |
          'PART_204_BREP_PRODUCT_SCHEMA.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 |
      'PART_204_BREP_PRODUCT_SCHEMA.FACE_OUTER_BOUND' IN TYPEOF(bnds)))
              = 1))) = 0))) = 0))) = 0;
   WR5 : SIZEOF (QUERY (msb <* QUERY (it <* items |
          'PART_204_BREP_PRODUCT_SCHEMA.MANIFOLD_SOLID_BREP'
                         IN TYPEOF(it)) |
          'PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_CLOSED_SHELL' IN
             TYPEOF (msb\manifold_solid_brep.outer))) = 0;
   WR6 : SIZEOF (QUERY (brv <* QUERY (it <* items |
          'PART_204_BREP_PRODUCT_SCHEMA.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 |
            'PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM' IN TYPEOF(it)) |
  NOT ('PART_204_BREP_PRODUCT_SCHEMA.FACETED_BREP_SHAPE_REPRESENTATION'
     IN  TYPEOF(mi\mapped_item.mapping_source.
                         mapped_representation)))) = 0;
END_ENTITY;

ENTITY fill_area_style;
  name       : label;
  fill_styles : SET [1:?] OF fill_style_select;
WHERE
  WR1: SIZEOF(QUERY(fill_style <* SELF.fill_styles |
         'PART_204_BREP_PRODUCT_SCHEMA.'+
         'FILL_AREA_STYLE_COLOUR' IN
         TYPEOF(fill_style)
         )) <= 1;
END_ENTITY;

ENTITY fill_area_style_colour;
  name        : label;
  fill_colour : colour;
```



```
END_ENTITY;

ENTITY founded_item;
END_ENTITY;

ENTITY functionally_defined_transformation;
  name        : label;
  description : text;
END_ENTITY;

ENTITY geometric_representation_context
  SUBTYPE OF (representation_context);
  coordinate_space_dimension : dimension_count;
END_ENTITY;



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))
  SUBTYPE OF (representation_item);
DERIVE
  dim : dimension_count := dimension_of(SELF);
WHERE
  WR1: SIZEOF (QUERY (using_rep <* using_representations (SELF) |
       NOT('PART_204_BREP_PRODUCT_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT'
             IN TYPEOF (using_rep.context_of_items)))) = 0;
END_ENTITY;



ENTITY global_unit_assigned_context
  SUBTYPE OF (representation_context);
  units : SET [1:?] OF unit;
END_ENTITY;

ENTITY hyperbola
   SUBTYPE OF (conic);
   semi_axis      : positive_length_measure;
   semi_imag_axis : positive_length_measure;
 END_ENTITY;

ENTITY invisibility;
```



```
  invisible_items : SET [1:?] OF invisible_item;
END_ENTITY;

ENTITY light_source
  SUPERTYPE OF (ONEOF (
              LIGHT_SOURCE_AMBIENT,
              LIGHT_SOURCE_DIRECTIONAL,
              LIGHT_SOURCE_POSITIONAL,
              LIGHT_SOURCE_SPOT))
  SUBTYPE OF (geometric_representation_item);
  light_colour : colour;
WHERE
  WR1: SIZEOF(USEDIN(SELF,'PART_204_BREP_PRODUCT_SCHEMA.'+

                         'STYLED_ITEM.ITEM')) = 0;
END_ENTITY;

ENTITY light_source_ambient
  SUBTYPE OF (light_source);
END_ENTITY;

ENTITY light_source_directional
  SUBTYPE OF (light_source);
  orientation : direction;
END_ENTITY;

ENTITY light_source_positional
  SUBTYPE OF (light_source);
  position             : cartesian_point;
  constant_attenuation : REAL;
  distance_attenuation : REAL;
END_ENTITY;

ENTITY light_source_spot
  SUBTYPE OF (light_source);
  position              : cartesian_point;
  orientation           : direction;
  concentration_exponent : REAL;
  constant_attenuation   : REAL;
  distance_attenuation   : REAL;
  spread_angle          : positive_plane_angle_measure;
END_ENTITY;

ENTITY line
   SUBTYPE OF (curve);
   pnt : cartesian_point;
   dir : vector;
 WHERE
   WR1: dir.dim  = pnt.dim;
 END_ENTITY;

ENTITY loop
```

```
   SUPERTYPE OF (ONEOF (
              VERTEX_LOOP,
              EDGE_LOOP,
              POLY_LOOP))
  SUBTYPE OF (topological_representation_item);
END_ENTITY;

ENTITY manifold_solid_brep
   SUBTYPE OF (solid_model);
   outer : closed_shell;
END_ENTITY;


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;

ENTITY measure_with_unit;
  value_component : measure_value;
  unit_component  : unit;
WHERE
  WR1: valid_units (SELF);
END_ENTITY;


ENTITY mechanical_context
   SUBTYPE OF (product_context);
   WHERE
     WR1 : SELF.discipline_type = 'mechanical';
 END_ENTITY;

ENTITY mechanical_design_geometric_presentation_area
  SUBTYPE OF (presentation_area);
WHERE
  WR1:  -- only presentation_views or axis2_placements in
        -- mechanical_design_geometric_presentation_area
        SIZEOF(QUERY(it1 <* SELF.items |
        NOT (('PART_204_BREP_PRODUCT_SCHEMA.AXIS2_PLACEMENT'
        IN TYPEOF(it1))
        OR
        (('PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'
        IN TYPEOF(it1)) AND
        ('PART_204_BREP_PRODUCT_SCHEMA.PRESENTATION_VIEW'
        IN TYPEOF
   (it1\mapped_item.mapping_source.mapped_representation))))))) = 0;
  WR2: -- only mechanical_design_geometric_presentation_representation
       -- via camera_image_3d_with_scale or axis2_placements in
       -- presentation_views
```



```
      SIZEOF(QUERY(pv <* QUERY(mi1 <* QUERY(it1 <* SELF.items |
      'PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'
      IN TYPEOF(it1)) |
      'PART_204_BREP_PRODUCT_SCHEMA.PRESENTATION_VIEW'
      IN TYPEOF
      (mi1\mapped_item.mapping_source.mapped_representation)) |
  -- search in all presentation_views for axis2_placements and
  -- mapped_items and for the subtype of mapped_item
  -- camera_image_3d_with_scale; the latter shall reference
  -- a mechanical_design_geometric_presentation_representation;
  -- the supertype mapped_item shall reference presentation_view.
      NOT (SIZEOF(QUERY(it2 <* pv\mapped_item.mapping_source.
      mapped_representation\representation.items |
      NOT (('PART_204_BREP_PRODUCT_SCHEMA.AXIS2_PLACEMENT'
      IN TYPEOF(it2))
      OR
      (('PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'
      IN TYPEOF(it2)) AND NOT
      ('PART_204_BREP_PRODUCT_SCHEMA.' +
      'CAMERA_IMAGE_3D_WITH_SCALE' IN TYPEOF(it2))) AND NOT (
      'PART_204_BREP_PRODUCT_SCHEMA.PRESENTATION_VIEW'
      IN TYPEOF
      (it2\mapped_item.mapping_source.mapped_representation)))
      OR
      (('PART_204_BREP_PRODUCT_SCHEMA.' +
      'CAMERA_IMAGE_3D_WITH_SCALE' IN TYPEOF(it2))
      AND NOT (
      ('PART_204_BREP_PRODUCT_SCHEMA.' +
      'MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_REPRESENTATION'
      IN TYPEOF
          (it2\mapped_item.mapping_source.mapped_representation) ))
      ))) = 0))) = 0;
WR3:  (SIZEOF(QUERY(ps <* USEDIN (SELF\presentation_area,
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'PRESENTATION_SIZE.UNIT') | ((ps.size\planar_extent.
                size_in_x <= 0)
      OR
      (ps.size\planar_extent.size_in_y <= 0)))) = 0)
      AND
      (SIZEOF(QUERY(ais <* USEDIN (SELF\presentation_area,
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'AREA_IN_SET.AREA') |
      (SIZEOF(QUERY(ps <* USEDIN (ais,
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'PRESENTATION_SIZE.UNIT') |
      ((ps.size\planar_extent.size_in_x <= 0)
      OR
      (ps.size\planar_extent.size_in_y <= 0)))) > 0))) = 0);
WR4:  (SIZEOF(QUERY(ps <* USEDIN (SELF\presentation_area,
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'PRESENTATION_SIZE.UNIT') |
      ('PART_204_BREP_PRODUCT_SCHEMA.' +
```

```
        'AXIS2_PLACEMENT_2D' IN TYPEOF (ps.size.placement)))) = 1)
        AND
        (SIZEOF(QUERY(ps <* USEDIN (SELF\presentation_area,
        'PART_204_BREP_PRODUCT_SCHEMA.' +
        'PRESENTATION_SIZE.UNIT') |
        ('PART_204_BREP_PRODUCT_SCHEMA.' +
        'AXIS2_PLACEMENT_3D' IN TYPEOF (ps.size.placement)))) = 0)
        OR
        ((SIZEOF(QUERY(ais <* USEDIN (SELF\presentation_area,
        'PART_204_BREP_PRODUCT_SCHEMA.' +
        'AREA_IN_SET.AREA') |
        (SIZEOF(QUERY(ps <* USEDIN (ais,
        'PART_204_BREP_PRODUCT_SCHEMA.' +
        'PRESENTATION_SIZE.UNIT') |
        ('PART_204_BREP_PRODUCT_SCHEMA.' +
        'AXIS2_PLACEMENT_2D' IN TYPEOF
                    (ps.size.placement)))) = 1))) = 1)
        AND
        (SIZEOF(QUERY(ais <* USEDIN (SELF\presentation_area,
        'PART_204_BREP_PRODUCT_SCHEMA.' +
        'AREA_IN_SET.AREA') |
        (SIZEOF(QUERY(ps <* USEDIN (ais,
        'PART_204_BREP_PRODUCT_SCHEMA.' +
        'PRESENTATION_SIZE.UNIT') |
        ('PART_204_BREP_PRODUCT_SCHEMA.' +
        'AXIS2_PLACEMENT_3D' IN
            TYPEOF (ps.size.placement)))) = 0))) = 1));
END_ENTITY;


ENTITY mechanical_design_geometric_presentation_representation
  SUBTYPE OF (representation);
WHERE
  WR1:  SIZEOF(QUERY(it <* SELF.items |
        NOT (SIZEOF(
        ['PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM',
        'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM',
        'PART_204_BREP_PRODUCT_SCHEMA.AXIS2_PLACEMENT',
        'PART_204_BREP_PRODUCT_SCHEMA.CAMERA_MODEL_D3']
        * TYPEOF(it)) = 1))) = 0;
  WR2:  -- only shape_representations and
  -- mechanical_design_geometric_presentation_representations
 -- shall be referenced from mapped_items
        SIZEOF(QUERY(mi <* QUERY(it <* SELF.items |
        ('PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'
        IN TYPEOF(it))) | NOT (SIZEOF(
        ['PART_204_BREP_PRODUCT_SCHEMA.' +
        'SHAPE_REPRESENTATION',
        'PART_204_BREP_PRODUCT_SCHEMA.' +
    'MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_REPRESENTATION']
   * TYPEOF(mi\mapped_item.mapping_source.mapped_representation))
        = 1))) = 0;
```



```
 WR3:  -- a mapped_item that is styled shall reference a
       -- shape_representation
       SIZEOF(QUERY(smi <* QUERY(si <* QUERY(it <* SELF.items |
       ('PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
       IN TYPEOF(it))) |
       ('PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'
       IN TYPEOF(si\styled_item.item))) | NOT (
       ('PART_204_BREP_PRODUCT_SCHEMA.' +
       'SHAPE_REPRESENTATION' IN TYPEOF (smi\styled_item.
item\mapped_item.mapping_source.mapped_representation))) )) = 0;
 WR4:  SIZEOF(QUERY(si <* QUERY(it <* SELF.items |
       'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
       IN TYPEOF(it)) | NOT
            (SIZEOF(QUERY(psa <* si\styled_item.styles |
       NOT (SIZEOF(QUERY(pss <* psa.styles | NOT (SIZEOF(
       ['PART_204_BREP_PRODUCT_SCHEMA.POINT_STYLE',
       'PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE',
       'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE']
       * TYPEOF(pss)) = 1))) = 0))) = 0))) = 0;
 WR5:  SIZEOF(QUERY(si <* QUERY(it <* SELF.items |
       'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
       IN TYPEOF(it)) |
       NOT
       (SIZEOF(QUERY(psbc <* QUERY(psa <* si\styled_item.styles |
       'PART_204_BREP_PRODUCT_SCHEMA.' +
       'PRESENTATION_STYLE_BY_CONTEXT' IN TYPEOF(psa)) |
        NOT (SIZEOF(
       ['PART_204_BREP_PRODUCT_SCHEMA.' +
       'REPRESENTATION_ITEM',
       'PART_204_BREP_PRODUCT_SCHEMA.REPRESENTATION']
       * TYPEOF(psbc\presentation_style_by_context.style_context))
       = 1))) = 0))) = 0;
 WR6:  SIZEOF(QUERY(si <* QUERY(it <* SELF.items |
       'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
       IN TYPEOF(it)) |
         NOT (SIZEOF(QUERY(psa <* si\styled_item.styles |
       NOT (SIZEOF(QUERY(ps <* QUERY(pss <* psa.styles |
       'PART_204_BREP_PRODUCT_SCHEMA.POINT_STYLE'
       IN TYPEOF(pss)) | NOT
       (('PART_204_BREP_PRODUCT_SCHEMA.' +
       'POSITIVE_LENGTH_MEASURE' IN
            TYPEOF (ps\point_style.marker_size))
       AND (SIZEOF(
       ['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
       'PART_204_BREP_PRODUCT_SCHEMA.' +
       'DRAUGHTING_PRE_DEFINED_COLOUR']
       * TYPEOF(ps\point_style.marker_colour))
       = 1)))) = 0))) = 0))) = 0;
 WR7:  SIZEOF(QUERY(si <* QUERY(it <* SELF.items |
       'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
       IN TYPEOF(it)) | NOT
           (SIZEOF(QUERY(psa <* si\styled_item.styles |
```

```
      NOT (SIZEOF(QUERY(cs <* QUERY(pss <* psa.styles |
      'PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE'
      IN TYPEOF(pss)) | NOT((SIZEOF(
      ['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'DRAUGHTING_PRE_DEFINED_COLOUR']
      * TYPEOF(cs\curve_style.curve_colour)) = 1)
      AND
      ('PART_204_BREP_PRODUCT_SCHEMA.' +
      'POSITIVE_LENGTH_MEASURE' IN
           TYPEOF (cs\curve_style.curve_width))
      AND (SIZEOF(
      ['PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE_FONT',
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'DRAUGHTING_PRE_DEFINED_CURVE_FONT']
      * TYPEOF(cs\curve_style.curve_font)) = 1))))
                = 0))) = 0))) = 0;
WR8:  SIZEOF(QUERY(si <* QUERY(it <* SELF.items |
      'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
      IN TYPEOF(it)) | NOT
           (SIZEOF(QUERY(psa <* si\styled_item.styles |
      NOT (SIZEOF(QUERY(ssu <* QUERY(pss <* psa.styles |
      'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
      IN TYPEOF(pss)) |
      NOT ('PART_204_BREP_PRODUCT_SCHEMA.' +
      'SURFACE_SIDE_STYLE' IN TYPEOF
      (ssu\surface_style_usage.style)))) = 0))) = 0))) = 0;
WR9:  SIZEOF(QUERY(si <* QUERY(it <* SELF.items |
      'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
      IN TYPEOF(it)) | NOT
           (SIZEOF(QUERY(psa <* si\styled_item.styles |
    NOT (SIZEOF(QUERY(ssu <* QUERY(pss <* psa.styles |
    'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
    IN TYPEOF(pss)) | NOT (SIZEOF(QUERY(sses <*
    ssu\surface_style_usage.style\surface_side_style.styles |
      NOT (SIZEOF(
      ['PART_204_BREP_PRODUCT_SCHEMA.' +
      'SURFACE_STYLE_PARAMETER_LINE',
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'SURFACE_STYLE_CONTROL_GRID',
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'SURFACE_STYLE_SILHOUETTE',
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'SURFACE_STYLE_SEGMENTATION_CURVE',
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'SURFACE_STYLE_FILL_AREA',
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'SURFACE_STYLE_BOUNDARY']
      * TYPEOF(sses)) = 1))) = 0))) = 0))) = 0))) = 0;
WR10: SIZEOF(QUERY(si <* QUERY(it <* SELF.items |
      'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
      IN TYPEOF(it)) | NOT
```



```
        (SIZEOF(QUERY(psa <* si\styled_item.styles |
      NOT (SIZEOF(QUERY(ssu <* QUERY(pss <* psa.styles |
      'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
      IN TYPEOF(pss)) | NOT (SIZEOF(QUERY(sspl <* QUERY(sses <*
      ssu\surface_style_usage.style\surface_side_style.styles |
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'SURFACE_STYLE_PARAMETER_LINE' IN TYPEOF(sses)) |
      NOT (('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE'
      IN TYPEOF
    (sspl\surface_style_parameter_line.style_of_parameter_lines))
      AND (SIZEOF(
      ['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'DRAUGHTING_PRE_DEFINED_COLOUR']
      * TYPEOF(sspl\surface_style_parameter_line.
      style_of_parameter_lines\curve_style.curve_colour)) = 1)
      AND (
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'POSITIVE_LENGTH_MEASURE' IN TYPEOF
      (sspl\surface_style_parameter_line.
      style_of_parameter_lines\curve_style.curve_width))
      AND (SIZEOF(
      ['PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE_FONT',
      'PART_204_BREP_PRODUCT_SCHEMA.'+
      'DRAUGHTING_PRE_DEFINED_CURVE_FONT']
      * TYPEOF(sspl\surface_style_parameter_line.
      style_of_parameter_lines\curve_style.curve_font)) = 1))))
      = 0))) = 0))) = 0))) = 0;
WR11: SIZEOF(QUERY(si <* QUERY(it <* SELF.items |
      'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
      IN TYPEOF(it)) | NOT
            (SIZEOF(QUERY(psa <* si\styled_item.styles |
      NOT (SIZEOF(QUERY(ssu <* QUERY(pss <* psa.styles |
      'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
      IN TYPEOF(pss)) | NOT (SIZEOF(QUERY(sscg <* QUERY(sses <*
      ssu\surface_style_usage.style\surface_side_style.styles |
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'SURFACE_STYLE_CONTROL_GRID' IN TYPEOF(sses)) |
      NOT (('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE'
      IN TYPEOF
        (sscg\surface_style_control_grid.style_of_control_grid))
      AND (SIZEOF(
      ['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'DRAUGHTING_PRE_DEFINED_COLOUR']
      * TYPEOF(sscg\surface_style_control_grid.
      style_of_control_grid\curve_style.curve_colour)) = 1)
      AND
      ('PART_204_BREP_PRODUCT_SCHEMA.' +
      'POSITIVE_LENGTH_MEASURE' IN TYPEOF
      (sscg\surface_style_control_grid.
      style_of_control_grid\curve_style.curve_width))
```

```
      AND (SIZEOF(
      ['PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE_FONT',
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'DRAUGHTING_PRE_DEFINED_CURVE_FONT']
      * TYPEOF(sscg\surface_style_control_grid.
      style_of_control_grid\curve_style.curve_font)) = 1))))
      = 0))) = 0))) = 0))) = 0;
WR12: SIZEOF(QUERY(si <* QUERY(it <* SELF.items |
      'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
      IN TYPEOF(it)) |
      NOT (SIZEOF(QUERY(psa <* si\styled_item.styles |
      NOT (SIZEOF(QUERY(ssu <* QUERY(pss <* psa.styles |
      'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
      IN TYPEOF(pss)) | NOT (SIZEOF(QUERY(sssh <* QUERY(sses <*
      ssu\surface_style_usage.style\surface_side_style.styles |
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'SURFACE_STYLE_SILHOUETTE' IN TYPEOF(sses)) |
      NOT (('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE'
      IN TYPEOF
        (sssh\surface_style_silhouette.style_of_silhouette))
      AND (SIZEOF(
      ['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'DRAUGHTING_PRE_DEFINED_COLOUR']
      * TYPEOF(sssh\surface_style_silhouette.
      style_of_silhouette\curve_style.curve_colour)) = 1)
      AND
      ('PART_204_BREP_PRODUCT_SCHEMA.' +
      'POSITIVE_LENGTH_MEASURE' IN TYPEOF
      (sssh\surface_style_silhouette.style_of_silhouette\curve_style.
      curve_width))
      AND (SIZEOF(
      ['PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE_FONT',
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'DRAUGHTING_PRE_DEFINED_CURVE_FONT']
      * TYPEOF(sssh\surface_style_silhouette.
      style_of_silhouette\curve_style.curve_font)) = 1))))
      = 0))) = 0))) = 0))) = 0;
WR13: SIZEOF(QUERY(si <* QUERY(it <* SELF.items |
      'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
      IN TYPEOF(it)) | NOT
          (SIZEOF(QUERY(psa <* si\styled_item.styles |
      NOT (SIZEOF(QUERY(ssu <* QUERY(pss <* psa.styles |
      'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
      IN TYPEOF(pss)) | NOT (SIZEOF(QUERY(sssc <* QUERY(sses <*
      ssu\surface_style_usage.style\surface_side_style.styles |
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'SURFACE_STYLE_SEGMENTATION_CURVE' IN TYPEOF(sses)) |
      NOT (('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE'
      IN TYPEOF
      (sssc\surface_style_segmentation_curve.
                    style_of_segmentation_curve))
```



```
        AND (SIZEOF(
        ['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
        'PART_204_BREP_PRODUCT_SCHEMA.' +
        'DRAUGHTING_PRE_DEFINED_COLOUR']
        * TYPEOF(sssc\surface_style_segmentation_curve.
        style_of_segmentation_curve\curve_style.curve_colour)) = 1)
        AND
        ('PART_204_BREP_PRODUCT_SCHEMA.' +
        'POSITIVE_LENGTH_MEASURE' IN TYPEOF
        (sssc\surface_style_segmentation_curve.
        style_of_segmentation_curve\curve_style.curve_width))
        AND (SIZEOF(
        ['PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE_FONT',
        'PART_204_BREP_PRODUCT_SCHEMA.' +
        'DRAUGHTING_PRE_DEFINED_CURVE_FONT']
        * TYPEOF(sssc\surface_style_segmentation_curve.
        style_of_segmentation_curve\curve_style.curve_font)) = 1))))
        = 0))) = 0))) = 0))) = 0;
  WR14: SIZEOF(QUERY(si <* QUERY(it <* SELF.items |
        'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
        IN TYPEOF(it)) | NOT
            (SIZEOF(QUERY(psa <* si\styled_item.styles |
        NOT (SIZEOF(QUERY(ssu <* QUERY(pss <* psa.styles |
        'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
        IN TYPEOF(pss)) | NOT (SIZEOF(QUERY(ssbd <* QUERY(sses <*
        ssu\surface_style_usage.style\surface_side_style.styles |
        'PART_204_BREP_PRODUCT_SCHEMA.' +
        'SURFACE_STYLE_BOUNDARY' IN TYPEOF(sses)) |
        NOT (('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE'
        IN TYPEOF (ssbd\surface_style_boundary.style_of_boundary))
        AND (SIZEOF(
        ['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
        'PART_204_BREP_PRODUCT_SCHEMA.' +
        'DRAUGHTING_PRE_DEFINED_COLOUR']
        * TYPEOF(ssbd\surface_style_boundary.
        style_of_boundary\curve_style.curve_colour)) = 1)
        AND
        ('PART_204_BREP_PRODUCT_SCHEMA.' +
        'POSITIVE_LENGTH_MEASURE' IN
              TYPEOF (ssbd\surface_style_boundary.
        style_of_boundary\curve_style.curve_width))
        AND (SIZEOF(
        ['PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE_FONT',
        'PART_204_BREP_PRODUCT_SCHEMA.' +

        'DRAUGHTING_PRE_DEFINED_CURVE_FONT']
        * TYPEOF(ssbd\surface_style_boundary.
        style_of_boundary\curve_style.curve_font)) = 1)))) = 0)))
        = 0))) = 0))) = 0;
END_ENTITY;

ENTITY mechanical_design_shaded_presentation_area
```

```
  SUBTYPE OF (presentation_area);
WHERE
WR1 : (* only presentation_views or axis2_placements in
         mechanical_design_shaded_presentation_area *)
      SIZEOF (QUERY (it1 <* SELF.items |
      NOT (('PART_204_BREP_PRODUCT_SCHEMA.AXIS2_PLACEMENT'
      IN TYPEOF (it1))
      OR
      (('PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'
      IN TYPEOF (it1)) AND
      ('PART_204_BREP_PRODUCT_SCHEMA.PRESENTATION_VIEW'
      IN TYPEOF
   (it1\mapped_item.mapping_source.mapped_representation)))))) = 0;
WR2 : (* only mechanical_design_shaded_presentation_representation
         via camera_image_3d_with_scale or axis2_placements in
         presentation_views *)
    SIZEOF (QUERY (pv <* QUERY (mi1 <* QUERY (it1 <* SELF.items |
      'PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'
      IN TYPEOF (it1)) |
      'PART_204_BREP_PRODUCT_SCHEMA.PRESENTATION_VIEW'
      IN TYPEOF
      (mi1\mapped_item.mapping_source.mapped_representation)) |
  (* search in all presentation_views for axis2_placements and
       mapped_items and for the subtype of mapped_item,
       camera_image_3d_with_scale; the latter shall reference
       a mechanical_design_geometric_presentation_representation;
     the supertype mapped_item shall reference presentation_view. *)
      NOT (SIZEOF(QUERY(it2 <* pv\mapped_item.mapping_source.
      mapped_representation\presentation_view.items |
      NOT (('PART_204_BREP_PRODUCT_SCHEMA.AXIS2_PLACEMENT'
      IN TYPEOF(it2))
      OR
      (('PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'
      IN TYPEOF(it2)) AND NOT
      ('PART_204_BREP_PRODUCT_SCHEMA.' +
      'CAMERA_IMAGE_3D_WITH_SCALE' IN TYPEOF(it2))) AND NOT (
      'PART_204_BREP_PRODUCT_SCHEMA.PRESENTATION_VIEW'
      IN TYPEOF
      (it2\mapped_item.mapping_source.mapped_representation)))
      OR
      (('PART_204_BREP_PRODUCT_SCHEMA.' +
      'CAMERA_IMAGE_3D_WITH_SCALE' IN TYPEOF(it2))
      AND NOT (
      ('PART_204_BREP_PRODUCT_SCHEMA.' +
      'MECHANICAL_DESIGN_SHADED_PRESENTATION_REPRESENTATION'
      IN TYPEOF
        (it2\mapped_item.mapping_source.mapped_representation) ))
          ))) = 0))) = 0;
WR3: (* Presentation_size shall be a positive rectangle for area and set.
        Check for this presentation_area subtype first. *)
      (SIZEOF (QUERY(ps <* USEDIN (SELF,
      'PART_204_BREP_PRODUCT_SCHEMA.' +
```



```
    'PRESENTATION_SIZE.UNIT') |
        NOT ((ps.size\planar_extent.size_in_x > 0)
      AND (ps.size\planar_extent.size_in_y > 0)) )) = 0)
      AND
      (* check secondly for presentation_set, via area_in_set *)
      (SIZEOF (QUERY(pset <* QUERY(ais <*
      USEDIN (SELF, 'PART_204_BREP_PRODUCT_SCHEMA.' +
      'AREA_IN_SET.AREA')
      | 'PART_204_BREP_PRODUCT_SCHEMA.' +
      'PRESENTATION_SET' IN TYPEOF (ais.in_set)) |
 (* after having collected all presentation_set, check their sizes *)
      SIZEOF (QUERY(psize <* USEDIN(pset,
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'PRESENTATION_SIZE.UNIT')
      | NOT ((psize.size\planar_extent.size_in_x > 0)
      AND (psize.size\planar_extent.size_in_y > 0)) )) = 0)) = 0);
WR4:  (* Drawing space shall be defined in 2D.
         Check for this presentation_area subtype first. *)
      (SIZEOF(QUERY( psize <* USEDIN (SELF,
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'PRESENTATION_SIZE.UNIT')
      | 'PART_204_BREP_PRODUCT_SCHEMA.' +
      'AXIS2_PLACEMENT_2D'
      IN TYPEOF (psize.size.placement))) = 1)
      AND
      (* check secondly for presentation_set, via area_in_set *)
      (SIZEOF (QUERY(pset <* QUERY(ais <*
      USEDIN (SELF, 'PART_204_BREP_PRODUCT_SCHEMA.' +
      'AREA_IN_SET.AREA')
      | 'PART_204_BREP_PRODUCT_SCHEMA.' +
      'PRESENTATION_SET' IN TYPEOF (ais.in_set)) |
  (* after having collected all presentation_set, check their
         dimension *)
      SIZEOF (QUERY(psize <* USEDIN(pset,
      'PART_204_BREP_PRODUCT_SCHEMA.' +
      'PRESENTATION_SIZE.UNIT')
      | NOT ('PART_204_BREP_PRODUCT_SCHEMA.' +
      'AXIS2_PLACEMENT_2D'
      IN TYPEOF (psize.size.placement)) )) = 0)) = 0);
WR5 : (* valid types of camera_models
      get for all presentation_areas their presentation_views *)
    SIZEOF (QUERY (pv <* QUERY (mi1 <* QUERY (it1 <* SELF.items |
      'PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'
      IN TYPEOF (it1)) |
      'PART_204_BREP_PRODUCT_SCHEMA.PRESENTATION_VIEW'
      IN TYPEOF
      (mi1\mapped_item.mapping_source.mapped_representation)) |
     (* search in all presentation_views for
        mapped_items and for the subtype of mapped_item,
        camera_image_3d_with_scale; the latter shall reference
        a camera_usage that shall have as its mapping_origin either
        camera_model_d3, camera_model_d3_with_hlhsr, or
```

```
          camera_model_with_light_sources. *)
        NOT (SIZEOF(QUERY(ci <* pv\mapped_item.mapping_source.
        mapped_representation\presentation_view.items |
        ('PART_204_BREP_PRODUCT_SCHEMA.'+
         'CAMERA_IMAGE_3D_WITH_SCALE' IN TYPEOF(ci))
        AND
        (SIZEOF(['PART_204_BREP_PRODUCT_SCHEMA.'+
        'CAMERA_MODEL_D3',
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'CAMERA_MODEL_D3_WITH_HLHSR',
        'PART_204_BREP_PRODUCT_SCHEMA.'+

        'CAMERA_MODEL_WITH_LIGHT_SOURCES'] * TYPEOF
     (ci\camera_image_3d_with_scale.mapping_source.mapping_origin))
        = 1))) = 0))) = 0;
END_ENTITY; (* mechanical_design_shaded_presentation_area *)


ENTITY mechanical_design_shaded_presentation_representation
   SUBTYPE OF (representation);
WHERE
  WR1:  SIZEOF(QUERY(it <* SELF.items |
        NOT (SIZEOF(
        ['PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM',
        'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM',
        'PART_204_BREP_PRODUCT_SCHEMA.AXIS2_PLACEMENT',
        'PART_204_BREP_PRODUCT_SCHEMA.CAMERA_MODEL_D3']
        * TYPEOF(it)) = 1))) = 0;
  WR2:  (* for all mapped_items check that only
           shape_representations and
           mechanical_design_shaded_presentation_representations
           are referenced *)
        SIZEOF(QUERY(mi <* QUERY(it <* SELF.items |
        ('PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'
        IN TYPEOF(it))) | NOT (SIZEOF(
        ['PART_204_BREP_PRODUCT_SCHEMA.' +
        'SHAPE_REPRESENTATION',
        'PART_204_BREP_PRODUCT_SCHEMA.' +
        'MECHANICAL_DESIGN_SHADED_PRESENTATION_REPRESENTATION']
      * TYPEOF(mi\mapped_item.mapping_source.mapped_representation))
        = 1))) = 0;
  WR3:  (* for all styled_item.item check that in case they are
           mapped_items that they are shape_representations *)
        SIZEOF(QUERY(smi <* QUERY(si <* QUERY(it <* SELF.items |
        ('PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
        IN TYPEOF(it))) |
        ('PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM'
        IN TYPEOF(si\styled_item.item))) | NOT (
        ('PART_204_BREP_PRODUCT_SCHEMA.' +
        'SHAPE_REPRESENTATION' IN TYPEOF (smi\styled_item.
     item\mapped_item.mapping_source.mapped_representation))) )) = 0;
  WR4 : (* for all styled_items get their styles via
```



```
         presentation_style_assignment.styles and check for valid
         style types *)
       SIZEOF (QUERY (si <* QUERY (it <* SELF.items |
       'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
       IN TYPEOF (it)) |
       NOT (SIZEOF (QUERY (psa <* si\styled_item.styles |
       NOT (SIZEOF (QUERY (pss <* psa.styles |
       NOT (SIZEOF (
       ['PART_204_BREP_PRODUCT_SCHEMA.POINT_STYLE',
       'PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE',
       'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE']
       *  TYPEOF (pss)) = 1))) = 0))) = 0))) = 0;
WR5 : (* for all styled_items get those assigned styles that
         are presentation_style_by_contexts and ensure that
         these reference only representation_items and
         representations as valid contexts *)
       SIZEOF (QUERY (si <* QUERY (it <* SELF.items |
       'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
       IN TYPEOF (it)) |
       NOT (SIZEOF (QUERY (psbc <*
             QUERY (psa <* si\styled_item.styles |
       'PART_204_BREP_PRODUCT_SCHEMA.'+
       'PRESENTATION_STYLE_BY_CONTEXT' IN TYPEOF (psa))  |
       NOT (SIZEOF (
       ['PART_204_BREP_PRODUCT_SCHEMA.REPRESENTATION_ITEM',
       'PART_204_BREP_PRODUCT_SCHEMA.REPRESENTATION']
       * TYPEOF
       (psbc\presentation_style_by_context.style_context)) = 1)))
            = 0))) = 0;
WR6 : (* for all styled_items get all assigned point_styles
         and ensure that marker_select, marker_size and
         marker_colour are valid *)
       SIZEOF (QUERY (si <* QUERY (it <* SELF.items |
       'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
       IN TYPEOF (it)) |
       NOT (SIZEOF (QUERY (psa <* si\styled_item.styles |
       NOT (SIZEOF (QUERY (ps <* QUERY (pss <* psa.styles |
       'PART_204_BREP_PRODUCT_SCHEMA.POINT_STYLE'
       IN TYPEOF (pss)) |
       NOT (
       ('PART_204_BREP_PRODUCT_SCHEMA.MARKER_TYPE'
       IN TYPEOF (ps\point_style.marker))
       AND
       ('PART_204_BREP_PRODUCT_SCHEMA.POSITIVE_LENGTH_MEASURE'
        IN TYPEOF(ps\point_style.marker_size))
       AND
       (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
       'PART_204_BREP_PRODUCT_SCHEMA.'+
       'DRAUGHTING_PRE_DEFINED_COLOUR'] * TYPEOF
       (ps\point_style.marker_colour)) = 1)))) = 0))) = 0))) = 0;
WR7 : (* for all styled_items get all assigned curve_styles
         and ensure that curve_width, curve_font and
```

```
         curve_colour are valid *)
       SIZEOF (QUERY (si <* QUERY (it <* SELF.items |
       'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
       IN TYPEOF (it)) |
       NOT (SIZEOF (QUERY (psa <* si\styled_item.styles |
       NOT (SIZEOF (QUERY (cs <* QUERY (pss <* psa.styles |
       'PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE'
       IN TYPEOF (pss)) |
       NOT (
       (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
       'PART_204_BREP_PRODUCT_SCHEMA.'+
       'DRAUGHTING_PRE_DEFINED_COLOUR'] * TYPEOF
       (cs\curve_style.curve_colour)) = 1)
       AND
       ('PART_204_BREP_PRODUCT_SCHEMA.POSITIVE_LENGTH_MEASURE'
         IN TYPEOF(cs\curve_style.curve_width))
       AND
       (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.'+
       'CURVE_STYLE_FONT', 'PART_204_BREP_PRODUCT_SCHEMA.'+
       'DRAUGHTING_PRE_DEFINED_CURVE_FONT'] * TYPEOF
       (cs\curve_style.curve_font)) = 1)))) = 0))) = 0))) = 0;
WR8 : (* for all styled_items get all assigned surface_style_usages
         and ensure that its style is a surface_side_style *)
       SIZEOF (QUERY (si <* QUERY (it <* SELF.items |
       'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
       IN TYPEOF (it)) |
       NOT (SIZEOF (QUERY (psa <* si\styled_item.styles |
       NOT (SIZEOF (QUERY (ssu <* QUERY (pss <* psa.styles |
       'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
       IN TYPEOF (pss)) |
       NOT ('PART_204_BREP_PRODUCT_SCHEMA.SURFACE_SIDE_STYLE'
   IN TYPEOF (ssu\surface_style_usage.style)) )) = 0))) = 0 ))) = 0;
WR9 : (* for all styled_items get all assigned surface_style_usages
         and the surface_side_styles that they reference, and ensure
         that the styles referenced by those surface_side_styles
         are among the valid subset *)
       SIZEOF (QUERY (si <* QUERY (it <* SELF.items |
       'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
       IN TYPEOF (it)) |
       NOT (SIZEOF (QUERY (psa <* si\styled_item.styles |
       NOT (SIZEOF (QUERY (ssu <* QUERY (pss <* psa.styles |
       'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
       IN TYPEOF (pss)) |
       NOT (SIZEOF (QUERY (sses <*
       ssu\surface_style_usage.style\surface_side_style.styles |
       NOT (SIZEOF (
       ['PART_204_BREP_PRODUCT_SCHEMA.'+
       'SURFACE_STYLE_PARAMETER_LINE',
       'PART_204_BREP_PRODUCT_SCHEMA.'+
       'SURFACE_STYLE_CONTROL_GRID',
       'PART_204_BREP_PRODUCT_SCHEMA.'+
       'SURFACE_STYLE_SILHOUETTE',
```



```
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'SURFACE_STYLE_SEGMENTATION_CURVE',
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'SURFACE_STYLE_BOUNDARY',
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'SURFACE_STYLE_FILL_AREA',
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'SURFACE_STYLE_RENDERING'] * TYPEOF (sses)) = 1)))
            = 0))) = 0))) = 0))) = 0;
WR10: (* for all surface_style_fill_areas that are referenced by
         any surface_side_styles ensure that they are valid
         with respect to their colour representation *)
        SIZEOF (QUERY (si <* QUERY (it <* SELF.items |
        'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
        IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (psa <* si\styled_item.styles |
        NOT (SIZEOF (QUERY (ssu <* QUERY (pss <* psa.styles |
        'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
        IN TYPEOF (pss)) |
        NOT (SIZEOF (QUERY (ssfa <* QUERY (sses <*
        ssu\surface_style_usage.style\surface_side_style.styles |
        'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_FILL_AREA'
        IN TYPEOF (sses)) |
        NOT (SIZEOF (QUERY (fss <*
        ssfa\surface_style_fill_area.fill_area.fill_styles |
        NOT (('PART_204_BREP_PRODUCT_SCHEMA.'+
        'FILL_AREA_STYLE_COLOUR' IN TYPEOF (fss))
        AND
        (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'DRAUGHTING_PRE_DEFINED_COLOUR'] * TYPEOF
        (fss\fill_area_style_colour.fill_colour)) = 1))))
           = 0))) = 0)))  = 0))) = 0))) = 0;
WR11: (* for all surface_style_parameter_lines that are referenced by
         any surface_side_styles ensure that they are valid
         with respect to the applied curve_style, which may include
         rendering *)
        SIZEOF (QUERY (si <* QUERY (it <* SELF.items |
        'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
        IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (psa <* si\styled_item.styles |
        NOT (SIZEOF (QUERY (ssu <* QUERY (pss <* psa.styles |
        'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
        IN TYPEOF (pss)) |
        NOT (SIZEOF (QUERY (sspl <* QUERY (sses <*
        ssu\surface_style_usage.style\surface_side_style.styles |
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'SURFACE_STYLE_PARAMETER_LINE' IN TYPEOF (sses)) |
        NOT ((
        ('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE'
        IN TYPEOF
         (sspl\surface_style_parameter_line.style_of_parameter_lines))
```

```
        AND
        (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'DRAUGHTING_PRE_DEFINED_COLOUR'] * TYPEOF
        (sspl\surface_style_parameter_line.
        style_of_parameter_lines\curve_style.curve_colour)) = 1)
        AND
        ('PART_204_BREP_PRODUCT_SCHEMA.POSITIVE_LENGTH_MEASURE' IN
        TYPEOF(sspl\surface_style_parameter_line.
        style_of_parameter_lines\curve_style.curve_width))
        AND
        (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.'+
        'CURVE_STYLE_FONT', 'PART_204_BREP_PRODUCT_SCHEMA.'+
        'DRAUGHTING_PRE_DEFINED_CURVE_FONT'] * TYPEOF
        (sspl\surface_style_parameter_line.
        style_of_parameter_lines\curve_style.curve_font)) = 1))
        OR (('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE_RENDERING'
        IN TYPEOF
        (sspl\surface_style_parameter_line.style_of_parameter_lines))
        AND
        (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'DRAUGHTING_PRE_DEFINED_COLOUR'] * TYPEOF
        (sspl\surface_style_parameter_line.style_of_parameter_lines\
         curve_style_rendering.rendering_properties.rendered_colour))
             = 1))) )) = 0))) = 0))) = 0))) = 0;
  WR12: (* for all surface_style_control_grids that are referenced by
           any surface_side_styles ensure that they are valid
           with respect to the applied curve_style, which may include
           rendering *)
        SIZEOF (QUERY (si <* QUERY (it <* SELF.items |
        'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
        IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (psa <* si\styled_item.styles |
        NOT (SIZEOF (QUERY (ssu <* QUERY (pss <* psa.styles |
        'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
        IN TYPEOF (pss)) |
        NOT (SIZEOF (QUERY (sscg <* QUERY (sses <*
        ssu\surface_style_usage.style\surface_side_style.styles |
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'SURFACE_STYLE_CONTROL_GRID' IN TYPEOF (sses)) |
        NOT ((
        ('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE'
        IN TYPEOF
            (sscg\surface_style_control_grid.style_of_control_grid))
        AND
        (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'DRAUGHTING_PRE_DEFINED_COLOUR'] * TYPEOF
        (sscg\surface_style_control_grid.
        style_of_control_grid\curve_style.curve_colour)) = 1)
        AND
```



```
        ('PART_204_BREP_PRODUCT_SCHEMA.POSITIVE_LENGTH_MEASURE' IN
        TYPEOF(sscg\surface_style_control_grid.
        style_of_control_grid\curve_style.curve_width))
        AND
        (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.'+
        'CURVE_STYLE_FONT', 'PART_204_BREP_PRODUCT_SCHEMA.'+
        'DRAUGHTING_PRE_DEFINED_CURVE_FONT'] * TYPEOF
        (sscg\surface_style_control_grid.
        style_of_control_grid\curve_style.curve_font)) = 1))
        OR (('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE_RENDERING'
        IN TYPEOF
        (sscg\surface_style_control_grid.style_of_control_grid))
        AND
        (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'DRAUGHTING_PRE_DEFINED_COLOUR'] * TYPEOF
        (sscg\surface_style_control_grid.style_of_control_grid\
        curve_style_rendering.rendering_properties.rendered_colour))
         = 1))) )) = 0))) = 0))) = 0))) = 0;
WR13: (* for all surface_style_silhouettes that are referenced by
         any surface_side_styles ensure that they are valid
         with respect to the applied curve_style, which may include
         rendering *)
        SIZEOF (QUERY (si <* QUERY (it <* SELF.items |
        'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
        IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (psa <* si\styled_item.styles |
        NOT (SIZEOF (QUERY (ssu <* QUERY (pss <* psa.styles |
        'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
        IN TYPEOF (pss)) |
        NOT (SIZEOF (QUERY (sssh <* QUERY (sses <*
        ssu\surface_style_usage.style\surface_side_style.styles |
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'SURFACE_STYLE_SILHOUETTE' IN TYPEOF (sses)) |
        NOT ((
        ('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE'
        IN TYPEOF (sssh\surface_style_silhouette.style_of_silhouette))
        AND
        (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'DRAUGHTING_PRE_DEFINED_COLOUR'] * TYPEOF
        (sssh\surface_style_silhouette.
        style_of_silhouette\curve_style.curve_colour)) = 1)
        AND
        ('PART_204_BREP_PRODUCT_SCHEMA.POSITIVE_LENGTH_MEASURE' IN TYPEOF
        (sssh\surface_style_silhouette.
        style_of_silhouette\curve_style.curve_width))
        AND
        (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.'+
        'CURVE_STYLE_FONT', 'PART_204_BREP_PRODUCT_SCHEMA.'+
        'DRAUGHTING_PRE_DEFINED_CURVE_FONT'] * TYPEOF
        (sssh\surface_style_silhouette.
```

```
        style_of_silhouette\curve_style.curve_font)) = 1))
        OR (('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE_RENDERING'
        IN TYPEOF (sssh\surface_style_silhouette.style_of_silhouette))
        AND
        (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'DRAUGHTING_PRE_DEFINED_COLOUR'] * TYPEOF
        (sssh\surface_style_silhouette.style_of_silhouette\
        curve_style_rendering.rendering_properties.rendered_colour))
         = 1))) )) = 0))) = 0))) = 0))) = 0;
  WR14: (* for all surface_style_segmentation_curves that are referenced by
           any surface_side_styles ensure that they are valid
           with respect to the applied curve_style, which may include
           rendering *)
        SIZEOF (QUERY (si <* QUERY (it <* SELF.items |
        'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
        IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (psa <* si\styled_item.styles |
        NOT (SIZEOF (QUERY (ssu <* QUERY (pss <* psa.styles |
        'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
        IN TYPEOF (pss)) |
        NOT (SIZEOF (QUERY (sssc <* QUERY (sses <*
        ssu\surface_style_usage.style\surface_side_style.styles |
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'SURFACE_STYLE_SEGMENTATION_CURVE' IN TYPEOF (sses)) |
        NOT ((
        ('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE'
        IN TYPEOF
        (sssc\surface_style_segmentation_curve.
          style_of_segmentation_curve))
        AND
        (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'DRAUGHTING_PRE_DEFINED_COLOUR'] * TYPEOF
        (sssc\surface_style_segmentation_curve.
        style_of_segmentation_curve\curve_style.curve_colour)) = 1)
        AND
        ('PART_204_BREP_PRODUCT_SCHEMA.POSITIVE_LENGTH_MEASURE' IN
        TYPEOF(sssc\surface_style_segmentation_curve.
        style_of_segmentation_curve\curve_style.curve_width))
        AND
        (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.'+
        'CURVE_STYLE_FONT', 'PART_204_BREP_PRODUCT_SCHEMA.'+
        'DRAUGHTING_PRE_DEFINED_CURVE_FONT'] * TYPEOF
        (sssc\surface_style_segmentation_curve.
        style_of_segmentation_curve\curve_style.curve_font)) = 1))
        OR (('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE_RENDERING'
        IN TYPEOF (sssc\surface_style_segmentation_curve.
        style_of_segmentation_curve))
        AND
        (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
        'PART_204_BREP_PRODUCT_SCHEMA.'+
```



```
      'DRAUGHTING_PRE_DEFINED_COLOUR'] * TYPEOF
   (sssc\surface_style_segmentation_curve.style_of_segmentation_curve\
      curve_style_rendering.rendering_properties.rendered_colour))
       = 1))) ))  = 0))) = 0))) = 0))) = 0;
WR15: (* for all surface_style_boundaries that are referenced by
         any surface_side_styles ensure that they are valid
         with respect to the applied curve_style, which may include
         rendering *)
      SIZEOF (QUERY (si <* QUERY (it <* SELF.items |
      'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
      IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (psa <* si\styled_item.styles |
      NOT (SIZEOF (QUERY (ssu <* QUERY (pss <* psa.styles |
      'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
      IN TYPEOF (pss)) |
      NOT (SIZEOF (QUERY (ssbd <* QUERY (sses <*
      ssu\surface_style_usage.style\surface_side_style.styles |
      'PART_204_BREP_PRODUCT_SCHEMA.'+
      'SURFACE_STYLE_BOUNDARY' IN TYPEOF (sses)) |
      NOT ((
      ('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE'
      IN TYPEOF (ssbd\surface_style_boundary.style_of_boundary))
      AND
      (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
      'PART_204_BREP_PRODUCT_SCHEMA.'+
      'DRAUGHTING_PRE_DEFINED_COLOUR'] * TYPEOF
      (ssbd\surface_style_boundary.
      style_of_boundary\curve_style.curve_colour)) = 1)
      AND
      ('PART_204_BREP_PRODUCT_SCHEMA.POSITIVE_LENGTH_MEASURE' IN
      TYPEOF(ssbd\surface_style_boundary.
      style_of_boundary\curve_style.curve_width))
      AND
      (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.'+
      'CURVE_STYLE_FONT', 'PART_204_BREP_PRODUCT_SCHEMA.'+
      'DRAUGHTING_PRE_DEFINED_CURVE_FONT'] * TYPEOF
      (ssbd\surface_style_boundary.
      style_of_boundary\curve_style.curve_font)) = 1))
      OR (('PART_204_BREP_PRODUCT_SCHEMA.CURVE_STYLE_RENDERING'
      IN TYPEOF (ssbd\surface_style_boundary.style_of_boundary))
      AND
      (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
      'PART_204_BREP_PRODUCT_SCHEMA.'+
      'DRAUGHTING_PRE_DEFINED_COLOUR'] * TYPEOF
      (ssbd\surface_style_boundary.
      style_of_boundary\curve_style_rendering.
       rendering_properties.rendered_colour)) = 1))) ))
      = 0))) = 0))) = 0))) = 0;
WR16: (* for all surface_style_renderings that are referenced by
         any surface_side_styles ensure that the colour
         representation is valid *)
      SIZEOF (QUERY (si <* QUERY (it <* SELF.items |
```

```
        'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM'
        IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (psa <* si\styled_item.styles |
        NOT (SIZEOF (QUERY (ssu <* QUERY (pss <* psa.styles |
        'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE'
        IN TYPEOF (pss)) |
        NOT (SIZEOF (QUERY (ssre <* QUERY (sses <*
        ssu\surface_style_usage.style\surface_side_style.styles |
        'PART_204_BREP_PRODUCT_SCHEMA.'+
        'SURFACE_STYLE_RENDERING' IN TYPEOF (sses)) |
        NOT (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.COLOUR_RGB',
        'PART_204_BREP_PRODUCT_SCHEMA.'+
         'DRAUGHTING_PRE_DEFINED_COLOUR'] * TYPEOF
        (ssre\surface_style_rendering.surface_colour)) = 1)))
        = 0))) = 0))) = 0))) = 0;
END_ENTITY;


ENTITY named_unit
  SUPERTYPE OF (ONEOF (
              SI_UNIT,
              CONVERSION_BASED_UNIT));
  dimensions : dimensional_exponents;
END_ENTITY;

ENTITY open_shell
   SUBTYPE OF (connected_face_set);
END_ENTITY;

END_ENTITY;
 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 ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_CLOSED_SHELL'
               IN TYPEOF (SELF.closed_shell_element));
END_ENTITY;

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 ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_EDGE' IN
                                     TYPEOF (SELF.edge_element));
END_ENTITY;

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 ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_FACE'
                              IN TYPEOF (SELF.face_element));
END_ENTITY;

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 ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_OPEN_SHELL'
                IN TYPEOF (SELF.open_shell_element));
END_ENTITY;

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 ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_PATH' IN
      TYPEOF (SELF.path_element));
END_ENTITY;

ENTITY over_riding_styled_item
  SUBTYPE OF (styled_item);
  over_ridden_style : styled_item;
END_ENTITY;

ENTITY parabola
   SUBTYPE OF (conic);
```

```
   focal_dist : length_measure;
 WHERE
   WR1: focal_dist <> 0.0;
 END_ENTITY;

ENTITY parametric_representation_context
  SUBTYPE OF (representation_context);
END_ENTITY;

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;

ENTITY pcurve
   SUBTYPE OF (curve);
   basis_surface      : surface;
   reference_to_curve : definitional_representation;
 WHERE
   WR1: SIZEOF(reference_to_curve\representation.items) = 1;
   WR2: 'PART_204_BREP_PRODUCT_SCHEMA.CURVE' IN TYPEOF

                    (reference_to_curve\representation.items[1]);
   WR3: reference_to_curve\representation.
                   items[1]\geometric_representation_item.dim =2;
 END_ENTITY;

ENTITY placement
   SUPERTYPE OF (ONEOF (
              AXIS1_PLACEMENT,
              AXIS2_PLACEMENT_2D,
              AXIS2_PLACEMENT_3D))
  SUBTYPE OF (geometric_representation_item);
   location : cartesian_point;
END_ENTITY;

ENTITY planar_box
  SUBTYPE OF (planar_extent);
  placement:  axis2_placement;
END_ENTITY;

ENTITY planar_extent
  SUBTYPE OF (geometric_representation_item);
   size_in_x : length_measure;
   size_in_y : length_measure;
END_ENTITY;
```



```
ENTITY plane
 SUBTYPE OF (elementary_surface);
 END_ENTITY;

ENTITY point
   SUPERTYPE OF (ONEOF (
              CARTESIAN_POINT))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY;

ENTITY point_style;
  name          : label;
  marker        : marker_select;
  marker_size   : size_select;
  marker_colour : colour;
END_ENTITY;

ENTITY polyline
   SUBTYPE OF (bounded_curve);
   points : LIST [2:?] OF cartesian_point;
 END_ENTITY;

ENTITY poly_loop
   SUBTYPE OF (loop,geometric_representation_item);
   polygon : LIST [3:?] OF UNIQUE cartesian_point;
END_ENTITY;

ENTITY pre_defined_colour
  SUBTYPE OF (pre_defined_item, colour);
END_ENTITY;

ENTITY pre_defined_curve_font
  SUBTYPE OF (pre_defined_item);
END_ENTITY;



ENTITY pre_defined_item;
  name : label;
END_ENTITY;

ENTITY presentation_area
  SUBTYPE OF (presentation_representation);
WHERE
  WR1: ((SIZEOF (QUERY (ais <*
       USEDIN (SELF, 'PART_204_BREP_PRODUCT_SCHEMA.' +
                                          'AREA_IN_SET.AREA') |
           SIZEOF (USEDIN (ais, 'PART_204_BREP_PRODUCT_SCHEMA.' +
                               'PRESENTATION_SIZE.UNIT')) =1)) > 0) OR
         (SIZEOF (USEDIN (SELF, 'PART_204_BREP_PRODUCT_SCHEMA.' +
                                    'PRESENTATION_SIZE.UNIT')) =1));
END_ENTITY;
```

```
ENTITY presentation_representation
  SUBTYPE OF (representation);
WHERE
  WR1: SELF\representation.
            context_of_items\geometric_representation_context.
            coordinate_space_dimension = 2;
  WR2: 'PART_204_BREP_PRODUCT_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT'
       IN TYPEOF (SELF\representation.context_of_items);
END_ENTITY;

ENTITY presentation_set;
INVERSE
  areas : SET [1:?] OF area_in_set FOR in_set;
END_ENTITY;

ENTITY presentation_size;
  unit : presentation_size_assignment_select;
  size : planar_box;
WHERE
  WR1: (('PART_204_BREP_PRODUCT_SCHEMA.PRESENTATION_REPRESENTATION'
         IN TYPEOF (SELF.unit)) AND
         item_in_context (SELF.size,
                          SELF.unit\representation.context_of_items)
       )
            OR
       (
        ('PART_204_BREP_PRODUCT_SCHEMA.AREA_IN_SET'

          IN TYPEOF (SELF.unit)) AND
     (SIZEOF (QUERY ( ais <* SELF.unit\area_in_set.in_set.areas |
             NOT item_in_context (SELF.size, ais.area\representation.
                                        context_of_items) )) = 0)
       );
END_ENTITY;


ENTITY presentation_style_assignment;
  styles : SET [1:?] OF presentation_style_select;
WHERE
  WR1: SIZEOF (QUERY (style1 <* SELF.styles |
         NOT (SIZEOF (QUERY (style2 <* (SELF.styles - style1) |
           NOT ((TYPEOF (style1) <> TYPEOF (style2)) OR
             (SIZEOF (['PART_204_BREP_PRODUCT_SCHEMA.' +
                       'SURFACE_STYLE_USAGE',
                       'PART_204_BREP_PRODUCT_SCHEMA.'+
                       'EXTERNALLY_DEFINED_STYLE'] *
                     TYPEOF (style1)) = 1)
           ))) = 0
         ))) = 0;
  WR2: SIZEOF (QUERY (style1 <* SELF.styles |
         'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_STYLE_USAGE' IN
```



```
        TYPEOF(style1)
        )) <= 2;
END_ENTITY;

ENTITY presentation_style_by_context
  SUBTYPE OF (presentation_style_assignment);
  style_context : style_context_select;
END_ENTITY;

ENTITY presentation_view
  SUBTYPE OF (presentation_representation);
END_ENTITY;

ENTITY product;
  id                : identifier;
  name              : label;
  description       : text;
  frame_of_reference : SET [1:?] OF product_context;
UNIQUE
  UR1: id;
END_ENTITY;


ENTITY product_context
  SUBTYPE OF (application_context_element);
  discipline_type : label;
END_ENTITY;


ENTITY product_definition;
  id                : identifier;
  description       : text;
  formation         : product_definition_formation;
  frame_of_reference : product_definition_context;
END_ENTITY;


ENTITY product_definition_context
  SUBTYPE OF (application_context_element);
  life_cycle_stage : label;
END_ENTITY;


ENTITY product_definition_formation;
  id          : identifier;
  description : text;
  of_product  : product;
UNIQUE
  UR1: id, of_product;
END_ENTITY;
```

```
ENTITY product_definition_relationship;
  id                          : identifier;
  name                        : label;
  description                 : text;
  relating_product_definition : product_definition;
  related_product_definition  : product_definition;
END_ENTITY;

ENTITY product_definition_shape
 SUBTYPE OF (property_definition);
UNIQUE
 UR1:  SELF\property_definition.definition;
WHERE
 WR1:  NOT ('PART_204_BREP_PRODUCT_SCHEMA.SHAPE_DEFINITION'

       IN TYPEOF (SELF\property_definition.definition));
END_ENTITY;

ENTITY product_definition_usage
  SUPERTYPE OF (ONEOF (
             ASSEMBLY_COMPONENT_USAGE))
  SUBTYPE OF (product_definition_relationship);
UNIQUE
  UR1: SELF\product_definition_relationship.id,
       SELF\product_definition_relationship.relating_product_definition,
       SELF\product_definition_relationship.related_product_definition;
WHERE
  WR1:  acyclic_product_definition_relationship
         (SELF,
        [SELF\product_definition_relationship.related_product_definition],
         'PART_204_BREP_PRODUCT_SCHEMA.PRODUCT_DEFINITION_USAGE');
END_ENTITY;

ENTITY property_definition;
  name        : label;
  description : text;
  definition  : characterized_definition;
END_ENTITY;

ENTITY property_definition_representation;
  definition          : property_definition;
  used_representation : representation;
END_ENTITY;

ENTITY quasi_uniform_curve
   SUBTYPE OF (b_spline_curve);
 END_ENTITY;

ENTITY quasi_uniform_surface
   SUBTYPE OF (b_spline_surface);
 END_ENTITY;
```



```
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;

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;

ENTITY representation;
  name             : label;
  items            : SET[1:?] OF representation_item;
  context_of_items : representation_context;
END_ENTITY;

ENTITY representation_context;
  context_identifier : identifier;
  context_type       : text;
INVERSE
  representations_in_context : SET [1:?] OF representation
    FOR context_of_items;
END_ENTITY;

ENTITY representation_item;
  name : label;
WHERE
  WR1: SIZEOF(using_representations(SELF)) > 0;
END_ENTITY;

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;

ENTITY shape_definition_representation
  SUBTYPE OF (property_definition_representation);
WHERE
  WR1: ('PART_204_BREP_PRODUCT_SCHEMA.SHAPE_DEFINITION' IN
       TYPEOF (SELF.definition.definition))
       OR
       ('PART_204_BREP_PRODUCT_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN
       TYPEOF (SELF.definition));
  WR2: 'PART_204_BREP_PRODUCT_SCHEMA.SHAPE_REPRESENTATION' IN

       TYPEOF(SELF.used_representation);
END_ENTITY;


ENTITY shape_representation
  SUBTYPE OF (representation);
END_ENTITY;

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;


ENTITY solid_model
   SUPERTYPE OF (ONEOF (
              MANIFOLD_SOLID_BREP))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY;

ENTITY spherical_surface
   SUBTYPE OF (elementary_surface);
   radius   : positive_length_measure;
END_ENTITY;


ENTITY styled_item
  SUBTYPE OF (representation_item);
  styles :  SET [1:?] OF presentation_style_assignment;
  item   :  representation_item;
```



```
WHERE
  WR1: (SIZEOF(SELF.styles) = 1)
              XOR
       (SIZEOF(QUERY(pres_style <* SELF.styles |
         NOT ('PART_204_BREP_PRODUCT_SCHEMA.' +

              'PRESENTATION_STYLE_BY_CONTEXT' IN
         TYPEOF(pres_style))
         )) = 0);
END_ENTITY;

ENTITY surface
   SUPERTYPE OF (ONEOF (
              ELEMENTARY_SURFACE,
              SWEPT_SURFACE,
              BOUNDED_SURFACE))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY;

ENTITY 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: ('PART_204_BREP_PRODUCT_SCHEMA.PCURVE' IN
        TYPEOF(associated_geometry[1])) OR
                        (master_representation <> pcurve_s1);
   WR3: ('PART_204_BREP_PRODUCT_SCHEMA.PCURVE' IN
                   TYPEOF(associated_geometry[2])) OR
                        (master_representation <> pcurve_s2);
   WR4: NOT ('PART_204_BREP_PRODUCT_SCHEMA.PCURVE' IN TYPEOF(curve_3d));

END_ENTITY;

ENTITY surface_of_linear_extrusion
   SUBTYPE OF (swept_surface);
   extrusion_axis      : vector;
END_ENTITY;

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;

ENTITY surface_rendering_properties;
  rendered_colour : colour;
END_ENTITY;

ENTITY surface_side_style;
  name   : label;
  styles : SET [1:7] OF surface_style_element_select;
WHERE
  WR1: SIZEOF(QUERY( style1 <* SELF.styles |
         SIZEOF(QUERY( style2 <* SELF.styles - style1 |
           TYPEOF(style1) = TYPEOF(style2)
           )) > 0
         )) = 0;
END_ENTITY;

ENTITY surface_style_boundary;
  style_of_boundary : curve_or_render;
END_ENTITY;

ENTITY surface_style_control_grid;
  style_of_control_grid : curve_or_render;
END_ENTITY;

ENTITY surface_style_fill_area;
  fill_area : fill_area_style;
END_ENTITY;

ENTITY surface_style_parameter_line;
  style_of_parameter_lines : curve_or_render;
  direction_counts         : SET [1:2] OF direction_count_select;
WHERE
  WR1: (HIINDEX(SELF.direction_counts) = 1)
                       XOR
       (TYPEOF(SELF.direction_counts[1]) <>
          TYPEOF(SELF.direction_counts[2]));
END_ENTITY;

ENTITY surface_style_reflectance_ambient;
  ambient_reflectance : REAL;
END_ENTITY;

ENTITY surface_style_reflectance_ambient_diffuse
  SUBTYPE OF (surface_style_reflectance_ambient);
  diffuse_reflectance : REAL;
END_ENTITY;

ENTITY surface_style_reflectance_ambient_diffuse_specular
  SUBTYPE OF (surface_style_reflectance_ambient_diffuse);
  specular_reflectance : REAL;
```



```
  specular_exponent    : REAL;
  specular_colour      : colour;
END_ENTITY;

ENTITY surface_style_rendering;
  rendering_method : shading_surface_method;
  surface_colour   : colour;
END_ENTITY;

ENTITY surface_style_rendering_with_properties
  SUBTYPE OF (surface_style_rendering);
  properties : SET [1:2] OF rendering_properties_select;
WHERE
  WR1: (HIINDEX(SELF.properties) = 1)
                  XOR
       (TYPEOF(SELF.properties[1]) <> TYPEOF(SELF.properties[2]));
END_ENTITY;

ENTITY surface_style_segmentation_curve;
  style_of_segmentation_curve : curve_or_render;
END_ENTITY;

ENTITY surface_style_silhouette;
  style_of_silhouette : curve_or_render;
END_ENTITY;

ENTITY surface_style_transparent;
  transparency : REAL;
WHERE
  WR1: {0.0 <= transparency <= 1.0};
END_ENTITY;

ENTITY surface_style_usage;
  side  : surface_side;
  style : surface_side_style_select;
END_ENTITY;

ENTITY swept_surface
   SUPERTYPE OF (ONEOF (
              SURFACE_OF_LINEAR_EXTRUSION,
              SURFACE_OF_REVOLUTION))
  SUBTYPE OF (surface);
   swept_curve : curve;
 END_ENTITY;


ENTITY topological_representation_item
   SUPERTYPE OF (ONEOF (
              VERTEX,
              EDGE,
              FACE_BOUND,
              FACE,
```

```
              CONNECTED_FACE_SET,(
              LOOP
              ANDOR
              PATH)))
  SUBTYPE OF (representation_item);
END_ENTITY;

ENTITY toroidal_surface
   SUBTYPE OF (elementary_surface);
   major_radius : positive_length_measure;
   minor_radius : positive_length_measure;
END_ENTITY;

ENTITY uniform_curve
   SUBTYPE OF (b_spline_curve);
END_ENTITY;

ENTITY uniform_surface
   SUBTYPE OF (b_spline_surface);
END_ENTITY;

ENTITY vector
   SUBTYPE OF (geometric_representation_item);
   orientation : direction;
   magnitude   : length_measure;
 WHERE
   WR1 : magnitude >= 0.0;
END_ENTITY;

ENTITY vertex
   SUBTYPE OF (topological_representation_item);
END_ENTITY;

ENTITY vertex_loop
   SUBTYPE OF (loop);
   loop_vertex : vertex;
END_ENTITY;

ENTITY vertex_point
    SUBTYPE OF(vertex,geometric_representation_item);
      vertex_geometry : point;
END_ENTITY;

ENTITY view_volume
  SUBTYPE OF (founded_item);
  projection_type           : central_or_parallel;
  projection_point          : cartesian_point;
  view_plane_distance       : length_measure;
  front_plane_distance      : length_measure;
  front_plane_clipping      : BOOLEAN;
  back_plane_distance       : length_measure;
  back_plane_clipping       : BOOLEAN;
```



```
  view_volume_sides_clipping : BOOLEAN;
  view_window                : planar_box;
END_ENTITY;


RULE advanced_or_elementary_or_faceted FOR(shape_representation);
WHERE
   WR1: SIZEOF (QUERY (sr <* shape_representation |
                NOT( SIZEOF(
  ['PART_204_BREP_PRODUCT_SCHEMA.ADVANCED_BREP_SHAPE_REPRESENTATION',
   'PART_204_BREP_PRODUCT_SCHEMA.ELEMENTARY_BREP_SHAPE_REPRESENTATION',
    'PART_204_BREP_PRODUCT_SCHEMA.FACETED_BREP_SHAPE_REPRESENTATION']
          * TYPEOF (sr)) =1 ))) = 0;
END_RULE;


RULE compatible_dimension FOR
  (cartesian_point,
  direction,
  representation_context,
  geometric_representation_context);
WHERE

  -- ensure that the count of coordinates of each cartesian_point
  -- matches the coordinate_space_dimension of each geometric_context in
  -- which it is geometrically founded
  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;

  -- ensure that the count of direction_ratios of each direction
  -- matches the coordinate_space_dimension of each geometric_context in
  -- which it is geometrically founded
  WR2: SIZEOF(QUERY(x <* direction | SIZEOF( QUERY
       (y <* geometric_representation_context | item_in_context(x,y) AND
       (HIINDEX(x.direction_ratios) <> y.coordinate_space_dimension)))
       > 0 )) = 0;
END_RULE;


RULE dependent_instantiation_of_geometry FOR
                         (geometric_representation_item);
   WHERE
      WR1 : SIZEOF ( QUERY (gri <* geometric_representation_item |
                 NOT (SIZEOF (USEDIN (gri,'')) > 0 ))) = 0;
END_RULE;


RULE dependent_instantiation_of_mapped_item FOR (mapped_item);
   WHERE
      WR1 : SIZEOF ( QUERY (mi <* mapped_item |
```

```
                NOT (SIZEOF (USEDIN (mi,'')) > 0 ))) = 0;
END_RULE;


RULE dependent_instantiation_of_mechanical_design_geometric_p_r
   FOR (mechanical_design_geometric_presentation_representation);
WHERE
  WR1 : SIZEOF ( QUERY (mdpr <*
        mechanical_design_geometric_presentation_representation |
        NOT (SIZEOF (USEDIN (mdpr,'')) > 0 ))) = 0;
END_RULE;


RULE dependent_instantiation_of_mechanical_design_shaded_p_r
   FOR (mechanical_design_shaded_presentation_representation);
WHERE
  WR1 : SIZEOF ( QUERY (mdpr <*
        mechanical_design_shaded_presentation_representation |
        NOT (SIZEOF (USEDIN (mdpr,'')) > 0 ))) = 0;
END_RULE;


RULE dependent_instantiation_of_named_unit FOR (named_unit);
   WHERE
      WR1 : SIZEOF ( QUERY (nmu <* named_unit |
                NOT (SIZEOF (USEDIN (nmu,'')) > 0 ))) = 0;
END_RULE;


RULE dependent_instantiation_of_pre_defined_item FOR
                  (pre_defined_item);
   WHERE
      WR1 : SIZEOF ( QUERY (pdi <* pre_defined_item |
                NOT (SIZEOF (USEDIN (pdi,'')) > 0 ))) = 0;
END_RULE;


RULE dependent_instantiation_of_presentation_style FOR
                  (presentation_style_assignment);
   WHERE
      WR1 : SIZEOF ( QUERY (psa <* presentation_style_assignment |
                NOT (SIZEOF (USEDIN (psa,'')) > 0 ))) = 0;
END_RULE;


RULE dependent_instantiation_of_product_context FOR (product_context);
WHERE
  WR1 : SIZEOF ( QUERY (pc <* product_context |
        NOT (SIZEOF (USEDIN (pc,'')) > 0 ))) = 0;
END_RULE;
```



```
RULE dependent_instantiation_of_product_definition
                              FOR (product_definition);
WHERE
  WR1 : SIZEOF ( QUERY (pd <* product_definition |
        NOT (SIZEOF (USEDIN (pd,'')) > 0 ))) = 0;
END_RULE;


RULE dependent_instantiation_of_product_definition_context FOR
     (product_definition_context);
WHERE
  WR1 : SIZEOF ( QUERY (pdc <* product_definition_context |
        NOT (SIZEOF (USEDIN (pdc,'')) > 0 ))) = 0;
END_RULE;


RULE dependent_instantiation_of_product_definition_formation
             FOR (product_definition_formation);
WHERE
  WR1 : SIZEOF ( QUERY (pdf <* product_definition_formation |
-- old        NOT (SIZEOF (USEDIN (pv,'')) > 0 ))) = 0;
-- new JH:
        NOT (SIZEOF (USEDIN (pdf,'')) > 0 ))) = 0;
END_RULE;


RULE dependent_instantiation_of_product_definition_relationship FOR
     (product_definition_relationship);
WHERE
  WR1 : SIZEOF ( QUERY (pdr <* product_definition_relationship |
        NOT (SIZEOF (USEDIN (pdr,'')) > 0 ))) = 0;
END_RULE;


RULE dependent_instantiation_of_shape_representation
                               FOR (shape_representation);
WHERE
  WR1 : SIZEOF ( QUERY (sr <* shape_representation |
        NOT (SIZEOF (USEDIN (sr,'')) > 0 ))) = 0;
END_RULE;

RULE dependent_instantiation_of_styled_item FOR (styled_item);
   WHERE
      WR1 : SIZEOF ( QUERY (si <* styled_item |
                 NOT (SIZEOF (USEDIN (si,'')) > 0 ))) = 0;
END_RULE;


RULE dependent_instantiation_of_topology
             FOR (topological_representation_item);
```

```
   WHERE
      WR1 : SIZEOF ( QUERY (tri <* topological_representation_item |
                  NOT (SIZEOF (USEDIN (tri,'')) > 0 ))) = 0;
END_RULE;


RULE global_units_in_geometric_representation_context FOR
      (geometric_representation_context);
WHERE
  WR1 :  SIZEOF (QUERY (grc <* geometric_representation_context |
         NOT ('PART_204_BREP_PRODUCT_SCHEMA.' +
         'GLOBAL_UNIT_ASSIGNED_CONTEXT' IN TYPEOF (grc)))) = 0 ;
END_RULE;


RULE global_units_required 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 |
                    'PART_204_BREP_PRODUCT_SCHEMA.LENGTH_UNIT' IN TYPEOF
                    (u))) =1 ))) = 0;
      WR3:  SIZEOF ( QUERY (guac <* global_unit_assigned_context |
                  NOT (SIZEOF (QUERY( u <* guac.units |
         NOT (( 'PART_204_BREP_PRODUCT_SCHEMA.LENGTH_UNIT' IN TYPEOF(u))
                  OR ( 'PART_204_BREP_PRODUCT_SCHEMA.PLANE_ANGLE_UNIT' IN

                       TYPEOF (u))))) = 0))) = 0;
END_RULE;


RULE illegal_complex_named_units FOR (named_unit);
WHERE
  WR1: SIZEOF (QUERY (nu <* named_unit |
       NOT (SIZEOF (TYPEOF(nu) *
       ['PART_204_BREP_PRODUCT_SCHEMA.LENGTH_UNIT',
        'PART_204_BREP_PRODUCT_SCHEMA.PLANE_ANGLE_UNIT']) < 2 )
       AND
       NOT (SIZEOF (TYPEOF(nu) *
       ['PART_204_BREP_PRODUCT_SCHEMA.LENGTH_UNIT',
        'PART_204_BREP_PRODUCT_SCHEMA.CONVERSION_BASED_UNIT']) < 2 )

       )) = 0;
END_RULE;


RULE no_complex_subtypes FOR(geometric_representation_item);
   WHERE
    WR1: SIZEOF (QUERY (gri <* geometric_representation_item |
    NOT (SIZEOF (TYPEOF(gri) *
```



```
            ['PART_204_BREP_PRODUCT_SCHEMA.CAMERA_IMAGE',
             'PART_204_BREP_PRODUCT_SCHEMA.CAMERA_MODEL',
   'PART_204_BREP_PRODUCT_SCHEMA.CARTESIAN_TRANSFORMATION_OPERATOR',
             'PART_204_BREP_PRODUCT_SCHEMA.CURVE',
              'PART_204_BREP_PRODUCT_SCHEMA.DIRECTION',
          'PART_204_BREP_PRODUCT_SCHEMA.EDGE_CURVE',
          'PART_204_BREP_PRODUCT_SCHEMA.FACE_SURFACE',
           'PART_204_BREP_PRODUCT_SCHEMA.PLACEMENT',
            'PART_204_BREP_PRODUCT_SCHEMA.POINT',
          'PART_204_BREP_PRODUCT_SCHEMA.POLY_LOOP',
            'PART_204_BREP_PRODUCT_SCHEMA.SOLID_MODEL',
             'PART_204_BREP_PRODUCT_SCHEMA.SURFACE',
              'PART_204_BREP_PRODUCT_SCHEMA.VECTOR',
          'PART_204_BREP_PRODUCT_SCHEMA.VERTEX_POINT']) < 2 ))) = 0;
END_RULE;


RULE product_context_mechanical FOR (product_context);
   WHERE
      WR1 : SIZEOF ( QUERY (pc <* product_context |
               NOT ('PART_204_BREP_PRODUCT_SCHEMA.MECHANICAL_CONTEXT'
            IN TYPEOF(pc)))) = 0;
END_RULE;


RULE product_definition_context_design FOR (product_definition_context);
   WHERE
      WR1 : SIZEOF ( QUERY (pdc <* product_definition_context |
                  NOT ('PART_204_BREP_PRODUCT_SCHEMA.DESIGN_CONTEXT'

                    IN TYPEOF(pdc)))) = 0;
END_RULE;


RULE shape_representation_required FOR (product_definition_shape);
   WHERE
      WR1 : SIZEOF ( QUERY (pds <* product_definition_shape |
              NOT ( SIZEOF(USEDIN
            (pds,'PART_204_BREP_PRODUCT_SCHEMA.'+
                 'SHAPE_DEFINITION_REPRESENTATION.DEFINITION')) > 0
                 ))) = 0;
END_RULE;  -- shape_representation_required



RULE single_curve_style FOR (curve);
   WHERE
      WR1 : SIZEOF ( QUERY (crv <* curve |
                  SIZEOF(USEDIN
            (crv,'PART_204_BREP_PRODUCT_SCHEMA.STYLED_ITEM.ITEM')) > 1
                 )) = 0;
```

```
END_RULE;  -- single_curve_style


RULE three_dimensional_geometry FOR(geometric_representation_item);
   WHERE
    WR1: SIZEOF(QUERY(gri <* geometric_representation_item |
         (dimension_of(gri) = 2) AND NOT ( (SIZEOF (
            ['PART_204_BREP_PRODUCT_SCHEMA.PLANAR_BOX',
            'PART_204_BREP_PRODUCT_SCHEMA.AXIS2_PLACEMENT_2D'] *
              TYPEOF(gri)) = 1 )
              OR
              (SIZEOF(QUERY(rep <* using_representations(gri) |
              'PART_204_BREP_PRODUCT_SCHEMA.DEFINITIONAL_REPRESENTATION'

           IN TYPEOF(rep) )) > 0 )
              ))) = 0;
 END_RULE;



FUNCTION acyclic_mapped_representation
  (parent_set   : SET OF representation;
   children_set : SET OF representation_item) : BOOLEAN;
  LOCAL
    x,y : SET OF representation_item;
    i   : INTEGER;
  END_LOCAL;
  -- Determine the subset of children_set that are mapped_items.
  x := QUERY(z <* children_set |
      'PART_204_BREP_PRODUCT_SCHEMA.MAPPED_ITEM' IN TYPEOF(z));
  -- Determine that the subset has elements.
  IF SIZEOF(x) > 0 THEN
    -- Check each element of the set.
    REPEAT i := 1 TO HIINDEX(x);
      -- If the selected element maps a representation in the
      --  parent_set, then return false.
      IF x[i]\mapped_item.mapping_source.mapped_representation
        IN parent_set THEN
        RETURN (FALSE);
      END_IF;
      -- Recursive check of the items of mapped_rep.
      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;
  -- Determine the subset of children_set that are not
  -- mapped_items.
```



```
  x := children_set - x;
  -- Determine that the subset has elements.
  IF SIZEOF(x) > 0 THEN
    -- For each element of the set:
    REPEAT i := 1 TO HIINDEX(x);
      -- Determine the set of representation_items referenced.
      y := QUERY(z <* bag_to_set( USEDIN(x[i], '')) |
           'PART_204_BREP_PRODUCT_SCHEMA.REPRESENTATION_ITEM'
               IN TYPEOF(z));
      -- Recursively check for an offending mapped_item.
      -- Return false for any errors encountered.
      IF NOT acyclic_mapped_representation(parent_set, y) THEN
        RETURN (FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  -- Return true when all elements are checked and
  -- no error conditions found.
  RETURN (TRUE);
END_FUNCTION;

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;             -- IN is based in instance equality

  x := QUERY (pd <* bag_to_set (USEDIN
       (relation.relating_product_definition,
        'PART_204_BREP_PRODUCT_SCHEMA.' +

        'PRODUCT_DEFINITION_RELATIONSHIP.' +
        'RELATED_PRODUCT_DEFINITION')) |
         specific_relation IN TYPEOF (pd));

  REPEAT I := 1 TO HIINDEX(x);             -- pre-checked loop
    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 aspect_ratio (p : planar_box) : positive_ratio_measure;
(* if the dimensions of the planar_box are greater than zero,
      compute the aspect ratio and return the resulting value. *)
   IF (p.size_in_x > 0.) AND (p.size_in_y > 0.) THEN
      RETURN (p.size_in_x / p.size_in_y);
   ELSE
      RETURN (?);
   END_IF;
END_FUNCTION;

FUNCTION associated_surface(arg : pcurve_or_surface) : surface;
   LOCAL
     surf : surface;
   END_LOCAL;
   IF 'PART_204_BREP_PRODUCT_SCHEMA.PCURVE' IN TYPEOF(arg) THEN
     surf := arg.basis_surface;
   ELSE
     surf := arg;
   END_IF;
   RETURN(surf);
END_FUNCTION;

FUNCTION bag_to_set (the_bag : BAG OF GENERIC : intype) :
                                    SET OF GENERIC : intype;
  LOCAL
    the_set: SET OF GENERIC : intype := [];
    i      : INTEGER;
  END_LOCAL;
  IF SIZEOF (the_bag) > 0 THEN
    REPEAT i := 1 to HIINDEX (the_bag);
      the_set := the_set + the_bag [i];
    END_REPEAT;
  END_IF;
  RETURN (the_set);
END_FUNCTION;

FUNCTION base_axis(dim : INTEGER; axis1, axis2, axis3 : direction) :
                                LIST [2:3] OF direction;
  LOCAL
    u      : LIST [2:3] OF direction;
    factor : REAL;
    d1, d2 : direction;
  END_LOCAL;
  IF (dim = 3) THEN
    d1 := NVL(normalise(axis3),  dummy_gri || direction([0.0,0.0,1.0]));
    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 < 0.0) 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, 0.0]), dummy_gri ||
                                                  direction([0.0, 1.0])];
      END_IF;
    END_IF;
  END_IF;
  RETURN(u);
END_FUNCTION;

FUNCTION boolean_choose (b : boolean;
           choice1, choice2 : generic : item) : generic : item;

   IF b THEN
     RETURN (choice1);
   ELSE
     RETURN (choice2);
   END_IF;
END_FUNCTION;

FUNCTION build_axes(axis, ref_direction : direction) :
                                        LIST [3:3] OF direction;
   LOCAL
     d1, d2 : direction;
   END_LOCAL;
  d1 := NVL(normalise(axis), dummy_gri || direction([0.0,0.0,1.0]));
  d2 := first_proj_axis(d1, ref_direction);
  RETURN([d2, normalise(cross_product(d1,d2)).orientation, d1]);
END_FUNCTION;

FUNCTION build_2axes(ref_direction : direction) : LIST [2:2] OF direction;
   LOCAL
     d : direction := NVL(normalise(ref_direction),
                          dummy_gri || direction([1.0,0.0]));
   END_LOCAL;

   RETURN([d, orthogonal_complement(d)]);
END_FUNCTION;

FUNCTION closed_shell_reversed (a_shell : closed_shell) :
```

```
                                        oriented_closed_shell;
 LOCAL
   the_reverse : oriented_closed_shell;
 END_LOCAL;
  IF ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_CLOSED_SHELL'
         IN TYPEOF (a_shell) ) THEN
     the_reverse := dummy_tri ||
                    connected_face_set (
                       a_shell\connected_face_set.cfs_faces) ||
                    closed_shell () || oriented_closed_shell(
                       a_shell\oriented_closed_shell.closed_shell_element,
                       NOT(a_shell\oriented_closed_shell.orientation));
  ELSE
     the_reverse := dummy_tri ||
                   connected_face_set (
                     a_shell\connected_face_set.cfs_faces) ||
               closed_shell () || oriented_closed_shell (a_shell, FALSE);
  END_IF;
  RETURN (the_reverse);
END_FUNCTION;

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;

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;
     IF (NOT('PART_204_BREP_PRODUCT_SCHEMA.PCURVE' IN
           TYPEOF(c\composite_curve.segments[k].parent_curve))) AND
        (NOT('PART_204_BREP_PRODUCT_SCHEMA.SURFACE_CURVE' IN
           TYPEOF(c\composite_curve.segments[k].parent_curve))) AND
     (NOT('PART_204_BREP_PRODUCT_SCHEMA.COMPOSITE_CURVE_ON_SURFACE' IN
           TYPEOF(c\composite_curve.segments[k].parent_curve)))  THEN
       RETURN (FALSE);
     END_IF;
   END_REPEAT;
   RETURN(TRUE);
END_FUNCTION;

FUNCTION constraints_param_b_spline(degree, up_knots, up_cp : INTEGER;
                                        knot_mult : LIST OF INTEGER;
```



```
                        knots : LIST OF parameter_value) : BOOLEAN;
    LOCAL
      result  : BOOLEAN := TRUE;
      k, sum  : INTEGER;
    END_LOCAL;

    (* Find sum of knot multiplicities. *)
    sum := knot_mult[1];
    REPEAT i := 2 TO up_knots;
      sum := sum + knot_mult[i];
    END_REPEAT;
    (* Check limits holding for all B-spline parametrisations *)
    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;
      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;

FUNCTION cross_product (arg1, arg2 : direction) : vector;
  LOCAL
    mag    : REAL;
    res    : direction;
    v1,v2  : LIST[3:3] OF REAL;
    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 := 0.0;
      REPEAT i := 1 TO 3;
        mag := mag + res.direction_ratios[i]*res.direction_ratios[i];
      END_REPEAT;
      IF (mag > 0.0) THEN
        result := dummy_gri || vector(res, SQRT(mag));
      ELSE
        result := dummy_gri || vector(arg1, 0.0);
      END_IF;
      RETURN(result);
    END;
  END_IF;
END_FUNCTION;

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;
     IF b.weights[i] <= 0.0  THEN
       result := FALSE;
       RETURN(result);
     END_IF;
   END_REPEAT;
   RETURN(result);
END_FUNCTION;

FUNCTION derive_dimensional_exponents (x : unit) :
                                    dimensional_exponents;
  LOCAL
    i      : INTEGER;
    result : dimensional_exponents :=
             dimensional_exponents(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
  END_LOCAL;

  IF 'PART_204_BREP_PRODUCT_SCHEMA.DERIVED_UNIT'
              IN TYPEOF(x) THEN -- x is a derived unit
    REPEAT i := LOINDEX(x.elements) TO HIINDEX(x.elements);

      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 -- x is a unitless or a named unit
     result := x.dimensions;
   END_IF;
   RETURN (result);
END_FUNCTION;

FUNCTION dimension_of(item : geometric_representation_item) :
   dimension_count;
   LOCAL
     x   : SET OF representation;
     y   : representation_context;
     dim : dimension_count;
   END_LOCAL;
-- For cartesian_point, direction, or vector dimension is determined by
-- counting components.
     IF 'PART_204_BREP_PRODUCT_SCHEMA.CARTESIAN_POINT' IN
        TYPEOF(item) THEN
        dim := SIZEOF(item\cartesian_point.coordinates);
        RETURN(dim);
     END_IF;
     IF 'PART_204_BREP_PRODUCT_SCHEMA.DIRECTION' IN TYPEOF(item) THEN
        dim := SIZEOF(item\direction.direction_ratios);
        RETURN(dim);
     END_IF;
```

```
    IF 'PART_204_BREP_PRODUCT_SCHEMA.VECTOR' IN TYPEOF(item) THEN
     dim := SIZEOF(item\vector.orientation\direction.direction_ratios);
       RETURN(dim);
    END_IF;
 -- For all other types of geometric_representation_item dim is obtained
  -- via context.
  -- Find the set of representation in which the item is used.
  x := using_representations(item);
  -- Determines the dimension_count of the
  -- geometric_representation_context. Note that the
  -- RULE compatible_dimension ensures that the context_of_items
  -- is of type geometric_representation_context and has
  -- the same dimension_count for all values of x.
  -- The SET x is non-empty since this is required by WR1 of
  -- representation_item.
    y := x[1].context_of_items;
    dim := y\geometric_representation_context.
                coordinate_space_dimension;
    RETURN (dim);
END_FUNCTION;

FUNCTION dimensions_for_si_unit(n : si_unit_name) : dimensional_exponents;
  CASE n  OF
    metre          : RETURN (dimensional_exponents
                             (1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    gram           : RETURN (dimensional_exponents
                             (0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    second         : RETURN (dimensional_exponents
                             (0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0));
    ampere         : RETURN (dimensional_exponents
                             (0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0));
    kelvin         : RETURN (dimensional_exponents
                             (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0));
    mole           : RETURN (dimensional_exponents
                             (0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0));
    candela        : RETURN (dimensional_exponents
                             (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
    radian         : RETURN (dimensional_exponents
                             (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    steradian      : RETURN (dimensional_exponents
                             (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    hertz          : RETURN (dimensional_exponents
                             (0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0));
    newton         : RETURN (dimensional_exponents
                             (1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    pascal         : RETURN (dimensional_exponents
                             (-1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    joule          : RETURN (dimensional_exponents
                             (2.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    watt           : RETURN (dimensional_exponents
                             (2.0, 1.0, -3.0, 0.0, 0.0, 0.0, 0.0));
    coulomb        : RETURN (dimensional_exponents
```



```
                              (0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0));
    volt           : RETURN (dimensional_exponents
                              (2.0, 1.0, -3.0, -1.0, 0.0, 0.0, 0.0));
    farad          : RETURN (dimensional_exponents
                              (-2.0, -1.0, 4.0, 1.0, 0.0, 0.0, 0.0));
    ohm            : RETURN (dimensional_exponents
                              (2.0, 1.0, -3.0, -2.0, 0.0, 0.0, 0.0));
    siemens        : RETURN (dimensional_exponents
                              (-2.0, -1.0, 3.0, 2.0, 0.0, 0.0, 0.0));
    weber          : RETURN (dimensional_exponents
                              (2.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0));
    tesla          : RETURN (dimensional_exponents
                              (0.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0));
    henry          : RETURN (dimensional_exponents
                              (2.0, 1.0, -2.0, -2.0, 0.0, 0.0, 0.0));
    degree_Celsius : RETURN (dimensional_exponents
                              (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0));
    lumen          : RETURN (dimensional_exponents
                              (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
    lux            : RETURN (dimensional_exponents
                              (-2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
    becquerel      : RETURN (dimensional_exponents
                              (0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0));
    gray           : RETURN (dimensional_exponents
                              (2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    sievert        : RETURN (dimensional_exponents
                              (2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0));
  END_CASE;
END_FUNCTION;

FUNCTION dot_product(arg1, arg2 : direction) : REAL;
   LOCAL
     scalar : REAL;
     vec1, vec2: direction;
     ndim : INTEGER;
   END_LOCAL;

   IF NOT EXISTS (arg1) OR NOT EXISTS (arg2) THEN
     scalar := ?;
  (* When function is called with invalid data a NULL result is returned *)
   ELSE
     IF (arg1.dim <> arg2.dim) THEN
       scalar := ?;
  (* When function is called with invalid data a NULL result is returned *)
     ELSE
       BEGIN
         vec1   := normalise(arg1);
         vec2   := normalise(arg2);
         ndim   := arg1.dim;
         scalar := 0.0;
         REPEAT  i := 1 TO ndim;
           scalar := scalar +
```

```
                        vec1.direction_ratios[i]*vec2.direction_ratios[i];
          END_REPEAT;
        END;
      END_IF;
    END_IF;
    RETURN (scalar);
END_FUNCTION;

FUNCTION edge_reversed (an_edge : edge) : oriented_edge;
  LOCAL
    the_reverse : oriented_edge;
  END_LOCAL;

  IF ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_EDGE'
           IN TYPEOF (an_edge) ) THEN
    the_reverse := dummy_tri ||
            edge(an_edge.edge_end, an_edge.edge_start) ||
            oriented_edge(an_edge\oriented_edge.edge_element,
                      NOT (an_edge\oriented_edge.orientation)) ;
  ELSE
    the_reverse := dummy_tri ||
            edge(an_edge.edge_end, an_edge.edge_start) ||
            oriented_edge(an_edge, FALSE);
  END_IF;
  RETURN (the_reverse);
END_FUNCTION;

FUNCTION face_bound_reversed (a_face_bound : face_bound) : face_bound;
  LOCAL
    the_reverse : face_bound ;
  END_LOCAL;
  IF ('PART_204_BREP_PRODUCT_SCHEMA.FACE_OUTER_BOUND'
        IN TYPEOF (a_face_bound) ) THEN
    the_reverse := dummy_tri ||
                    face_bound(a_face_bound\face_bound.bound,
                        NOT (a_face_bound\face_bound.orientation))
                           || face_outer_bound() ;
  ELSE
    the_reverse := dummy_tri ||
     face_bound(a_face_bound.bound, NOT(a_face_bound.orientation));
  END_IF;
 RETURN (the_reverse);
END_FUNCTION;

FUNCTION face_reversed (a_face : face) : oriented_face;
  LOCAL
    the_reverse : oriented_face ;
  END_LOCAL;
  IF ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_FACE'
           IN TYPEOF (a_face) ) THEN
    the_reverse := dummy_tri ||
      face(set_of_topology_reversed(a_face.bounds)) ||
```



```
        oriented_face(a_face\oriented_face.face_element,
                  NOT (a_face\oriented_face.orientation)) ;
  ELSE
    the_reverse := dummy_tri ||
      face(set_of_topology_reversed(a_face.bounds)) ||
                          oriented_face(a_face, FALSE) ;
  END_IF;
     RETURN (the_reverse);
END_FUNCTION;

FUNCTION first_proj_axis(z_axis, arg : direction) : direction;
  LOCAL
    x_axis : direction;
    v      : direction;
    z      : direction;
    x_vec  : vector;
  END_LOCAL;

  IF (NOT EXISTS(z_axis)) THEN
    RETURN (?) ;
  ELSE
    z := normalise(z_axis);
    IF NOT EXISTS(arg) THEN
      IF ((z.direction_ratios <> [1.0,0.0,0.0]) AND
          (z.direction_ratios <> [-1.0,0.0,0.0]))  THEN
        v :=  dummy_gri || direction([1.0,0.0,0.0]);
      ELSE
        v := dummy_gri || direction([0.0,1.0,0.0]);
      END_IF;
    ELSE
      IF  (arg.dim <> 3) THEN
        RETURN (?) ;
      END_IF;
      IF ((cross_product(arg,z).magnitude) = 0.0) 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;

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 'PART_204_BREP_PRODUCT_SCHEMA.PCURVE' IN TYPEOF (c) THEN
    surfs := [c\pcurve.basis_surface];
  ELSE
    IF 'PART_204_BREP_PRODUCT_SCHEMA.SURFACE_CURVE' IN TYPEOF (c) THEN
      n := SIZEOF(c\surface_curve.associated_geometry);
      REPEAT i := 1 TO n;
      surfs := surfs +
             associated_surface(c\surface_curve.associated_geometry[i]);
      END_REPEAT;
    END_IF;
  END_IF;
  IF 'PART_204_BREP_PRODUCT_SCHEMA.COMPOSITE_CURVE_ON_SURFACE'
        IN TYPEOF (c) THEN
(* For a composite_curve_on_surface the basis_surface is the
 intersection of the basis_surfaces of all the segments. *)
     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;
         surfs := surfs * get_basis_surface(c\composite_curve.
                                          segments[i].parent_curve);
       END_REPEAT;
     END_IF;
  END_IF;
  RETURN(surfs);
END_FUNCTION;

FUNCTION item_in_context
  (item  : representation_item;
   cntxt : representation_context) : BOOLEAN;
  LOCAL
    i : INTEGER;
    y : BAG OF representation_item;
  END_LOCAL;
  -- If there is one or more representation using both the item
  -- and cntxt return true.
  IF SIZEOF(USEDIN(item,
          'PART_204_BREP_PRODUCT_SCHEMA.REPRESENTATION.ITEMS')
             * cntxt.representations_in_context) > 0 THEN
    RETURN (TRUE);
    -- Determine the bag of representation_items that reference
    -- item.
    ELSE
    y := QUERY(z <* USEDIN (item , '') |
     'PART_204_BREP_PRODUCT_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z));
     -- Ensure that the set is not empty.
      IF SIZEOF(y) > 0 THEN
      -- For each element in the set
      REPEAT i := 1 TO HIINDEX(y);
        -- Check to see it is an item in the input cntxt.
        IF item_in_context(y[i], cntxt) THEN
```



```
          RETURN (TRUE);
        END_IF;
      END_REPEAT;
    END_IF;
  END_IF;
  -- Return false when all possible branches have been checked
  -- with no success.
  RETURN (FALSE);
END_FUNCTION;

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);
     loops := loops +(f.bounds[i].bound);
   END_REPEAT;
   RETURN(loops);
END_FUNCTION;

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);
     the_reverse := topology_reversed (a_list [i]) + the_reverse;
   END_REPEAT;
   RETURN (the_reverse);
END_FUNCTION;

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;
       res[low+i-1] := lis[i];
     END_REPEAT;
     RETURN(res);
   END_IF;
END_FUNCTION;
```

```
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);
     s := s + l[i];
   END_REPEAT;
   RETURN(s);
END_FUNCTION;

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;
(* Check input dimensions for consistency *)
   IF (u1-low1+1) <> SIZEOF(lis) THEN
     RETURN (?);
   END_IF;
   IF (u2 - low2 + 1 ) <> SIZEOF(lis[1]) THEN
     RETURN (?) ;
   END_IF;
(* Initialise res with values from lis[1] *)
   res := [list_to_array(lis[1], low2, u2) : (u1-low1 + 1)];
   REPEAT i := 2 TO HIINDEX(lis);
     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;

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 := ('PART_204_BREP_PRODUCT_SCHEMA.POLY_LOOP'
        IN TYPEOF(l[1]));
   REPEAT i := 2 TO SIZEOF(l);
     IF(('PART_204_BREP_PRODUCT_SCHEMA.POLY_LOOP' IN
           TYPEOF(l[i])) <> poly_loop_type) THEN
         RETURN(TRUE);
      END_IF;
   END_REPEAT;
   RETURN(FALSE);
END_FUNCTION;
```



```
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;

FUNCTION normalise (arg : vector_or_direction) : vector_or_direction;
   LOCAL
     ndim   : INTEGER;
     v      : direction;
     result : vector_or_direction;
     vec    : vector;
     mag    : REAL;
   END_LOCAL;
   IF NOT EXISTS (arg) THEN
     result := ?;
(* When function is called with invalid data a NULL result is returned *)
   ELSE
     ndim := arg.dim;
     IF 'PART_204_BREP_PRODUCT_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
       BEGIN
        v := dummy_gri || direction(arg.orientation.direction_ratios);
         IF arg.magnitude = 0.0 THEN
           RETURN(?);
         ELSE
          vec := dummy_gri || vector (v, 1.0);
         END_IF;
       END;
     ELSE
       v := dummy_gri || direction (arg.direction_ratios);
     END_IF;
     mag := 0.0;
     REPEAT  i := 1 TO ndim;
       mag := mag + v.direction_ratios[i]*v.direction_ratios[i];
     END_REPEAT;
     IF mag > 0.0 THEN
       mag := SQRT(mag);
       REPEAT  i := 1 TO ndim;
         v.direction_ratios[i] := v.direction_ratios[i]/mag;
       END_REPEAT;
       IF 'PART_204_BREP_PRODUCT_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
         vec.orientation := v;
         result := vec;
       ELSE
         result := v;
       END_IF;
     ELSE
```

```
      RETURN(?);
    END_IF;
  END_IF;
  RETURN (result);
END_FUNCTION;

FUNCTION open_shell_reversed ( a_shell : open_shell) :
                                        oriented_open_shell;
  LOCAL
    the_reverse : oriented_open_shell;
  END_LOCAL;
  IF ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_OPEN_SHELL'
                   IN TYPEOF (a_shell) ) THEN
    the_reverse := dummy_tri ||
                   connected_face_set (
                       a_shell\connected_face_set.cfs_faces) ||
                   open_shell () || oriented_open_shell(
                     a_shell\oriented_open_shell.open_shell_element,
                       (NOT (a_shell\oriented_open_shell.orientation)));
  ELSE
    the_reverse := dummy_tri ||
                   connected_face_set (
                       a_shell\connected_face_set.cfs_faces) ||
                   open_shell () ||  oriented_open_shell (a_shell, FALSE);
  END_IF;
  RETURN (the_reverse);
END_FUNCTION;

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;

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;
      p := p AND (a_path.edge_list[i-1].edge_end :=:
                  a_path.edge_list[i].edge_start);
    END_REPEAT;
    RETURN (p);
```



```
END_FUNCTION;

FUNCTION path_reversed (a_path : path) : oriented_path;
  LOCAL
    the_reverse : oriented_path ;
  END_LOCAL;
  IF ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_PATH'
        IN TYPEOF (a_path) ) THEN
    the_reverse := dummy_tri ||
       path(list_of_topology_reversed (a_path.edge_list)) ||
          oriented_path(a_path\oriented_path.path_element,
                      NOT(a_path\oriented_path.orientation)) ;
  ELSE
    the_reverse := dummy_tri ||
           path(list_of_topology_reversed (a_path.edge_list)) ||
                       oriented_path(a_path, FALSE);
  END_IF;
  RETURN (the_reverse);
END_FUNCTION;

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 'PART_204_BREP_PRODUCT_SCHEMA.VECTOR' IN TYPEOF (vec) THEN

      v   := dummy_gri || direction(vec.orientation.direction_ratios);
      mag := scalar * vec.magnitude;
    ELSE
      v   := dummy_gri || direction(vec.direction_ratios);
      mag := scalar;
    END_IF;
    IF (mag < 0.0 ) THEN
      REPEAT i := 1 TO SIZEOF(v.direction_ratios);
        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;

FUNCTION second_proj_axis(z_axis, x_axis, arg: direction) : direction;
  LOCAL
```

```
    y_axis : vector;
    v      : direction;
    temp   : vector;
  END_LOCAL;

  IF NOT EXISTS(arg) THEN
    v := dummy_gri || direction([0.0,1.0,0.0]);
  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;

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);
     the_reverse := the_reverse + topology_reversed (a_set [i]);
   END_REPEAT;
 RETURN (the_reverse);
END_FUNCTION;

FUNCTION shell_reversed (a_shell : shell) : shell;
  IF ('PART_204_BREP_PRODUCT_SCHEMA.OPEN_SHELL' IN TYPEOF (a_shell) ) THEN
    RETURN (open_shell_reversed (a_shell));
  ELSE
    IF ('PART_204_BREP_PRODUCT_SCHEMA.CLOSED_SHELL' IN
                            TYPEOF (a_shell) ) THEN
      RETURN (closed_shell_reversed (a_shell));
    ELSE
      RETURN (?);
    END_IF;
  END_IF;
END_FUNCTION;

FUNCTION surface_weights_positive(b: rational_b_spline_surface) :
                                                              BOOLEAN;
   LOCAL
     result        : BOOLEAN := TRUE;
   END_LOCAL;
   REPEAT i := 0 TO b.u_upper;
     REPEAT j := 0 TO b.v_upper;
```



```
        IF (b.weights[i][j] <= 0.0)  THEN
          result := FALSE;
          RETURN(result);
        END_IF;
      END_REPEAT;
    END_REPEAT;
    RETURN(result);
END_FUNCTION;

FUNCTION topology_reversed (an_item : reversible_topology)
                                    : reversible_topology;

    IF ('PART_204_BREP_PRODUCT_SCHEMA.EDGE' IN TYPEOF (an_item)) THEN
      RETURN (edge_reversed (an_item));
    END_IF;
    IF ('PART_204_BREP_PRODUCT_SCHEMA.PATH' IN TYPEOF (an_item)) THEN
      RETURN (path_reversed (an_item));
    END_IF;
    IF ('PART_204_BREP_PRODUCT_SCHEMA.FACE_BOUND'
          IN TYPEOF (an_item)) THEN
      RETURN (face_bound_reversed (an_item));
    END_IF;
    IF ('PART_204_BREP_PRODUCT_SCHEMA.FACE' IN TYPEOF (an_item)) THEN
      RETURN (face_reversed (an_item));
    END_IF;
    IF ('PART_204_BREP_PRODUCT_SCHEMA.SHELL' IN TYPEOF (an_item)) THEN
      RETURN (shell_reversed (an_item));
    END_IF;
    IF ('SET' IN TYPEOF (an_item)) THEN
      RETURN (set_of_topology_reversed (an_item));
    END_IF;
    IF ('LIST' IN TYPEOF (an_item)) THEN
      RETURN (list_of_topology_reversed (an_item));
    END_IF;
    RETURN (?);
END_FUNCTION;

FUNCTION using_items (item : founded_item_select;
                      checked_items: SET OF founded_item_select)
                    : SET OF founded_item_select;
  LOCAL
    new_check_items    : SET OF founded_item_select;
    result_items       : SET OF founded_item_select;
    next_items         : SET OF founded_item_select;
  END_LOCAL;
  result_items := [];
  new_check_items := checked_items + item;
  -- Find the set of representation_items or founded_items
  -- in which item is used directly.
  next_items := QUERY(z <* bag_to_set( USEDIN(item , '')) |
    ('PART_204_BREP_PRODUCT_SCHEMA.REPRESENTATION_ITEM'
                                        IN TYPEOF(z)) OR
```

```
    ('PART_204_BREP_PRODUCT_SCHEMA.FOUNDED_ITEM'
          IN TYPEOF(z)));
      -- If the set of next_items is not empty;
  IF SIZEOF(next_items) > 0 THEN
    -- For each element in the set, find the using_items recursively
    REPEAT i := 1 TO HIINDEX(next_items);
      -- Check for loop in data model, i.e. one of the next_items
      -- occurred earlier in the set of check_items;
      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 the set of representation_items or founded_items
  -- in which the input item is used directly and indirectly.
  RETURN (result_items);
END_FUNCTION;

FUNCTION using_representations (item : founded_item_select)
  : SET OF representation;
  LOCAL
    results            : SET OF representation;
    result_bag         : BAG OF representation;
    intermediate_items : SET OF founded_item_select;
  END_LOCAL;
  -- Find the representations in which the item is used and add to the
  -- results set.
  results := [];
  result_bag :=
USEDIN(item,'PART_204_BREP_PRODUCT_SCHEMA.REPRESENTATION.ITEMS');
  IF SIZEOF(result_bag) > 0 THEN
    REPEAT i := 1 TO HIINDEX(result_bag);
      results := results + result_bag[i];
    END_REPEAT;
  END_IF;
  -- Find all representation_items or founded_items
  -- by which item is referenced directly or indirectly.
  intermediate_items := using_items(item,[]);
  -- If the set of intermediate items is not empty;
  IF SIZEOF(intermediate_items) > 0 THEN
    -- For each element in the set, add the
    -- representations of that element.
    REPEAT i := 1 TO HIINDEX(intermediate_items);
      result_bag := USEDIN(intermediate_items[i],
              'PART_204_BREP_PRODUCT_SCHEMA.REPRESENTATION.ITEMS');

      IF SIZEOF(result_bag) > 0 THEN
        REPEAT j := 1 TO HIINDEX(result_bag);
          results := results + result_bag[j];
        END_REPEAT;
      END_IF;
```



```
    END_REPEAT;
  END_IF;
  -- Return the set of representation in which the input item is
  -- used directly and indirectly (through intervening
  -- representation_items or founded items).
  RETURN (results);
END_FUNCTION;
 FUNCTION valid_units ( m : measure_with_unit ) : BOOLEAN ;
  IF 'PART_204_BREP_PRODUCT_SCHEMA.LENGTH_MEASURE'
              IN TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
 dimensional_exponents ( 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;
  IF 'PART_204_BREP_PRODUCT_SCHEMA.MASS_MEASURE' IN
        TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
   dimensional_exponents ( 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;
  IF 'PART_204_BREP_PRODUCT_SCHEMA.TIME_MEASURE'
            IN TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
  dimensional_exponents ( 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;
 IF 'PART_204_BREP_PRODUCT_SCHEMA.ELECTRIC_CURRENT_MEASURE'
          IN TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
 dimensional_exponents ( 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;
  IF 'PART_204_BREP_PRODUCT_SCHEMA.THERMODYNAMIC_TEMPERATURE_MEASURE'
  IN TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
   dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;
  IF 'PART_204_BREP_PRODUCT_SCHEMA.AMOUNT_OF_SUBSTANCE_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
    IF derive_dimensional_exponents ( m.unit_component ) <>
  dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0 ) THEN
      RETURN (FALSE);
    END_IF;
  END_IF;
  IF 'PART_204_BREP_PRODUCT_SCHEMA.LUMINOUS_INTENSITY_MEASURE'
       IN TYPEOF ( m.value_component ) THEN
```

```
  IF derive_dimensional_exponents ( m.unit_component ) <>
dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'PART_204_BREP_PRODUCT_SCHEMA.PLANE_ANGLE_MEASURE'
           IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'PART_204_BREP_PRODUCT_SCHEMA.SOLID_ANGLE_MEASURE'
      IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
 dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'PART_204_BREP_PRODUCT_SCHEMA.AREA_MEASURE'
         IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
 dimensional_exponents ( 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'PART_204_BREP_PRODUCT_SCHEMA.VOLUME_MEASURE'
           IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
 dimensional_exponents ( 3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'PART_204_BREP_PRODUCT_SCHEMA.RATIO_MEASURE'
       IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
 dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;
IF 'PART_204_BREP_PRODUCT_SCHEMA.POSITIVE_LENGTH_MEASURE'
     IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
 dimensional_exponents ( 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
    RETURN (FALSE);
  END_IF;
END_IF;

IF 'PART_204_BREP_PRODUCT_SCHEMA.POSITIVE_PLANE_ANGLE_MEASURE'
       IN TYPEOF ( m.value_component ) THEN
  IF derive_dimensional_exponents ( m.unit_component ) <>
 dimensional_exponents ( 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ) THEN
```



```
      RETURN (FALSE);
    END_IF;
  END_IF;
  RETURN (TRUE);
END_FUNCTION;

FUNCTION vector_difference(arg1, arg2 : vector_or_direction) : vector;
  LOCAL
    result          : vector;
    res, vec1, vec2 : direction;
    mag, mag1, mag2 : REAL;
    ndim            : INTEGER;
  END_LOCAL;

  IF ((NOT EXISTS (arg1)) OR (NOT EXISTS (arg2))) OR
           (arg1.dim <> arg2.dim)
      THEN
    RETURN (?) ;
   ELSE
    BEGIN
      IF 'PART_204_BREP_PRODUCT_SCHEMA.VECTOR' IN TYPEOF(arg1) THEN
        mag1 := arg1.magnitude;
        vec1 := arg1.orientation;
      ELSE
        mag1 := 1.0;
        vec1 := arg1;
      END_IF;
      IF 'PART_204_BREP_PRODUCT_SCHEMA.VECTOR' IN TYPEOF(arg2) THEN

        mag2 := arg2.magnitude;
        vec2 := arg2.orientation;
      ELSE
        mag2 := 1.0;
        vec2 := arg2;
      END_IF;
      vec1 := normalise (vec1);
      vec2 := normalise (vec2);
      ndim := SIZEOF(vec1.direction_ratios);
      mag := 0.0;
      res := dummy_gri || direction(vec1.direction_ratios);
      REPEAT i := 1 TO ndim;
        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 > 0.0 ) THEN
      result := dummy_gri || vector( res, SQRT(mag));
      ELSE
        result := dummy_gri || vector( vec1,  0.0);
      END_IF;
    END;
  END_IF;
```

```
  RETURN (result);
END_FUNCTION;

END_SCHEMA; -- part_204_brep_product_schema
(*
```


Annex B
(normative)

AIM short names

Table B.1 provides the short names of entities specified in the AIM of this part of ISO 10303. Requirements on the use of the short names are found in the implementation methods included in ISO 10303.

Table B.1 – AIM short names of entities

Entity data types names	Short names
ADVANCED_BREP_SHAPE_REPRESENTATION	ABSR
ADVANCED_FACE	ADVFAC
APPLICATION_CONTEXT	APPCNT
APPLICATION_CONTEXT_ELEMENT	APCNEL
ASSEMBLY_COMPONENT_USAGE	ASCMUS
AXIS1_PLACEMENT	AX1PLC
AXIS2_PLACEMENT_2D	A2PL2D
AXIS2_PLACEMENT_3D	A2PL3D
BACKGROUND_COLOUR	BCKCLR
BEZIER_CURVE	BZRCRV
BEZIER_SURFACE	BZRSRF
BOUNDED_CURVE	BNDCR
BOUNDED_SURFACE	BNDSRF
BREP_WITH_VOIDS	BRWTVD
B_SPLINE_CURVE	BSPCR
B_SPLINE_CURVE_WITH_KNOTS	BSCWK
B_SPLINE_SURFACE	BSPSR
B_SPLINE_SURFACE_WITH_KNOTS	BSSWK
CAMERA_IMAGE	CMRIMG
CAMERA_IMAGE_3D_WITH_SCALE	CI3WS
CAMERA_MODEL	CMRMDL
CAMERA_MODEL_D3	CMMDD3
CAMERA_MODEL_D3_WITH_HLSR	CMDWH

Table B.1 (continued)

Entity name	Short name
CAMERA_MODEL_WITH_LIGHT_SOURCES	CMWLS
CARTESIAN_POINT	CRTPNT
CARTESIAN_TRANSFORMATION_OPERATOR	CRTROP
CARTESIAN_TRANSFORMATION_OPERATOR_3D	CTO3
CIRCLE	CIRCLE
CLOSED_SHELL	CLSSHL
COLOUR	COLOUR
COLOUR_RGB	CLRRGB
COLOUR_SPECIFICATION	CLRSPC
CONIC	CONIC
CONICAL_SURFACE	CNCSRF
CONNECTED_FACE_SET	CNFCST
CONVERSION_BASED_UNIT	CNBSUN
CURVE	CURVE
CURVE_STYLE	CRVSTY
CURVE_STYLE_FONT	CRSTFN
CURVE_STYLE_FONT_PATTERN	CSFP
CURVE_STYLE_RENDERING	CRSTRN
CYLINDRICAL_SURFACE	CYLSRF
DESIGN_CONTEXT	DSGCNT
DIRECTION	DRCTN
DRAUGHTING_PRE_DEFINED_COLOUR	DPDC
DRAUGHTING_PRE_DEFINED_CURVE_FONT	DPDCF
EDGE	EDGE
EDGE_CURVE	EDGCRV
EDGE_LOOP	EDGLP
ELEMENTARY_BREP_SHAPE_REPRESENTATION	EBSR

Table B.1 (continued)

Entity name	Short name
ELEMENTARY_SURFACE	ELMSRF
ELLIPSE	ELLPS
FACE	FACE
FACETED_BREP	FCTBRP
FACETED_BREP_SHAPE_REPRESENTATION	FBSR
FACE_BOUND	FCBND
FACE_OUTER_BOUND	FCOTBN
FACE_SURFACE	FCSRF
FILL_AREA_STYLE_COLOUR	FASC
FUNCTIONALLY_DEFINED_TRANSFORMATION	FNDFTR
GEOMETRIC_REPRESENTATION_CONTEXT	GMRPCN
GEOMETRIC_REPRESENTATION_ITEM	GMRPIT
GLOBAL_UNIT_ASSIGNED_CONTEXT	GUAC
HYPERBOLA	HYPRBL
INVISIBILITY	INVSBL
LENGTH_MEASURE_WITH_UNIT	LMWU
LENGTH_UNIT	LNGUNT
LIGHT_SOURCE	LGHSRC
LIGHT_SOURCE_AMBIENT	LGSRAM
LIGHT_SOURCE_DIRECTIONAL	LGSRDR
LIGHT_SOURCE_POSITIONAL	LGSRPS
LIGHT_SOURCE_SPOT	LGSRSP
LINE	LINE
LOOP	LOOP
MANIFOLD_SOLID_BREP	MNSLBR
MAPPED_ITEM	MPPITM
MECHANICAL_CONTEXT	MCHCNT

Table B.1 (continued)

Entity name	Short name
MECHANICAL_DESIGN_PRE_DEFINED_TEXT_FONT	MDPDTF
MECHANICAL_DESIGN_PRE_DEFINED_TEXT_STYLE	MDPDTS
MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_AREA	MDGPA
MECHANICAL_DESIGN_GEOMETRIC_PRESENTATION_REPRESENTATION	MDGPR
MECHANICAL_DESIGN_SHADED_PRESENTATION_AREA	MDSPA
MECHANICAL_DESIGN_SHADED_PRESENTATION_REPRESENTATION	MDSPR
MEASURE_WITH_UNIT	MSWTUN
NAMED_UNIT	NMDUNT
OPEN_SHELL	OPNSHL
ORIENTED_CLOSED_SHELL	ORCLSH
ORIENTED_EDGE	ORNEDG
ORIENTED_FACE	ORNFC
PARABOLA	PRBL
PATH	PATH
PCURVE	PCURVE
PLACEMENT	PLCMNT
PLANAR_BOX	PLNBX
PLANAR_EXTENT	PLNEXT
PLANE	PLANE
PLANE_ANGLE_MEASURE_WITH_UNIT	PAMWU
PLANE_ANGLE_UNIT	PLANUN
POINT	POINT
POINT_STYLE	PNTSTY
POLYLINE	PLYLN
POLY_LOOP	PLYLP
PRESENTATION_AREA	PRSAR
PRESENTATION_REPRESENTATION	PRSRPR

Table B.1 (continued)

Entity name	Short name
PRESENTATION_SIZE	PRSSZ
PRESENTATION_STYLE_ASSIGNMENT	PRSTAS
PRESENTATION_STYLE_BY_CONTEXT	PSBC
PRE_DEFINED_COLOUR	PRDFCL
PRE_DEFINED_ITEM	PRDFIT
PRE_DEFINED_CURVE_FONT	PDCF
PRODUCT	PRDCT
PRODUCT_CONTEXT	PRDCNT
PRODUCT_DATA_REPRESENTATION_VIEW	PDRV
PRODUCT_DATA_REPRESENTATION_VIEW_WITH_HLHSR	PDRVWH
PRODUCT_DEFINITION	PRDDFN
PRODUCT_DEFINITION_CONTEXT	PRDFCN
PRODUCT_DEFINITION_RELATIONSHIP	PRDFRL
PRODUCT_DEFINITION_SHAPE	PRDFSH
PRODUCT_DEFINITION_USAGE	PRDFUS
PRODUCT_VERSION	PRDVRS
QUASI_UNIFORM_CURVE	QSUNCR
QUASI_UNIFORM_SURFACE	QSUNSR
RATIONAL_B_SPLINE_CURVE	RBSC
RATIONAL_B_SPLINE_SURFACE	RBSS
REPRESENTATION	RPRSNT
REPRESENTATION_CONTEXT	RPRCNT
REPRESENTATION_DEPENDENT_STYLED_ITEM	RDSI
REPRESENTATION_ITEM	RPRITM
REPRESENTATION_MAP	RPRMP
SHAPE_DEFINITION_REPRESENTATION	SHDFRP
SHAPE_REPRESENTATION	SHPRPR

Table B.1 (continued)

Entity name	Short name
SI_UNIT	SUNT
SOLID_MODEL	SLDMDL
STYLED_ITEM	STYITM
SURFACE	SRFC
SURFACE_CURVE	SRFCRV
SURFACE_OF_LINEAR_EXTRUSION	SL
SURFACE_OF_REVOLUTION	SROFRV
SURFACE_RENDERING_PROPERTIES	SRRNPR
SURFACE_SIDE_STYLE	SRSDST
SURFACE_STYLE_BOUNDARY	SRSTBN
SURFACE_STYLE_CONTROL_GRID	SSCG
SURFACE_STYLE_FILL_AREA	SSFA
SURFACE_STYLE_PARAMETER_LINE	SSPL
SURFACE_STYLE_REFLECTANCE_AMBIENT	SSRA
SURFACE_STYLE_REFLECTANCE_AMBIENT_DIFFUSE	SSRAD
SURFACE_STYLE_REFLECTANCE_AMBIENT_DIFFUSE_SPECULAR	SSRADS
SURFACE_STYLE_RENDERING	SRSTRN
SURFACE_STYLE_RENDERING_WITH_PROPERTIES	SSRWP
SURFACE_STYLE_SEGMENTATION_CURVE	SSSC
SURFACE_STYLE_SILHOUETTE	SRSTSL
SURFACE_STYLE_TRANSPARENT	SRSTTR
SURFACE_STYLE_USAGE	SRSTUS
SWEPT_SURFACE	SWPSRF
TOPOLOGICAL_REPRESENTATION_ITEM	TPRPIT
TOROIDAL_SURFACE	TRDSRF
UNIFORM_CURVE	UNFCRV

Table B.1 (concluded)

Entity name	Short name
UNIFORM_SURFACE	UNFSRF
VECTOR	VECTOR
VERTEX	VERTEX
VERTEX_LOOP	VRTLP
VERTEX_POINT	VRTPNT
VIEW_VOLUME	VWVLM

Annex C
(normative)

Implementation method specific requirements

The primary implementation form supported by this AP is an exchange structure implementation using the exchange structure specified in ISO 10303-21. The file format shall be encoded according to the syntax and EXPRESS language mapping defined in ISO 10303-21 and in the AIM defined in annex A of this part of ISO 10303.

The header section of an exchange structure conforming to ISO 10303-21 shall identify the use of this part of ISO 10303 by the schema name **part_204_brep_product_schema** and the conformance class of the implementation shall be given to identify the types of B-rep model represented in the file.

Other forms of implementation, such as the Standard Data Access Interface (SDAI) conforming to ISO 10303-22, may be supported but no specific requirements are given for these.

Annex D
(normative)

PICS (Protocol Implementation Conformance Statement) proforma

This clause lists the optional elements of this part of ISO 10303. An implementation may choose to support any combination of these optional elements. However, certain combinations of options are likely to be implemented together. These combinations are called conformance classes and are described in the subclauses of this annex.

This annex is in the form of a questionnaire. This questionnaire is intended to be filled out by the implementor and may be used in preparation for conformance testing by a testing laboratory. The completed PICS proforma is referred to as a PICS.

The information in the PICS may be used to identify the appropriate elements of the abstract test suite for use in a conformance assessment test campaign.

Three conformance classes are identified in this part of ISO 10303, each of these classes contains two options. A conforming implementation shall support one of these classes and may support one, or two, of the optional additions. These classes are specified in clause 6 of ISO 10303-204.

Questionnaire: 1. Please provide an identifier for the product or system for which conformance is claimed.
Product name and version identification:

2. Please indicate the implementation method:

____ ISO 10303-21 Exchange structure - preprocessor, Preprocessor version identification:

____ ISO 10303-21 Exchange structure - postprocessor, Postprocessor version identification:

____ ISO 10303-22 Standard Data Access Interface SDAI version identification:

3.Please indicate the classes and options for which conformance is claimed:

____ Class 1: B-rep level 1 (faceted B-rep).
____ With preservation of user defined names.
____ With visual presentation for B-reps.

____ Class 2: B-rep level 2 (elementary B-rep).
____ With preservation of user defined names.
____ With visual presentation for B-reps.

____ Class 2: B-rep level 3 (advanced B-rep).
____ With preservation of user defined names.
____ With visual presentation for B-reps.

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(204) 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 part_204_brep_product_schema in an open information system, the object identifier

{ iso standard 10303 part(204) version(0) schema(1) part-204-B-rep-product-schema(1) }

is assigned to the part_204_brep_product_schema 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 (AAM)

The application activity model (AAM) is provided as an aid in understanding the scope and information requirements defined in this application protocol. The model is presented as a set of figures that contain the activity diagrams and a set of definitions of the activities and their data. The application activity model is given in figure F.1. Activities and data flows that are out of scope are marked with an asterisk.

The viewpoint of the application activity model is the use and exchange of boundary representation models in the mechanical engineering environment.

F.1 AAM definitions

The following terms are used in the application activity model. Terms marked with an asterisk are outside the scope of this application protocol.

The definitions given in this annex do not supersede the definitions given in the main body of the text.

F.1.1 CAD model:

The computer internal description of product shape produced by a computer-aided design system.

F.1.2 Calculation:*

The process of performing a mathematical computation.

F.1.3 Calculation system*:

A system which produces a mathematical model of some aspect of the physical structure or behaviour of a product.

F.1.4 Concept development*:

The process of producing the conceptual design.

F.1.5 Conceptual change request*:

A request for a change in the conceptual design of a product.

F.1.6 Conceptual design*:

The preliminary definition of the product with respect to technical, economic, ecological and strategic requirements.

F.1.7 Definitional change request*:

A request for a change in the definition of a product.

F.1.8 Design

The development of the shape of a product taking account of its functionality and physical properties.

F.1.9 Design change request:

A formal request to change any form, fit, or function aspect of an existing design.

F.1.10 Development tool:*

A computer tool which assists in the development of a product definition.

F.1.11 Evaluation*:

The testing and validation of a product design, or the prototype of a product, against its requirements.

F.1.12 FEA system *

A computer system for the analysis of CAD models with respect to such properties as stress, dynamic behaviour, or heat transfer using the finite element method.

F.1.13 Holography*:

A process for producing three dimensional images using lasers.

F.1.14 Knowhow*:

The knowledge and experience of the people involved in the development process.

F.1.15 Machine tool*:

A mechanical device, usually for cutting metal, used in the manufacturing process.

F.1.16 Machine control code*:

A computer sensible code controlling a machine tool.

F.1.17 Management*:

The process of controlling the use of manpower and resources.

F.1.18 Manpower*:

The people, or human resources, involved in the process.

F.1.19 Manufacturing*:

The realisation of the actual product.

F.1.20 Marketing strategy*:

The planned method for selling the product.

F.1.21 Material*:

The matter from which the product is manufactured.

F.1.22 Measured data*:

Data which is obtained by measuring a product or physical model.

F.1.23 NC-machining*:

A manufacturing process of material removal using a numerically controlled machine tool.

F.1.24 Part design:

The process of defining the form, fit, and function of a part.

F.1.25 Physical model*:

A representation of the product or part made of some material substance.

F.1.26 Planning tools*:

Methods and equipment for the support of planning tasks, e.g. a process planning system.

F.1.27 Product definition*:

The complete specicification of a product including shape, size, geometric configuration, and materials.

F.1.28 Prototyping*:

The process of producing a first example of a product for evaluation.

F.1.29 Product plan*:

All available information about the manufacturing process for a product, including the sequence of manufacturing operations.

F.1.30 Product requirement specification*:

The collection of all known requirements and constraints which specify the functionality and characteristics of a product.

F.1.31 Product specification:

The description of the characteristics of a product.

F.1.32 Production planning*:

The process of producing the product plan.

F.1.33 Quality assurance*:

The process of ensuring that the product meets its specification.

F.1.34 Shape design:

The process of developing the precise specification of the geometric form of a product.

F.1.35 Sketch*:

An approximate, possibly hand-drawn, presentation of the form of a product.

F.1.36 Simulation*:

The imitation of a process, using a physical model or using a computer model.

F.1.37 Standards*:

Formal documents, which may be international, national, or company-specific, which influence the design process to ensure commonality between products.

F.1.38 Stereolithography*:

A method of producing three dimensional physical models of a product.

F.1.39 Test plan*:

A plan for testing a product design, or prototype, to ensure that all requirements are met.

F.1.40 Visualisation:

The pictorial presentation of a product.

F.1.41 Visualisation system*:

A system which processes the product description data into a pictorial presentation.

F.2 Description of AAM scenario

Mechanical engineering design and manufacturing use many different

CAD and CAE and CIM systems. Different heterogeneous CAD models are inherent in these systems. The data exchange between those systems requires a neutral standard rather than system-dependent models or methods. In the following subclauses the mechanical design requirements are described.

F.2.1 Mechanical Design Requirements for Model Exchange

The different groups involved in the design, engineering, or manufacturing processes need to have access to the design data. The basic geometric form of a product is fixed during the conceptual design of the product. The details of the product shape may be added by a design group using a CAD system. The design represented in a CAD model may be transferred from system to system by means of a neutral or standard file. The transfer medium might be an electronic network (i.e. LAN, WAN), physical media, or some form of shared database.

The CAD model data is transferred to other groups within the company or transferred to another company for analysis or manufacturing purposes. The following scenarios should be considered for model exchange.

F.2.1.1 Design to Product Analysis and Simulation

NOTE 1 See nodes A3 and A4 in diagrams

Analysis and simulation of the product are used at different stages of the development process for early testing and validation of the design. Product data is received from the design groups via data transfer of CAD models and manufacturing data (e.g., NC programmes). The results of the prototype evaluation process (e.g., test protocols or requirements) are returned to the design department and may lead to a new iteration. The types of analysis performed during the validation process are dependent upon the functionality of the product and may include stress analysis, thermal analysis, or aerodynamic analysis on a given mechanical design.

F.2.1.2 Design to Design groups

A complex product may be designed with parts from different design and manufacturing sources. The product may be an assembly of designs from different companies. In this case the designs of individual parts may be exchanged between the design companies. A contractor may transfer to a subcontractor a 'hull of a design' which indicates the outer boundaries of parts in detail at only specific interfacing points but not the complete shape of the parts (e.g., a dimensioned box for a motor block with some details for the bearings). This scenario requires the transfer of the exact shape model.

F.2.1.3 Design to Prototype Manufacturing

NOTE 1 See nodes A31, A32 and A43 in diagrams

Prototypes of the product are used at different stages of the development process for testing and validation of the design. The necessary data is received from the design departments via the data transfer of CAD models and manufacturing data (NC programmes etc.). The results of the prototype evaluation process (test protocols, requirements, etc.) are returned to the design department and may lead to a new iteration of the design and validation process.

F.2.1.4 Design to Manufacturing Planning and Manufacturing

NOTE 1 See nodes A5 and A6 in diagrams

The manufacturing planning department prepares the detailed manufacturing process plan in order to manufacture a product. If the part is to be manufactured with metal cutting machinery then the appropriate technology (milling, drilling, turning, and grinding) is selected, and numerical controlled (NC) machine tool programs are set up. The data are derived from the CAD model (required data: geometry, topology, material, administrative data). The NC tool paths are generated in the NC programming system; collision checks with the machine tools and tool path simulations are performed. The result is a NC machine program which can be 'downloaded' to the NC machine. A similar procedure is performed if robots and robot programming systems are applied (e.g., for material supply to NC machines).

F.2.1.5 Design to Assembly Planning

In cases where robots are applied to assemble the products, a robot simulation program might be applied in order to visualize the assembly process prior to the physical assembly process. Assembly path simulations and collision checks are performed based on geometric assembly paths which are typically generated interactively in a simulation workstation. Final-assembly-fitting simulation is performed on given shapes of parts.

F.2.1.6 Design to Quality Assurance

Quality assurance today is performed in parallel to all design, analysis, and manufacturing processes. The design data are used as reference data for any quality assurance in subsequent processes after the design.

EXAMPLE 1 The design model is the basis for analysis (e.g., stress) by means of finite element models. The analysis results give data for various means of assuring quality (e.g., modification of the shape of a design).

EXAMPLE 2 The design model is a reference for manufacturing inspection (e.g., when applying a coordinate-measurement machine after a mechanical part is machined by a milling machine). If there is a deviation beyond a certain tolerance limit, the manufactured part is not acceptable.

F.2.1.7 Design to documentation

The design master model, resident in a computer and based on topology, geometry, and other associated information, is used to derive CAD drawing documentation. The 2D CAD documentation typically consists of orthogonal projections of the 3D model to 2D space: the results are 2D CAD drawings. These contain the geometry of the designed parts, explanatory text, dimensioning, hatching for sectioned areas, and administrative information. The sum of the design data is typically kept in a data management system.

F.2.1.8 Feedback to the design department

The initial design of a part which will be manufactured is typically not right the first time. Therefore model data typically have to be transferred back and modified in the design department after they have been transferred either to the manufacturing process planning or to the part analysis engineering departments. These departments give feedback to the design office in the form of change requests relating to the original CAD model data file. The process described above can, of course, be handled via a data base system which could be accessible from different departments. Other intermediate engineering or pre-engineering organisational entities might be involved in the design process and in its feedback loop.

F.3 Mechanical design requirements for model contents and completeness

The Mechanical Design area has a very large scope which is intentionally limited here to essential information blocks. Figure F.1 gives an overview on the conceptual structure of a Mechanical Design Product.

A mechanical design product model typically is composed of the following data:

F.3.1 Shape description (Geometry and topology)

The shape description of part models can have different levels of completeness and constraints; therefore, different model descriptions are applied today:

— A: Volume-based design (B-rep or CSG);

— B: Surface-based design;

— C: Wireframe-based design (curves).

The different geometric descriptions typically do not all exist simultaneously for one product; rather, they are set up during the design process in one of the above mentioned forms.

F.3.2 Draughting Data

Another kind of representation of a product is the classical technology of a drawing on paper. This is still the most commonly used methodology for design, manufacturing, and archiving of product data. The drawings can be produced in a number of different ways:

— by using a pen and paper

— by using a pure 2D CAD draughting system

— from a 3D CAD model projected into 2D drawing space;
the result may be a simple drawing or a drawing with associative properties.

F.3.3 Physical property description

There are a variety of physical properties which could be attached to a part model; essential amongst these are the material properties. Material properties may be determined by a designer quite late in the phase of detail design. Each model part in this Application Protocol assumes the use of one homogeneous material. There might be other physical properties (e.g., surface roughness) assigned to parts, but these are beyond the scope of this Application Protocol. The material property is included as representative of other properties which could be included later.

F.3.4 Part administrative data

This contains an aggregation of administrative data for a specific part. The administrative description has a reference to the shape description and might also have a reference to the material description.

F.3.5 Assemblies

Assemblies are composed of either sub-assemblies and parts, or, of at least two parts. An assembly is typically represented as a tree of parts and sub-assemblies. The relationship between assemblies and sub-assemblies or parts contains positional and orientational information in three-dimensional space.

F.3.6 Product Information Description

This section covers basic administrative data for managing the product through the design process and for the use of other departments in a manufacturing engineering company.

EXAMPLE 1 identification of the product (name);

EXAMPLE 2 ownership (company, department, designer,manager);

EXAMPLE 3 history (date of design,updates, release status, revisions).

F.3.7 Industry segments in which volume-based design is applied

Below is a list of products which can be designed very effectively with solid modelling. This list is compiled in industry sectors.

Machinery industry:

This industry manufactures and assembles manufacturing machines for different target markets.

EXAMPLE 1 Companies manufacturing machines for:

— metal cutting (e.g., lathes, milling machines, grinding machines);

— metal forming (e.g., presses and dies);

— assembly lines (e.g., robots);

— plastic moulding (e.g., plastic injection machines);

— welding constructions (e.g. welding robots);

Electro-mechanical manufacturing industry:

This industry manufactures and assembles electro-mechanical equipment for industrial use and for the consumer market.

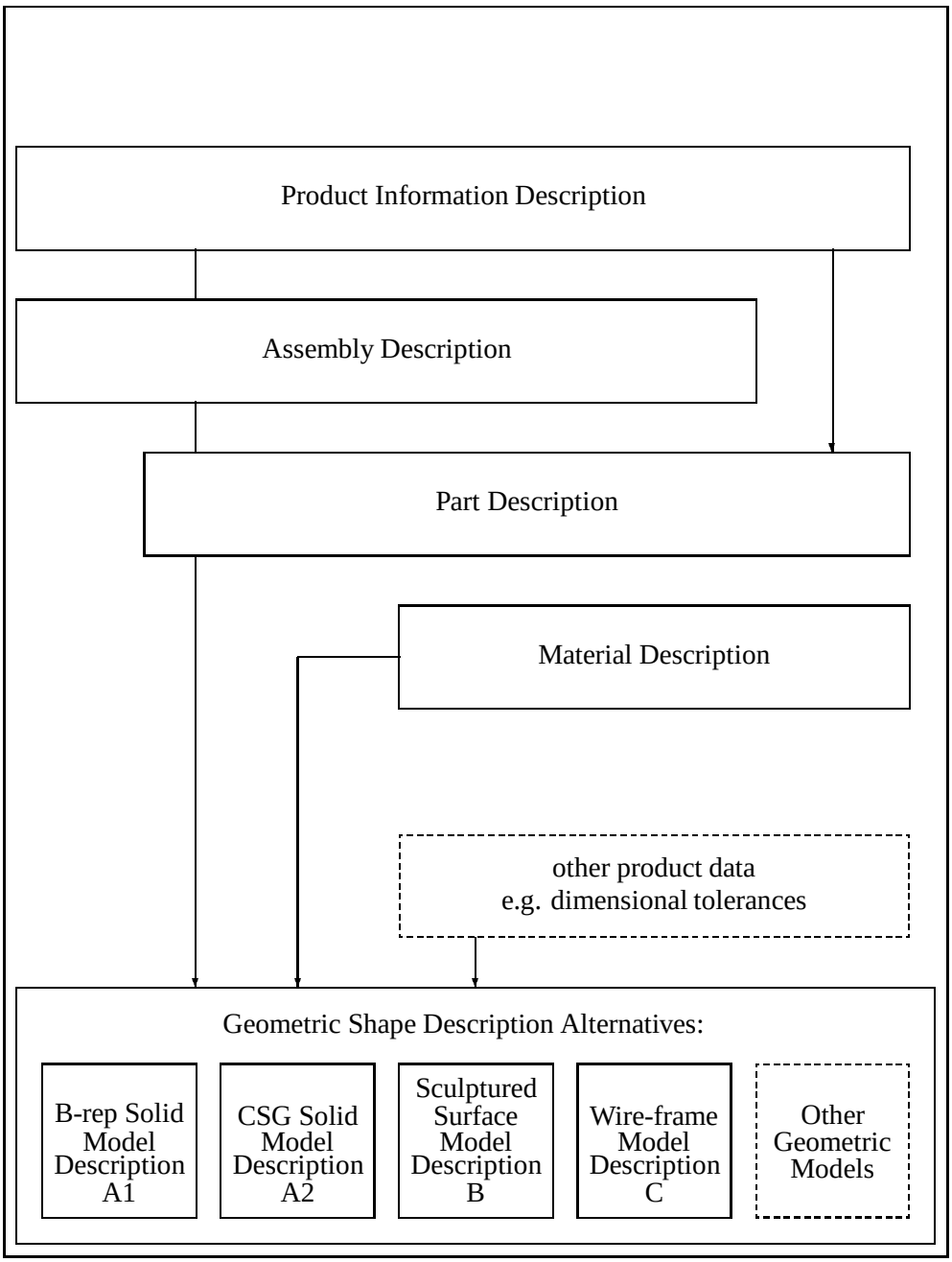


Figure F.1 – Conceptual structure of mechanical design product

EXAMPLE 2 Companies in this industry manufacture:

— computer products (printers, disc drives, plotters), (parts examples: plastic housings, switches);

— electromechanical equipment for the automotive industry, (electric motors, electricity generator, starter motor, batteries);

— consumer products (e.g. telephones, hand-held drilling machines, hair dryers).

Transportation Industry

This industry manufactures and assembles parts used for transportation.

EXAMPLE 3 Companies which manufacture:

— automobile engines;

— mechanical power transmission equipment, gears, transaxles;

— wheels, tyres;

— hydraulic equipment e.g., valves;

F.4 AAM diagrams

The application activity model diagrams are given in figures F.2 to F.5. The graphical form of the application activity model is presented in the IDEF0 activity modelling format. Activities and data flows that are out of scope are marked with asterisks.

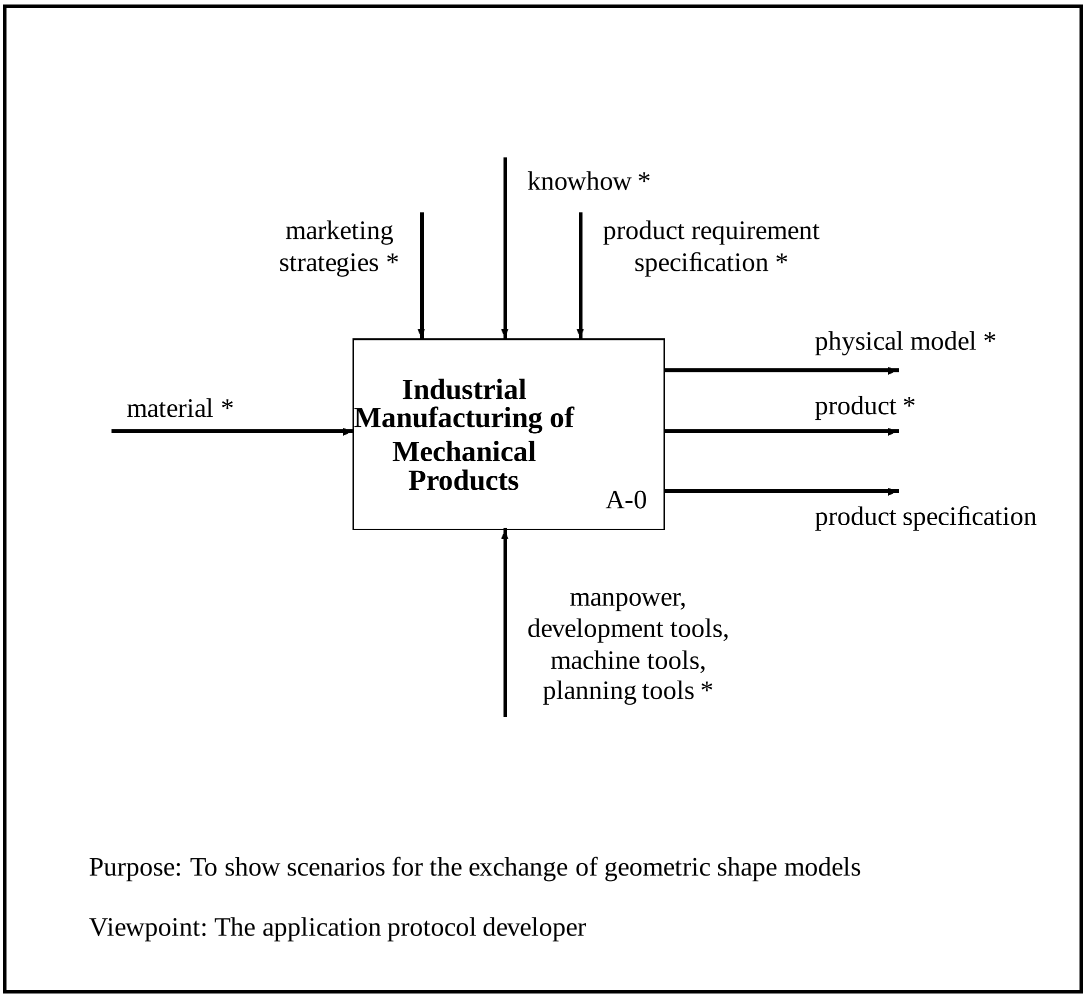


Figure F.2 – Industrial manufacturing of mechanical products (node A0)

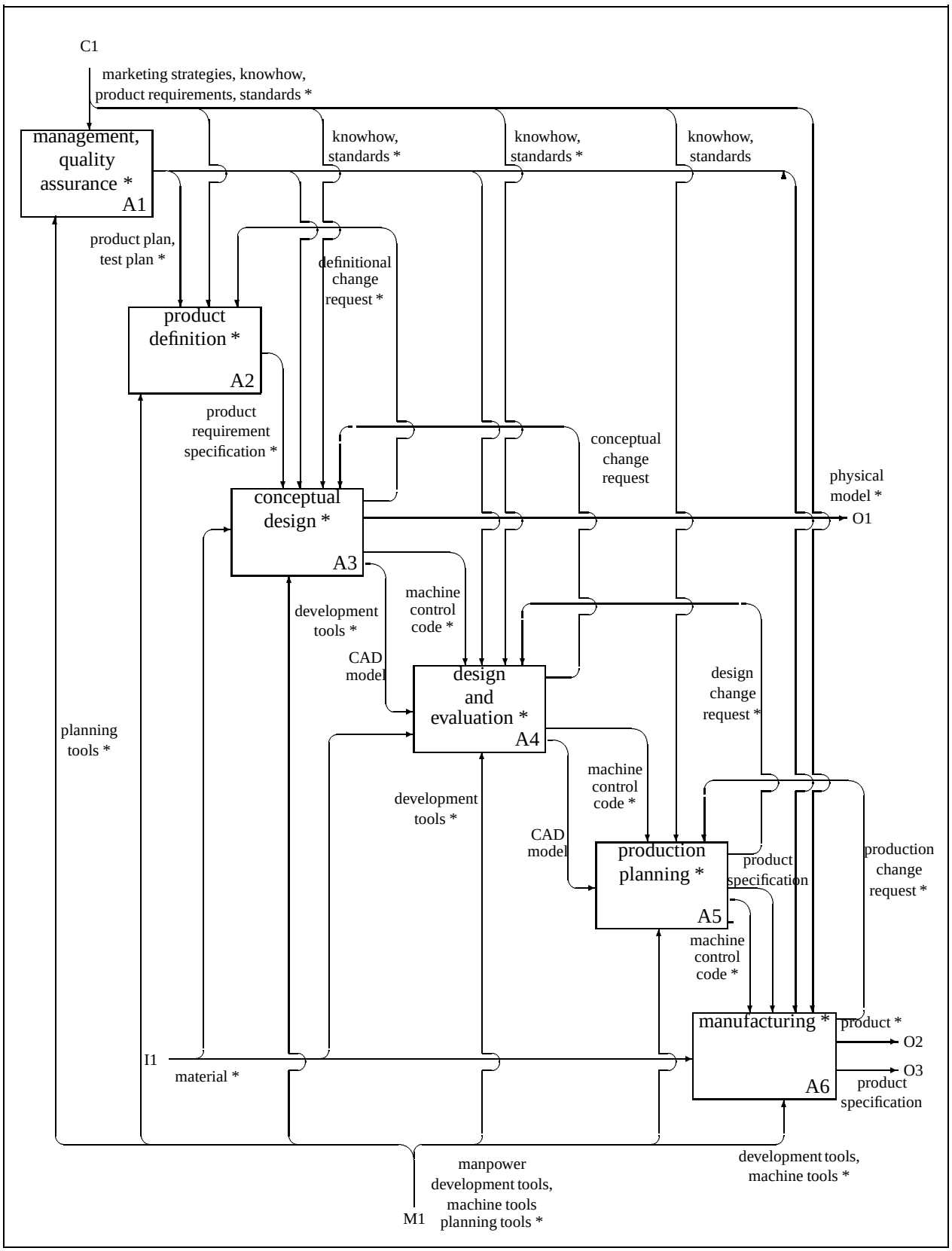


Figure F.3 – Industrial manufacturing of mechanical products (node A0 expanded)

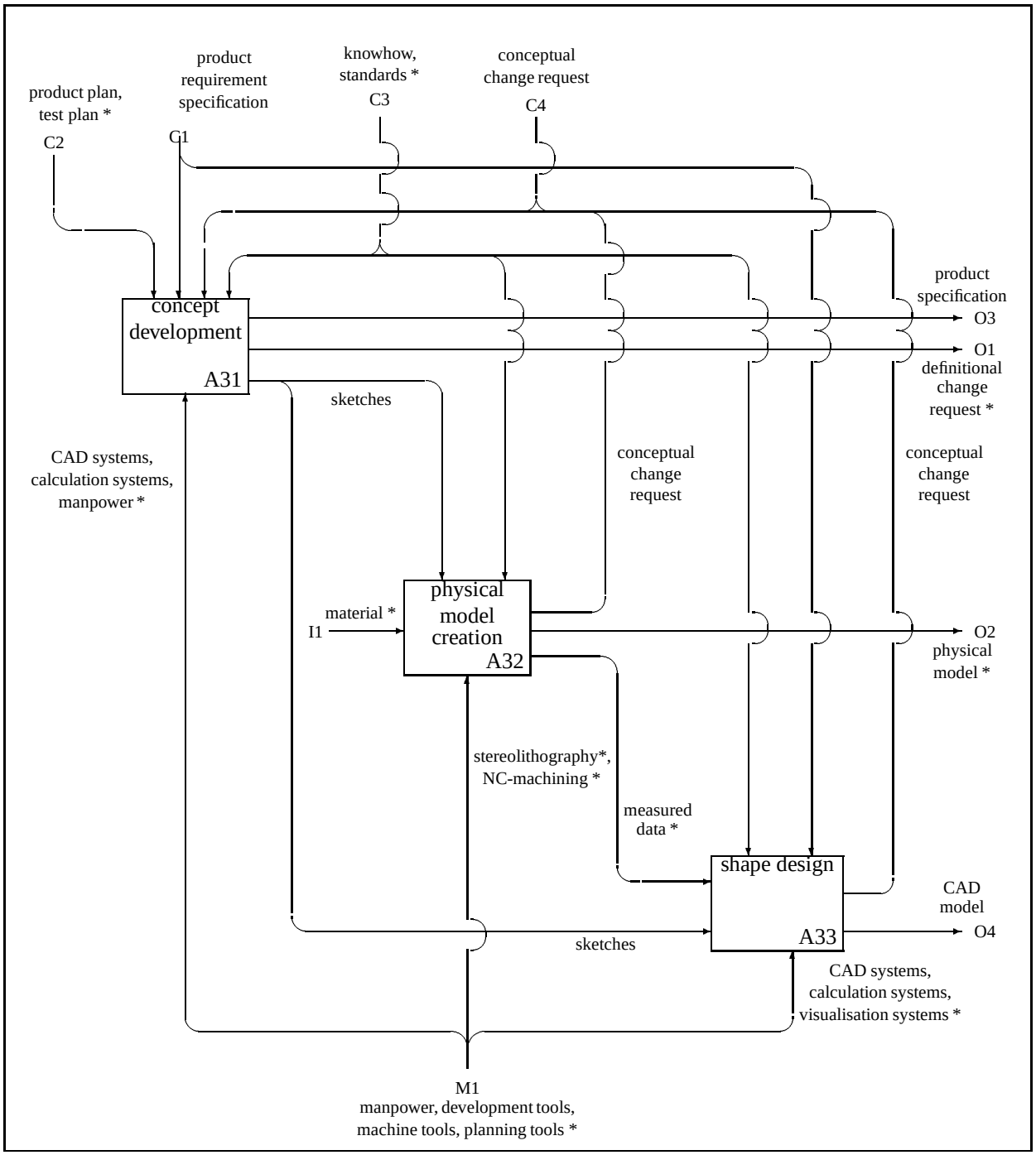


Figure F.4 – Conceptual design (node A3)

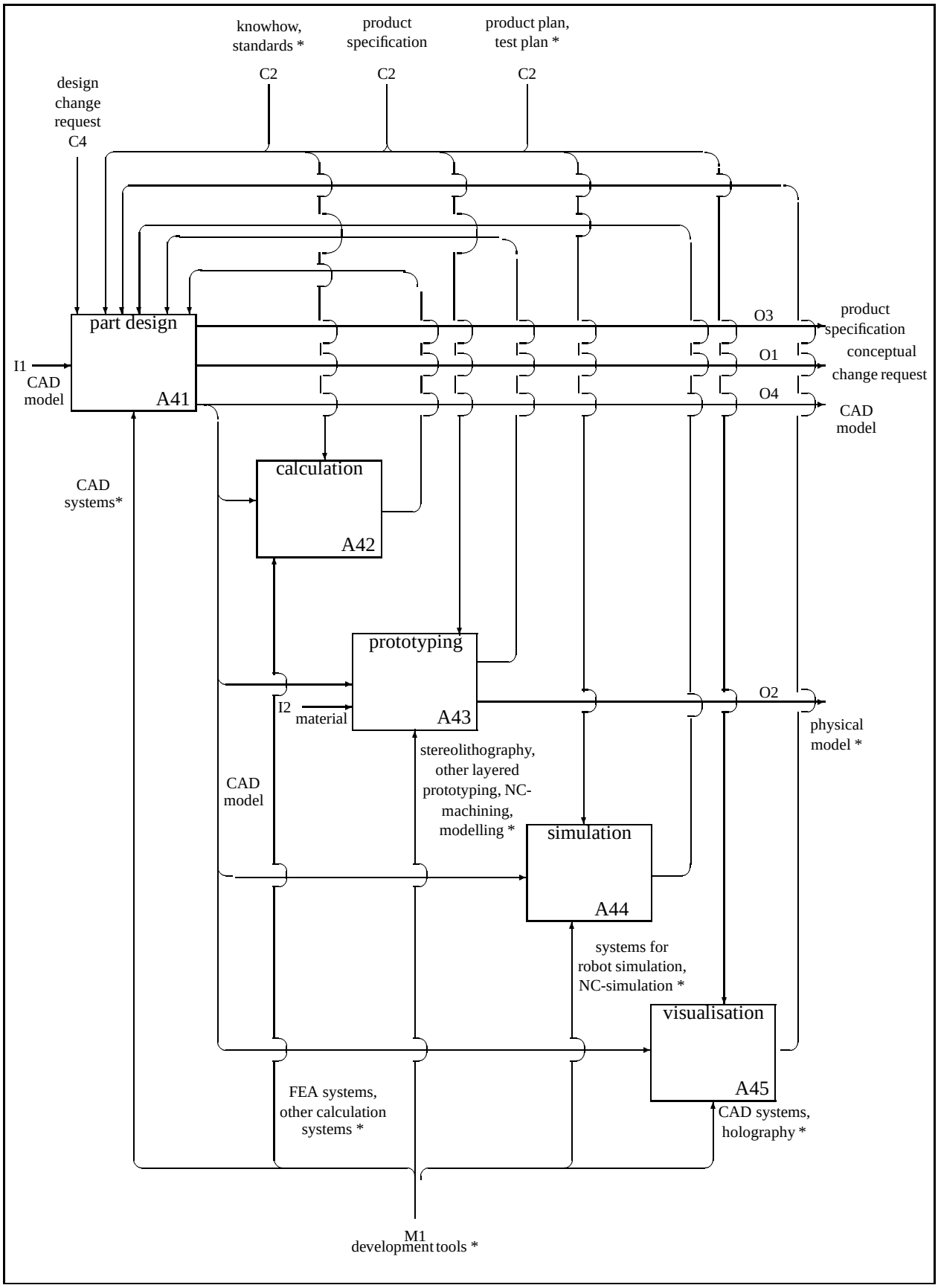


Figure F.5 – Design and evaluation (Node A4)

Annex G
(informative)

Application reference model diagrams

This annex provides the application reference model for this part of ISO 10303 and is given in G.1 through G.10. 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 the NIAM format. The application reference model is independent from any implementation method.

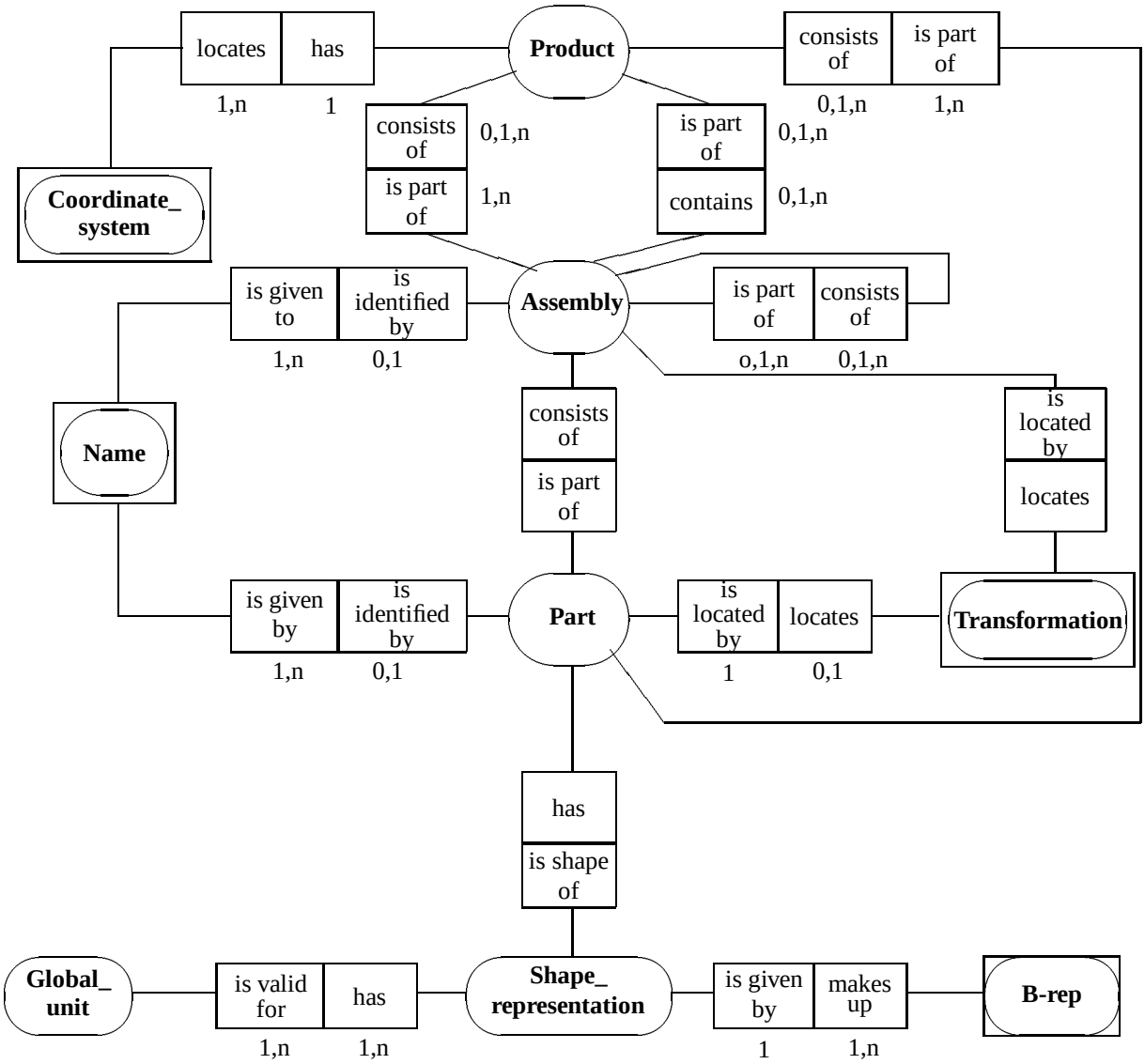


Figure G.1 – ARM diagram (1 of 12)

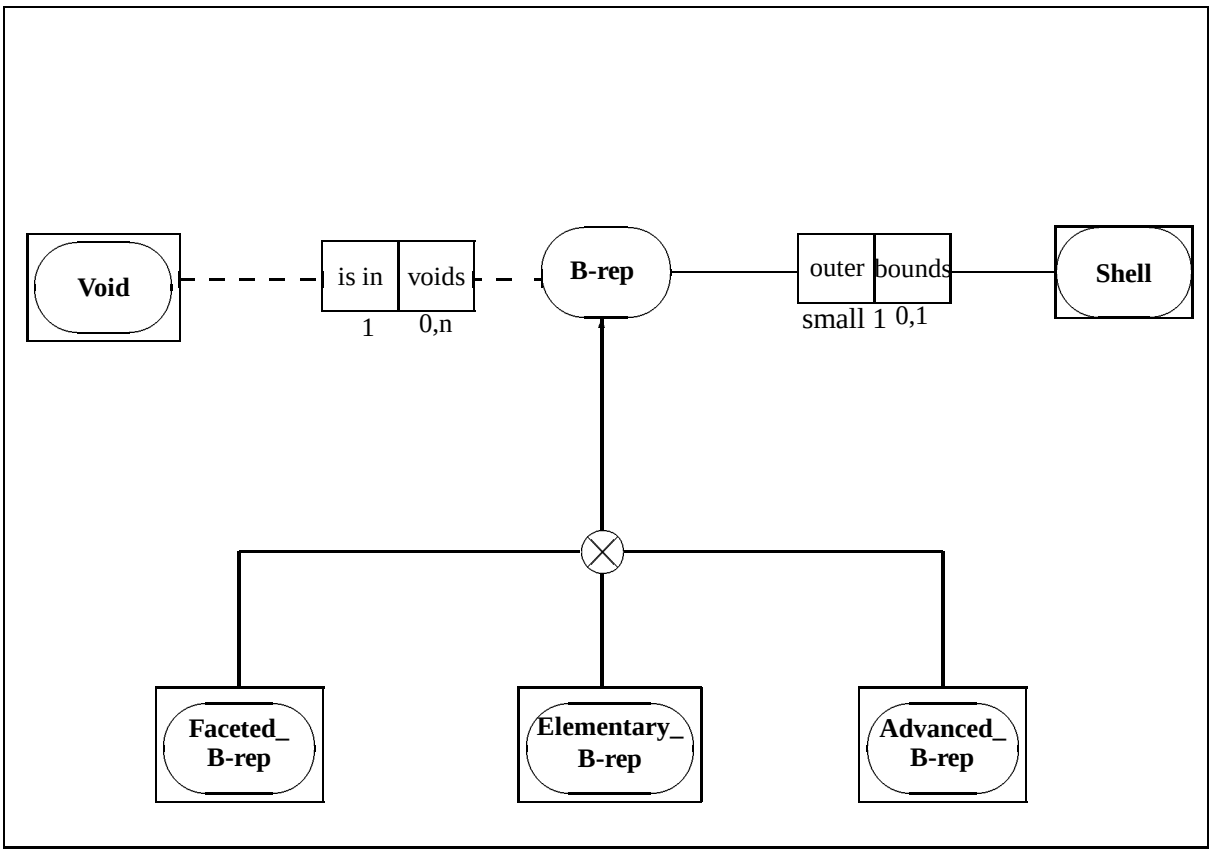


Figure G.2 – ARM diagram (2 of 12)

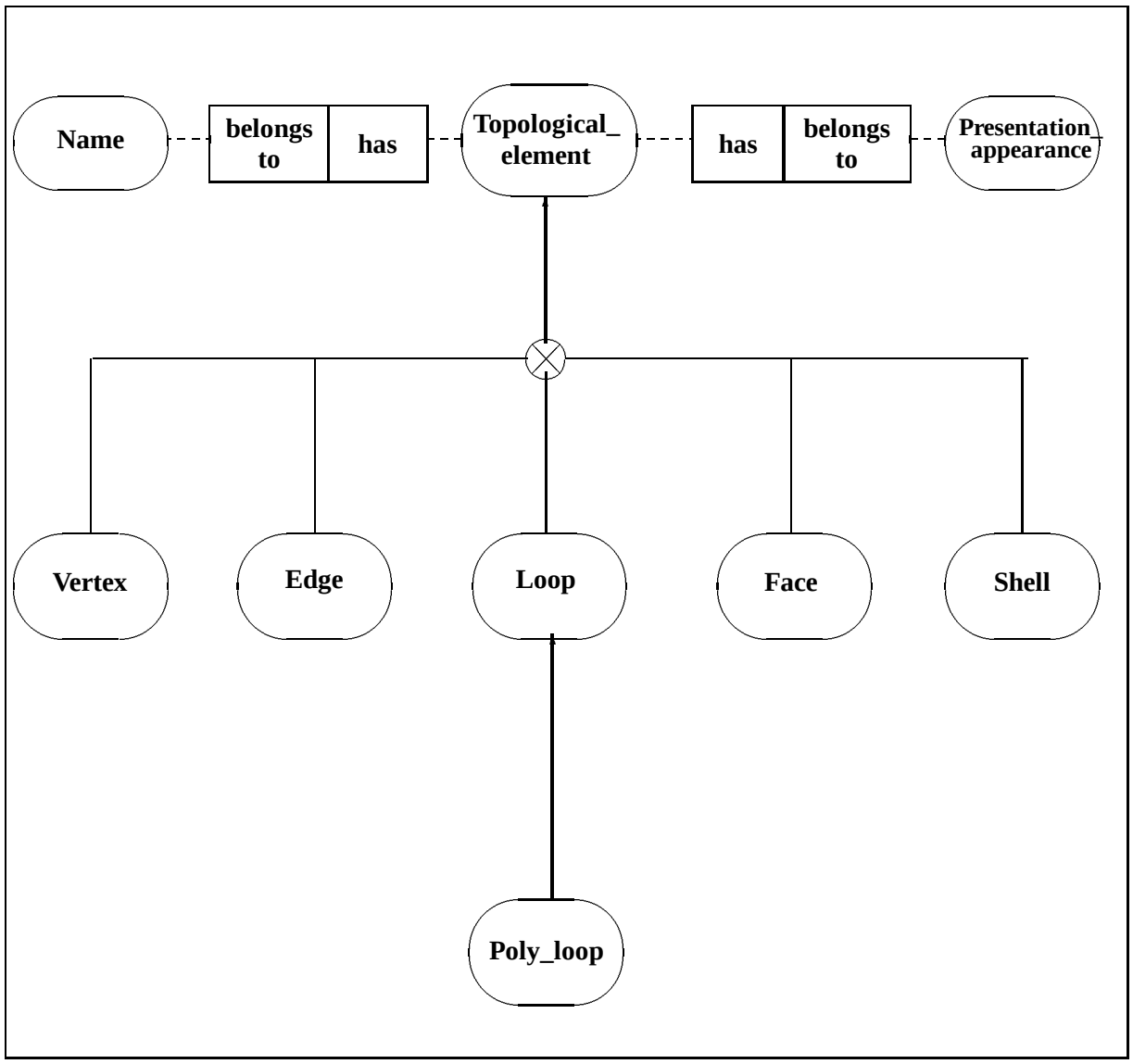


Figure G.3 – ARM diagram (3 of 12)

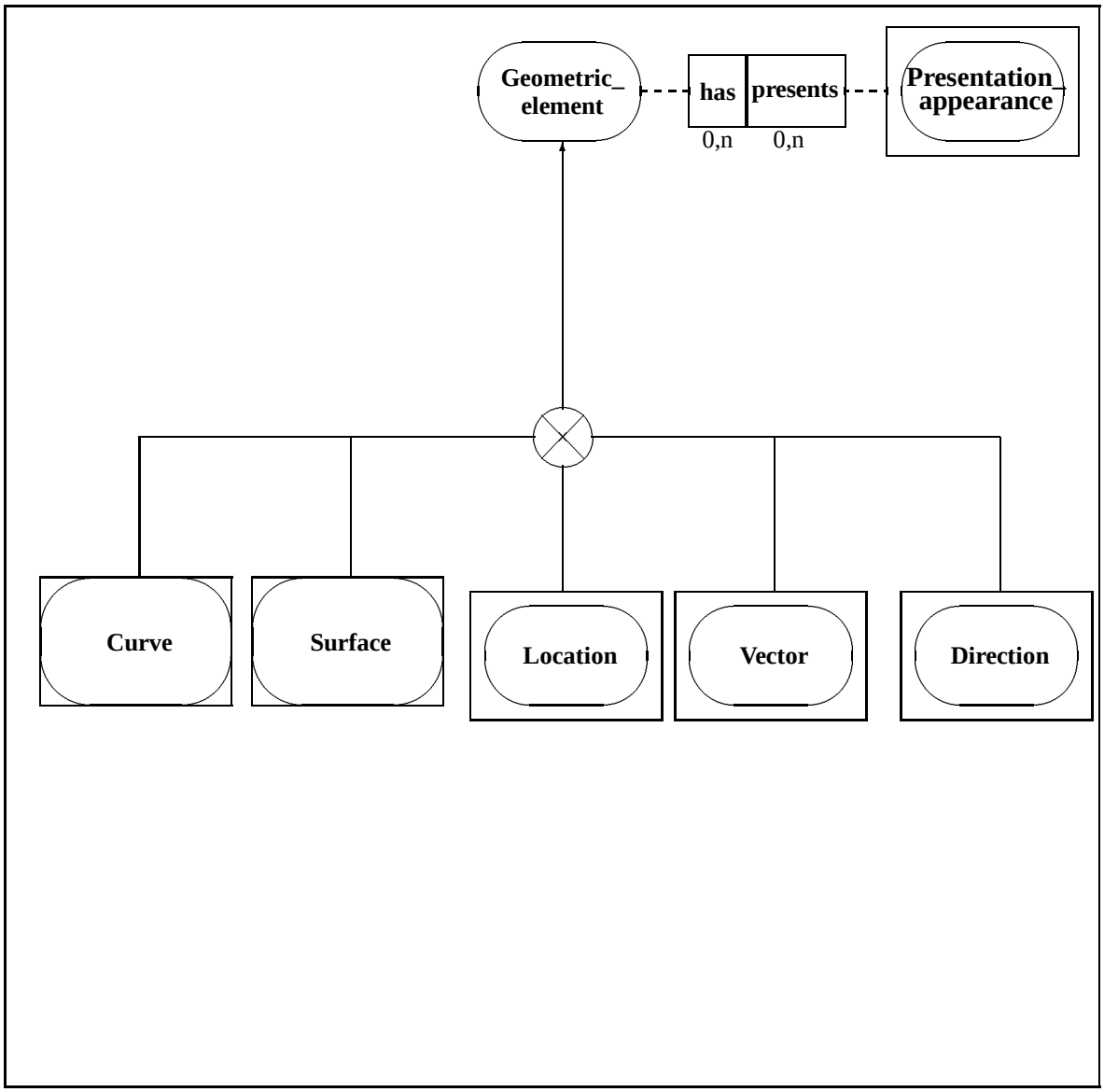


Figure G.4 – ARM diagram (4 of 12)

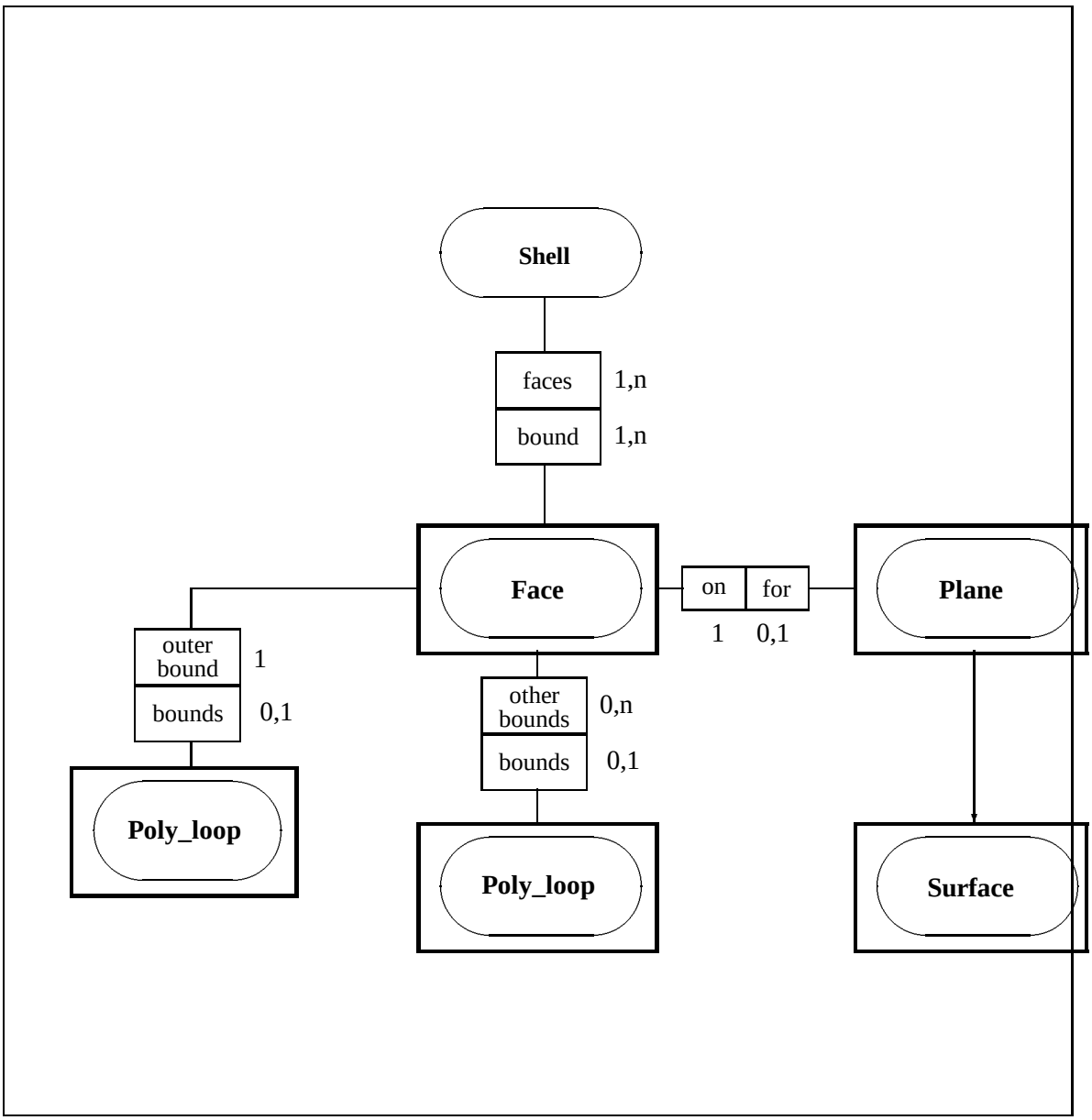


Figure G.5 – ARM diagram (5 of 12) shell in faceted B-rep

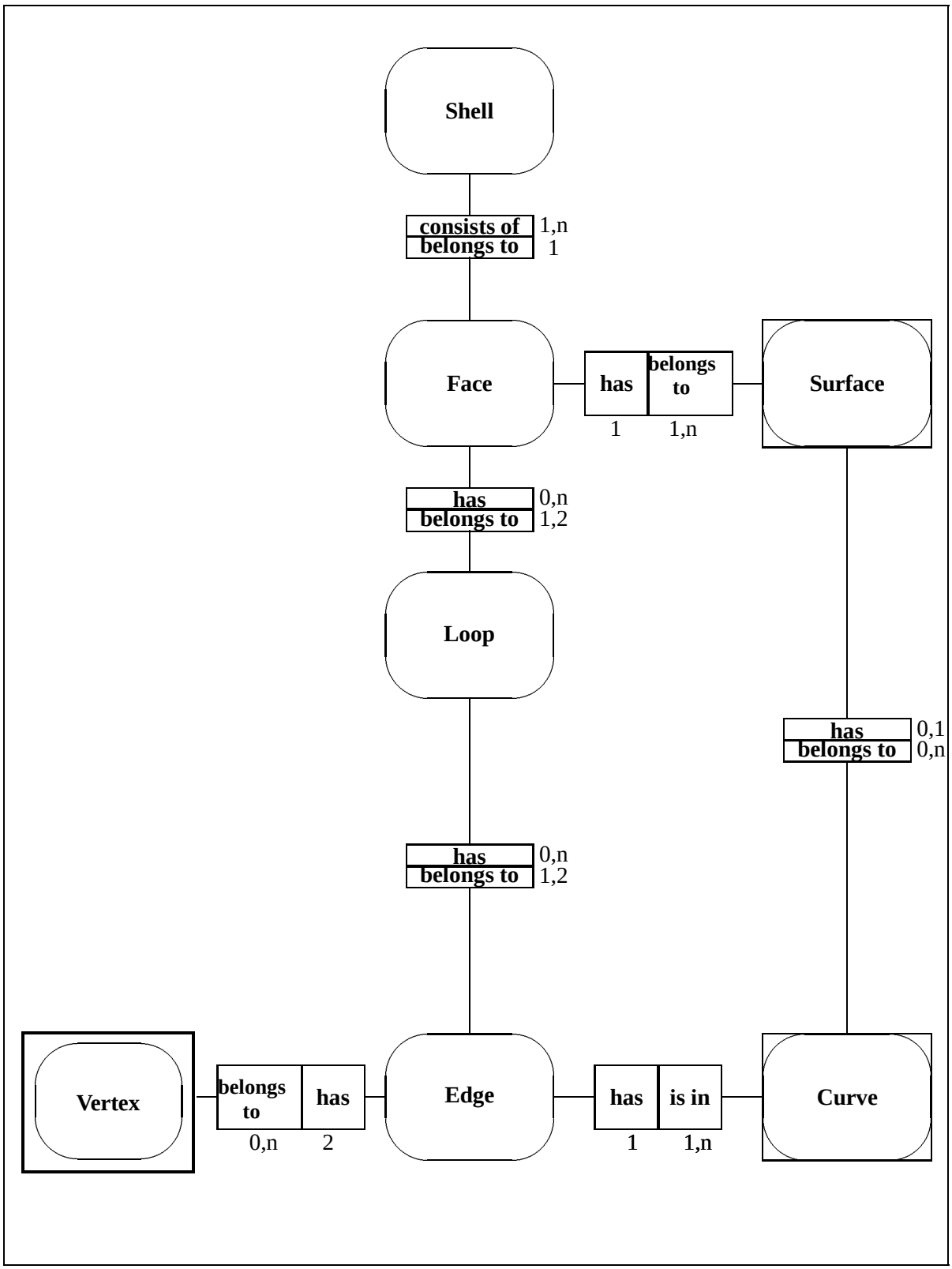


Figure G.6 – ARM diagram (6 of 12) shell in elementary or advanced_B-rep

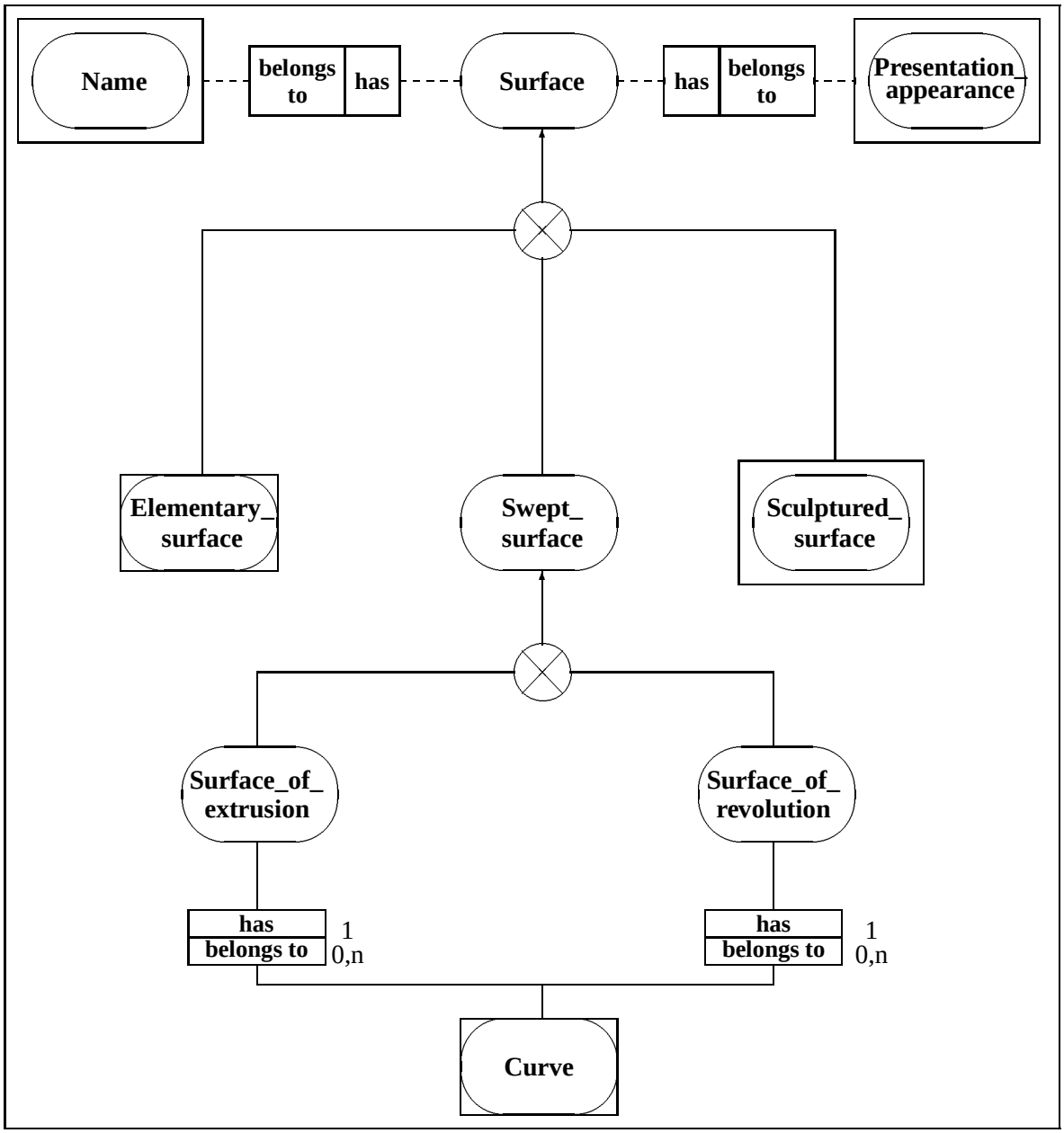


Figure G.7 – ARM diagram (7 of 12) surface in advanced B-rep

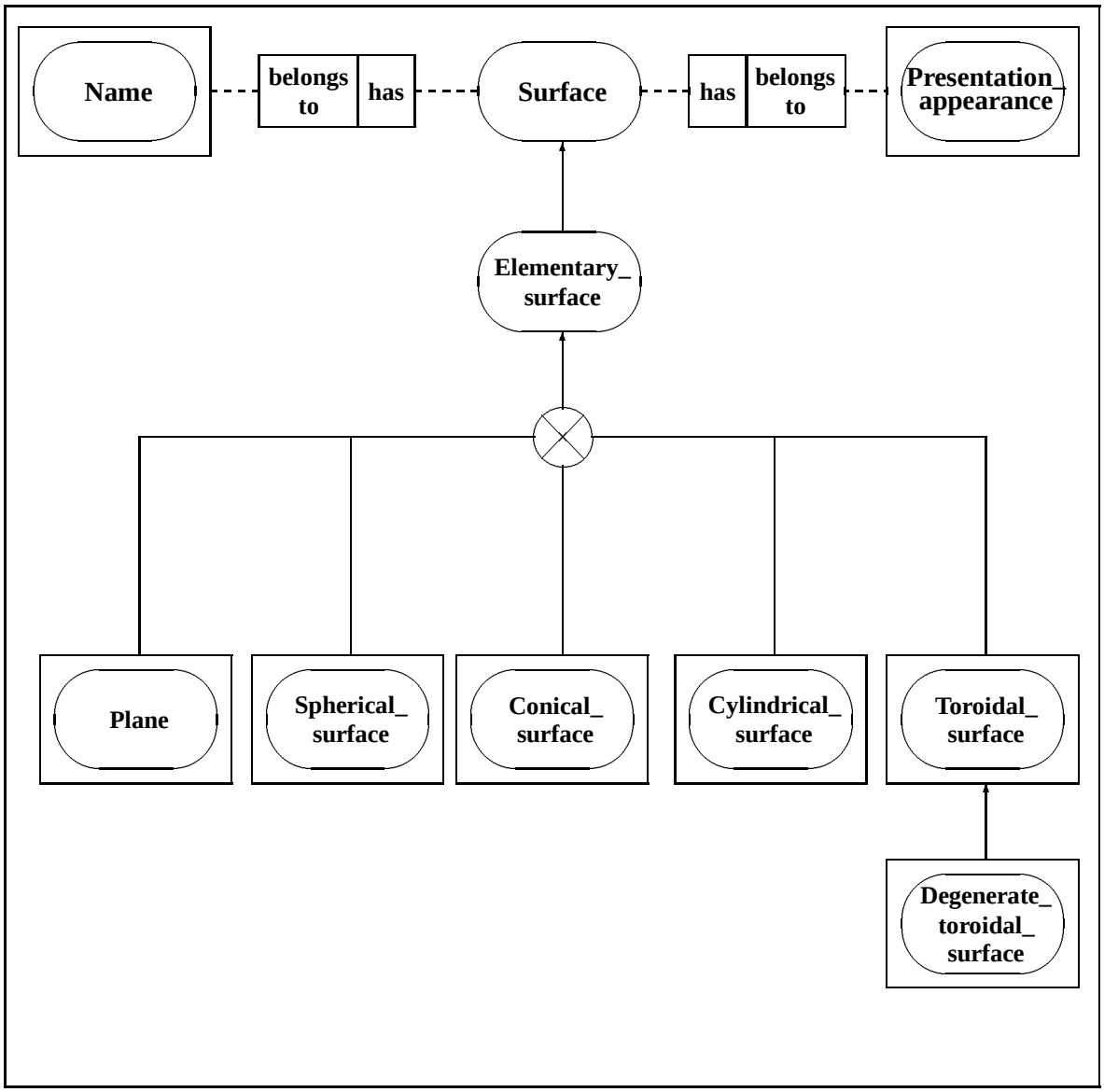


Figure G.8 – ARM diagram (8 of 12) surface in elementary B-rep

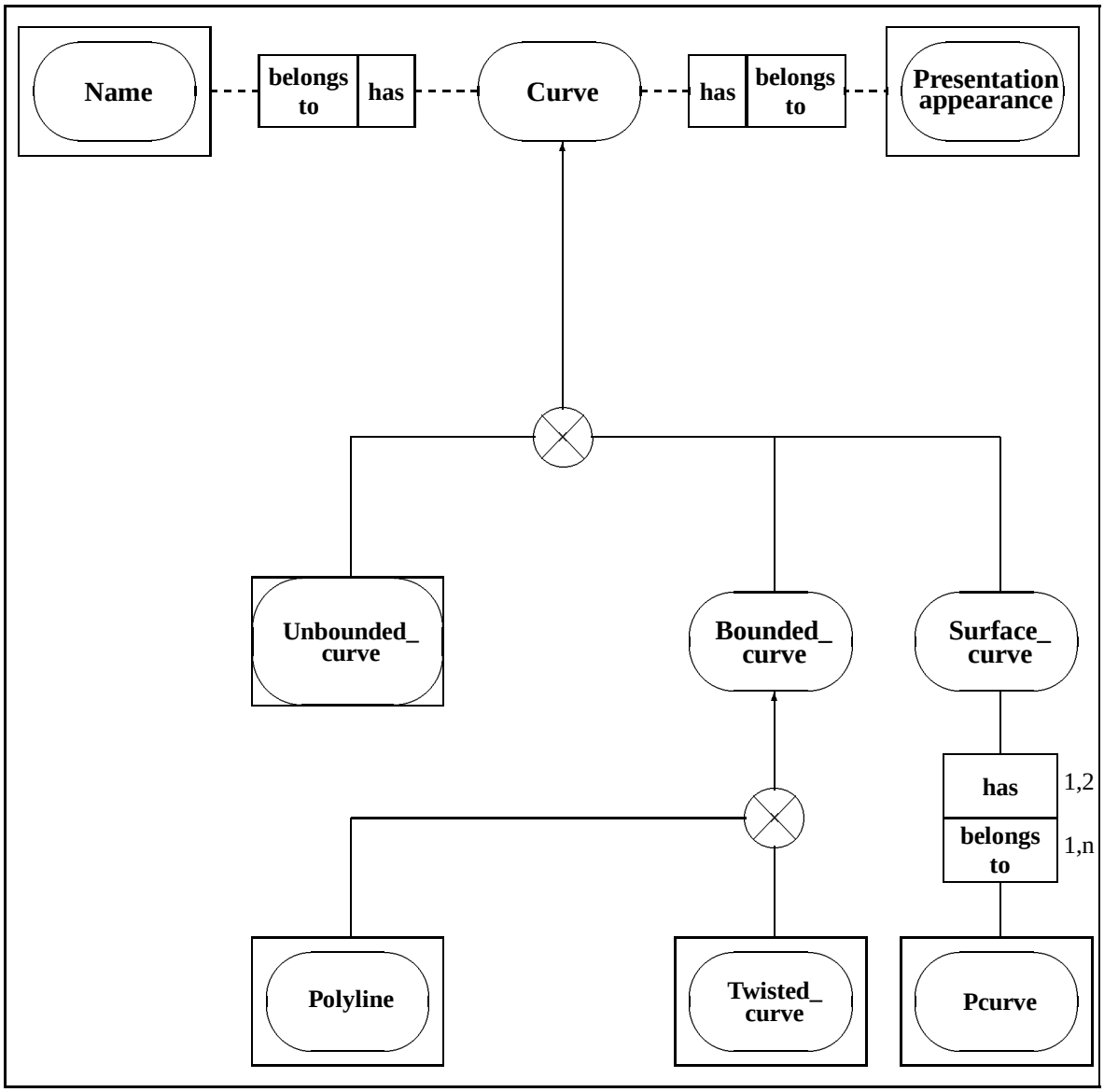


Figure G.9 – ARM diagram (9 of 12) curve in advanced_B-rep

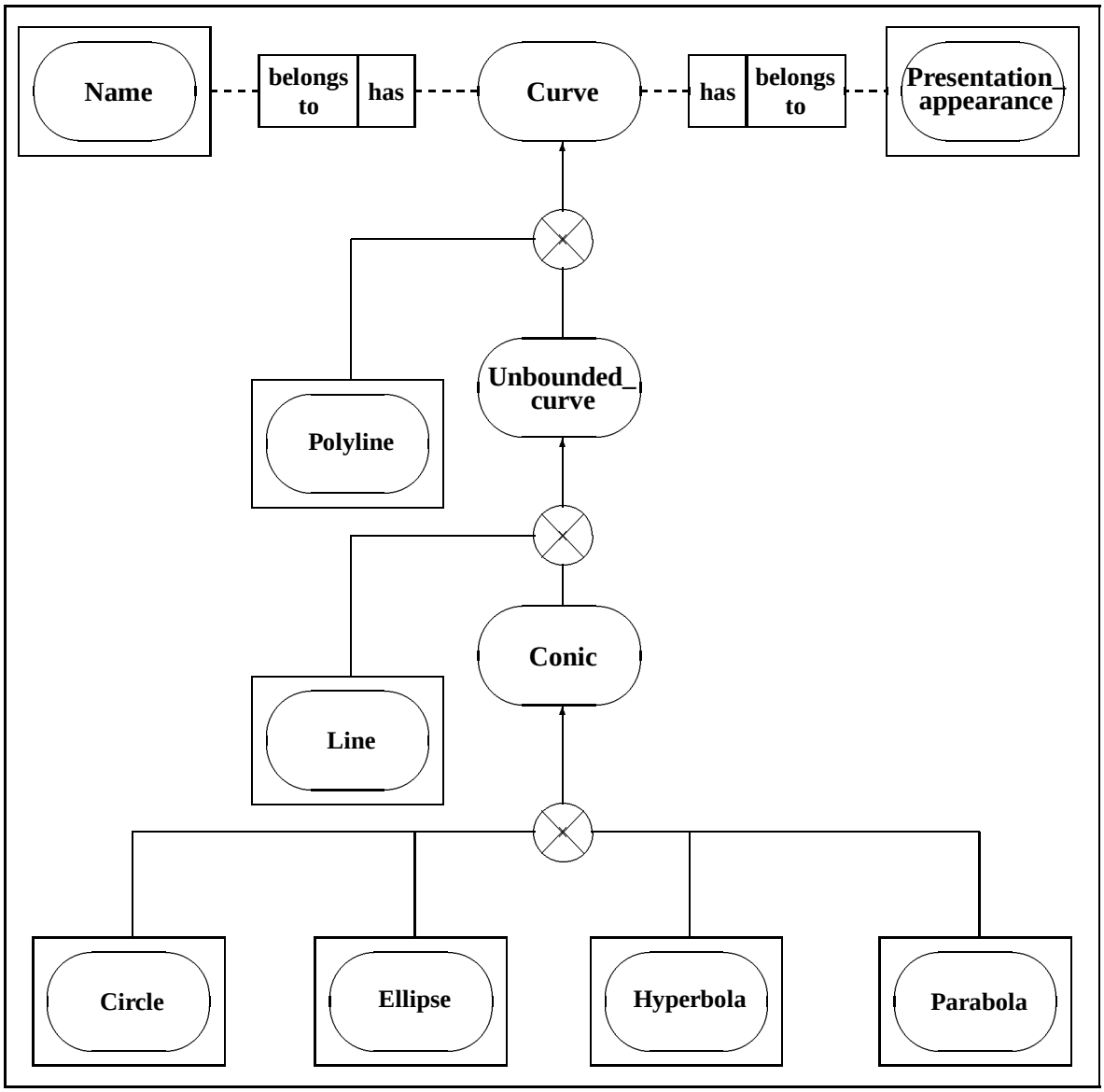


Figure G.10 – ARM diagram (10 of 12) curve in elementary_B-rep

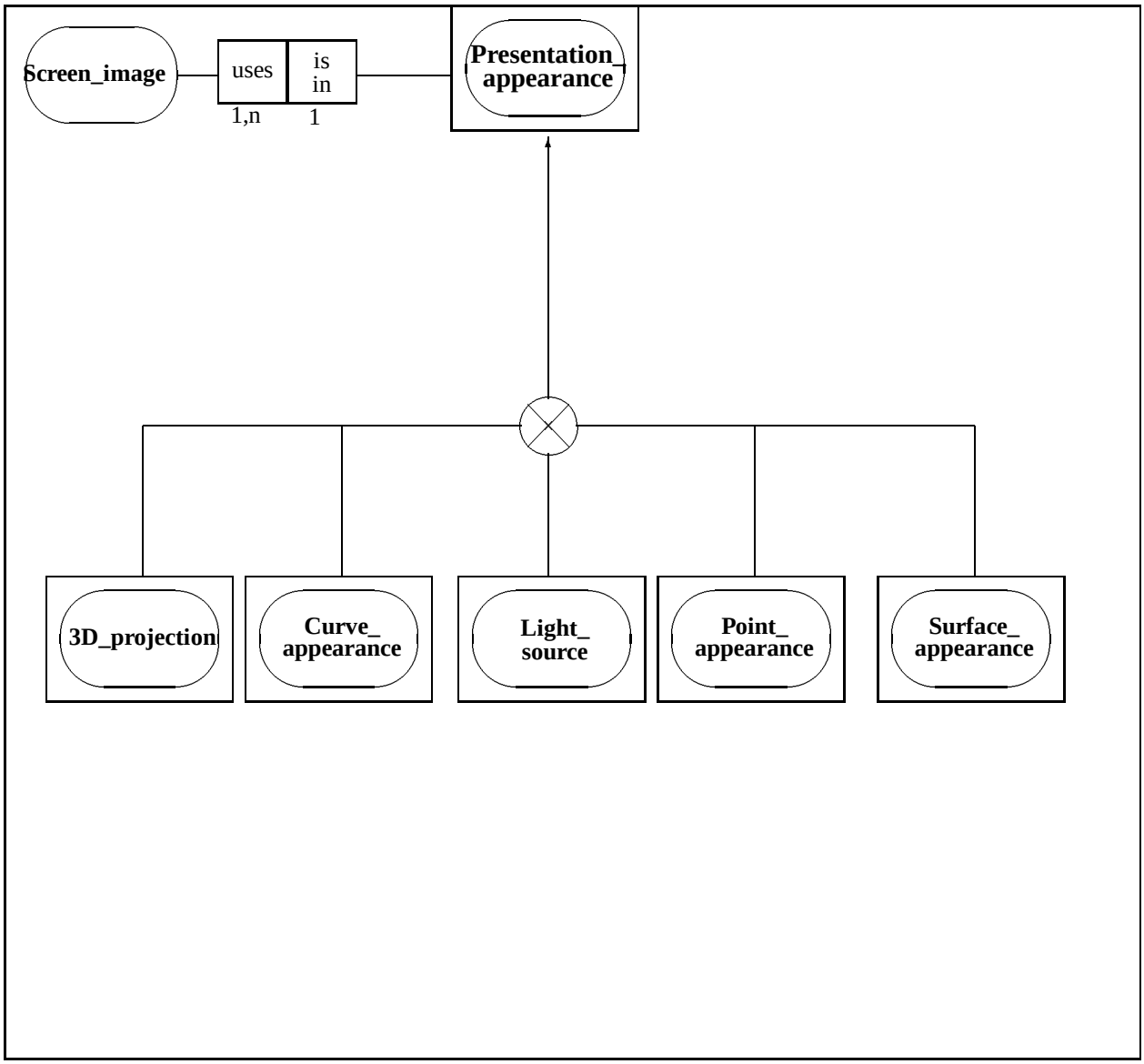


Figure G.11 – ARM diagram (11 of 12)

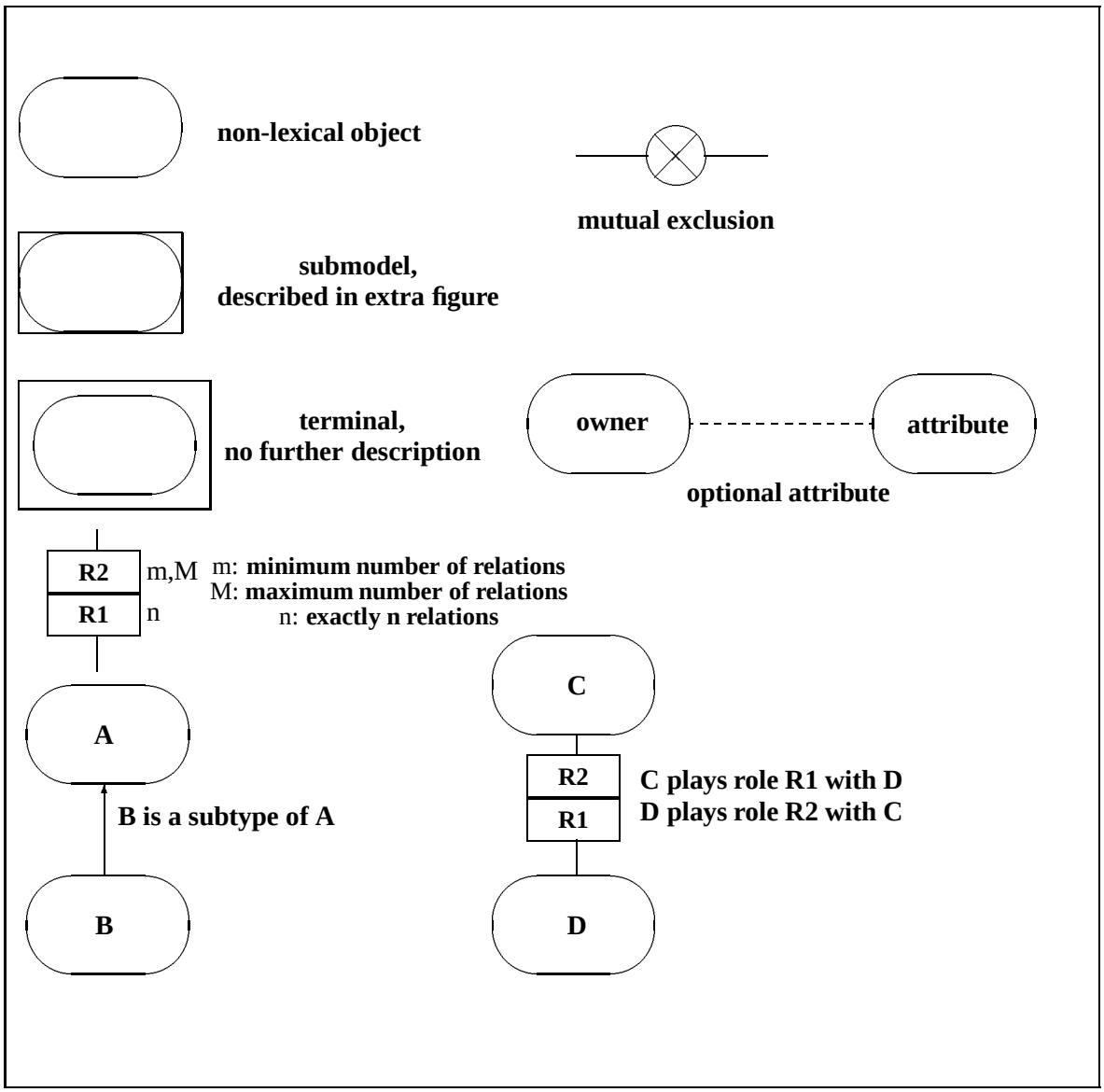


Figure G.12 – ARM diagram (12 of 12) conventions used in NIAM diagrams

Annex H
(informative)

AIM EXPRESS-G

Figures H.1 through H.17 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.

NOTE The rules in the schema exclude the instantiation of some entities which are implicitly interfaced and, therefore, shown in the diagrams. These entities are marked with a " * " in the diagrams.

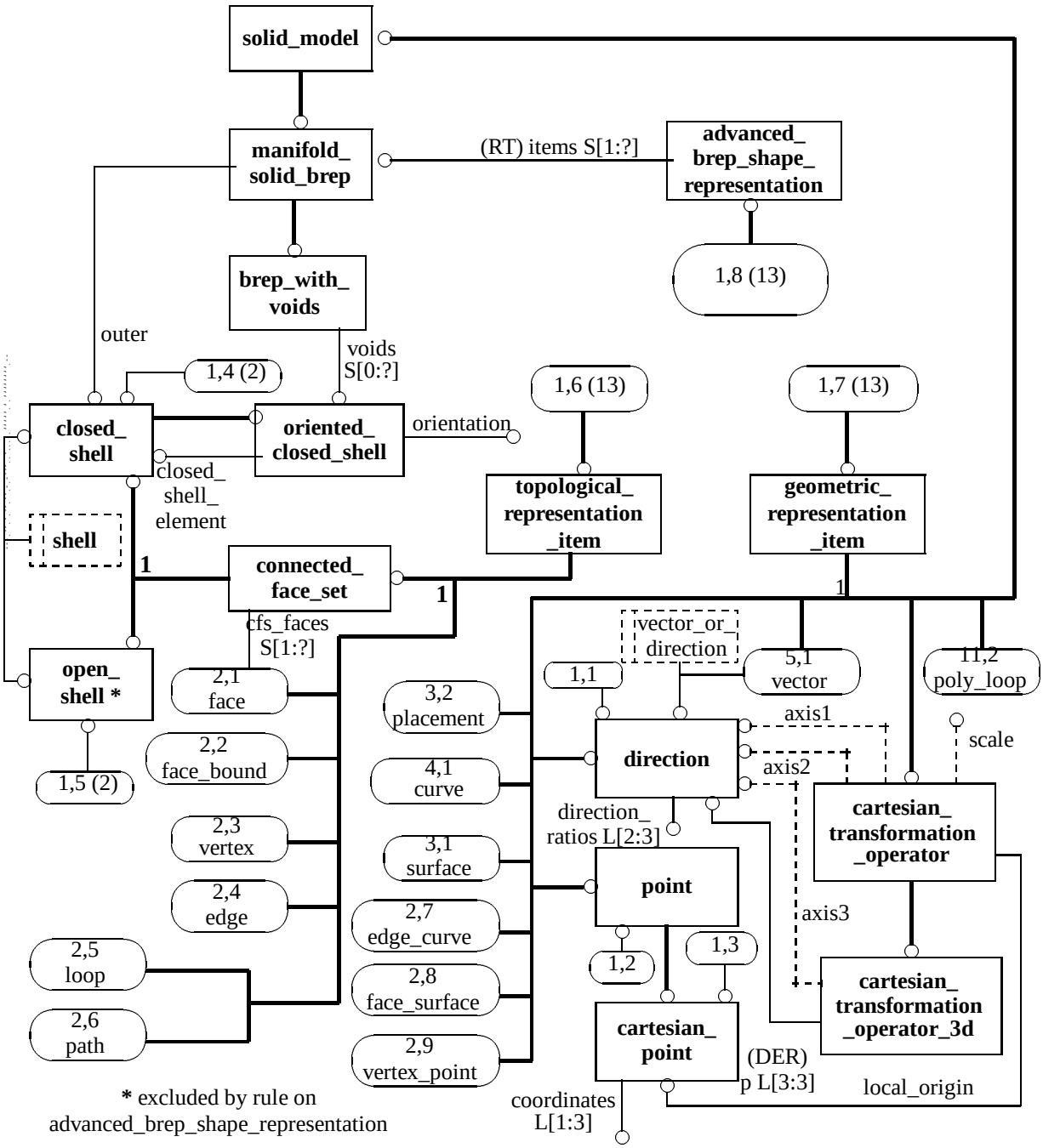


Figure H.1 – AIM EXPRESS-G diagram advanced B-rep

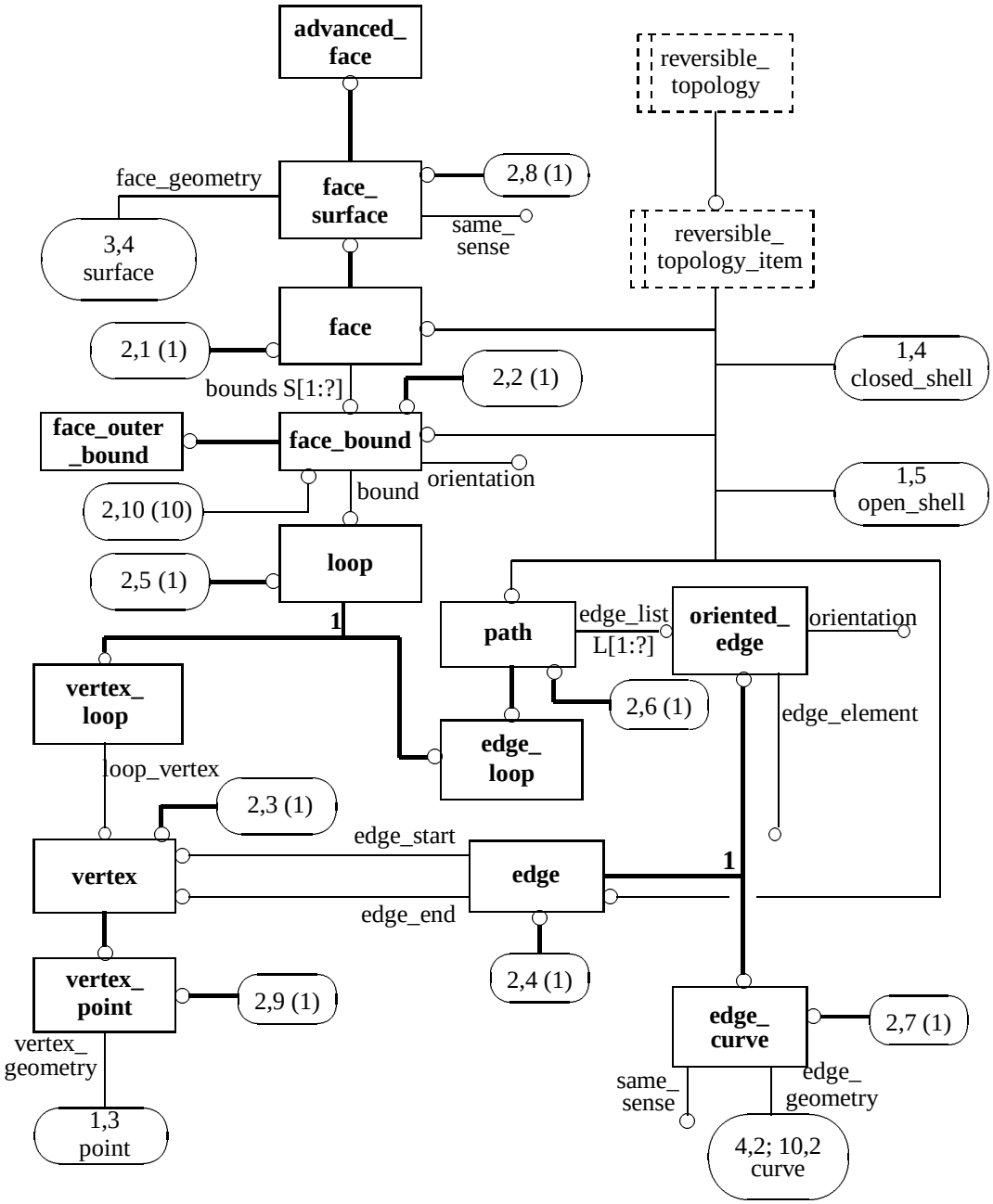


Figure H.2 – AIM EXPRESS-G diagram advanced_face

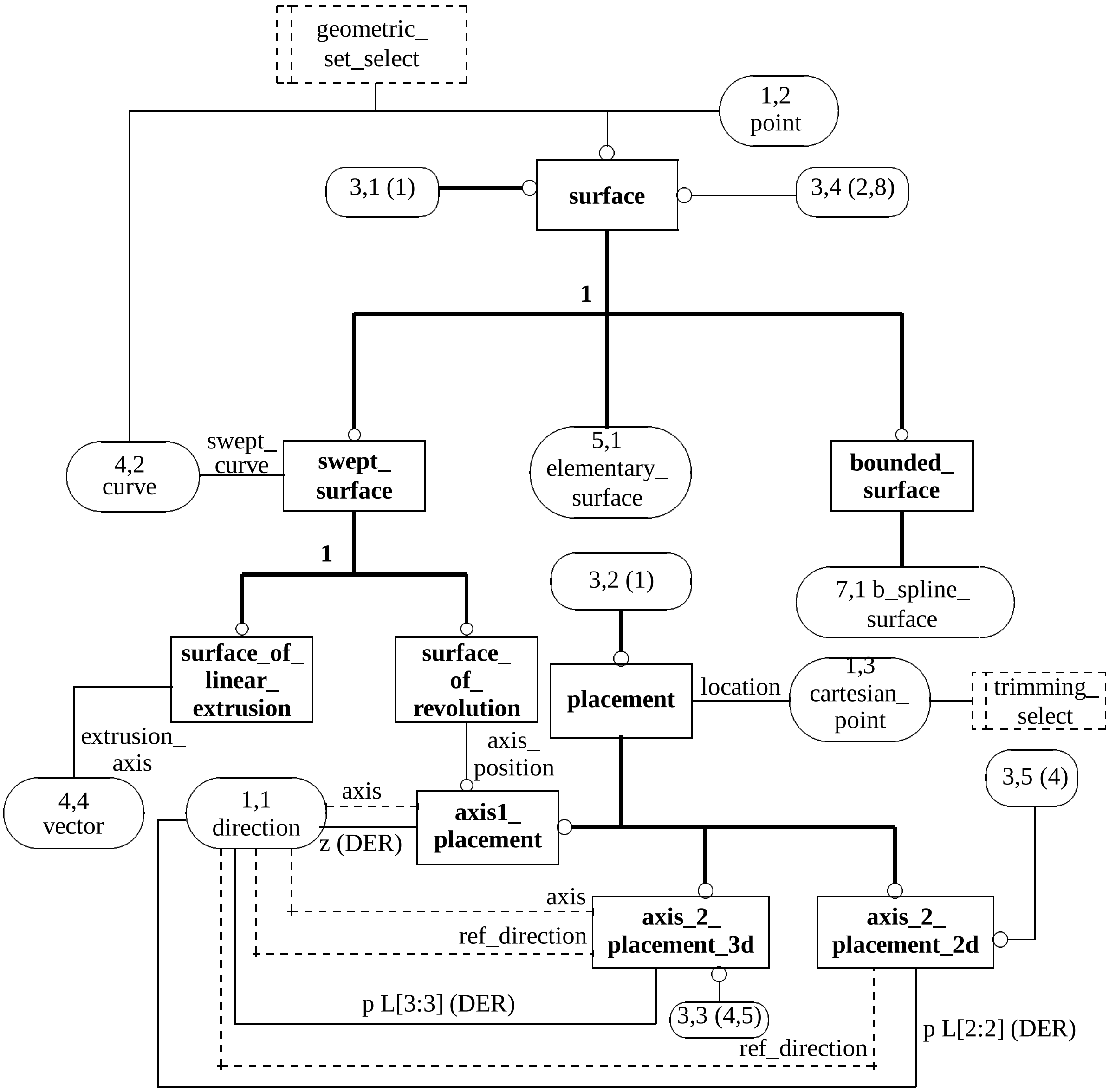


Figure H.3 – AIM EXPRESS-G diagram surfaces

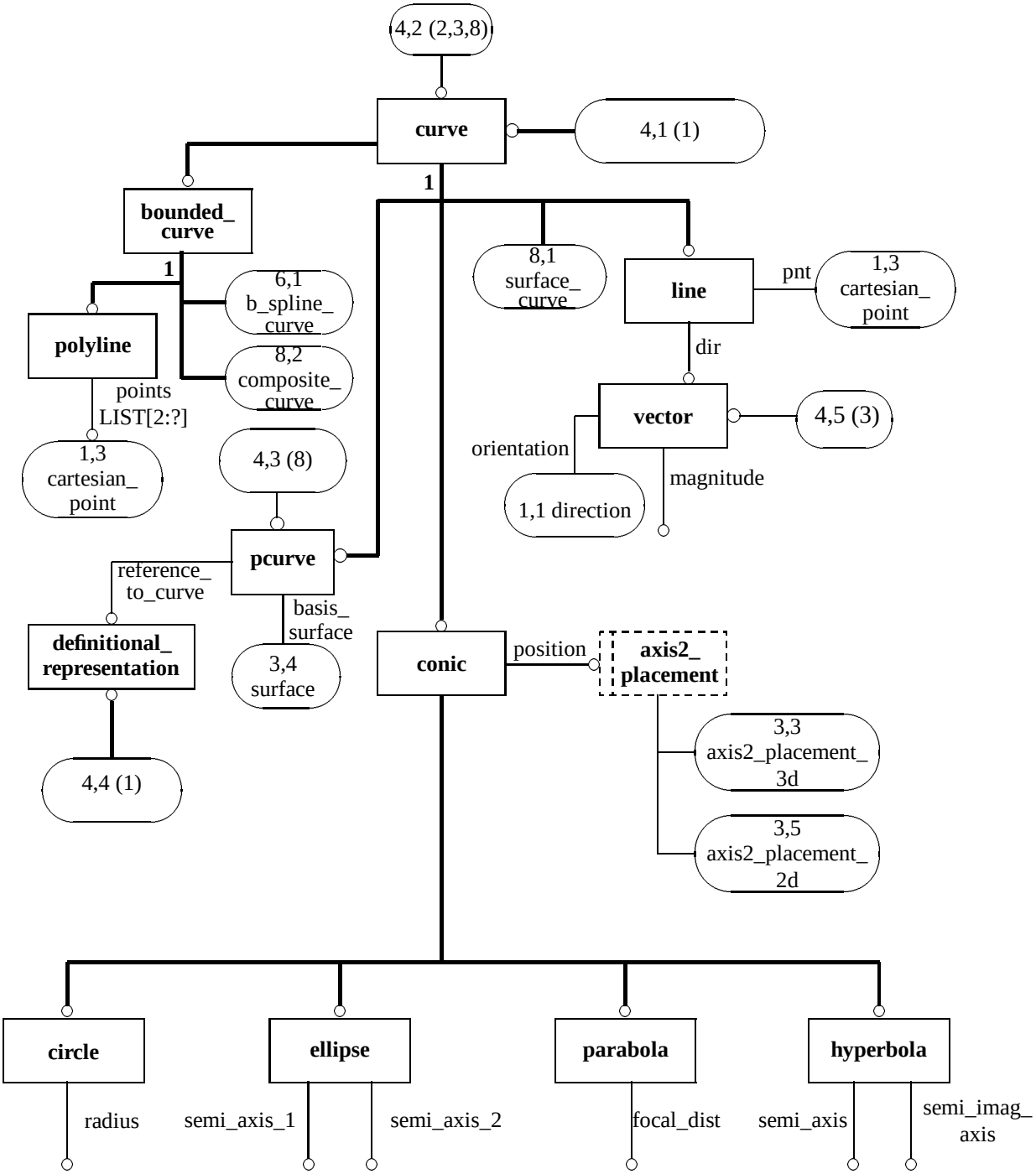


Figure H.4 – AIM EXPRESS-G diagram curves

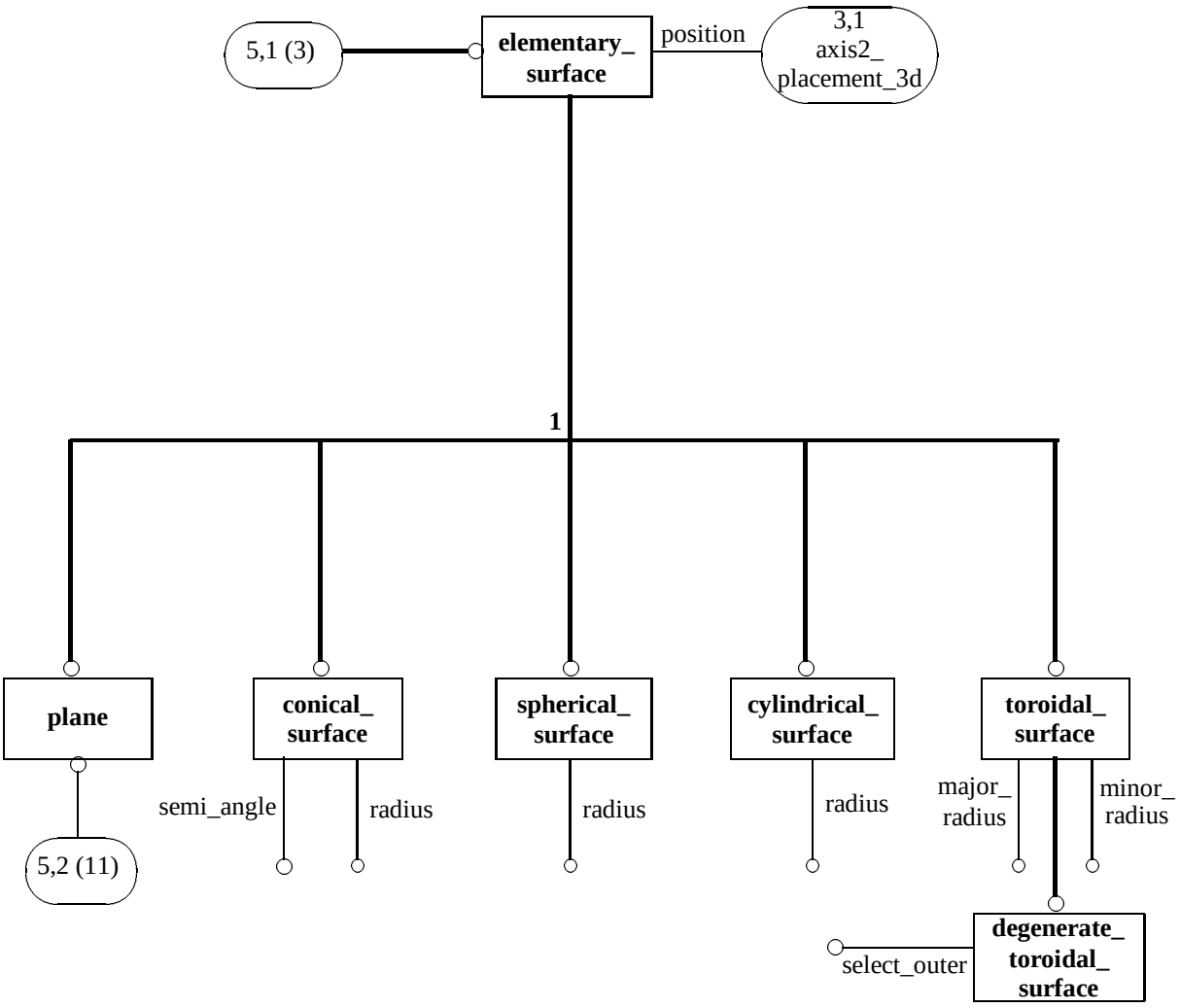


Figure H.5 – AIM EXPRESS-G diagram elementary_surface

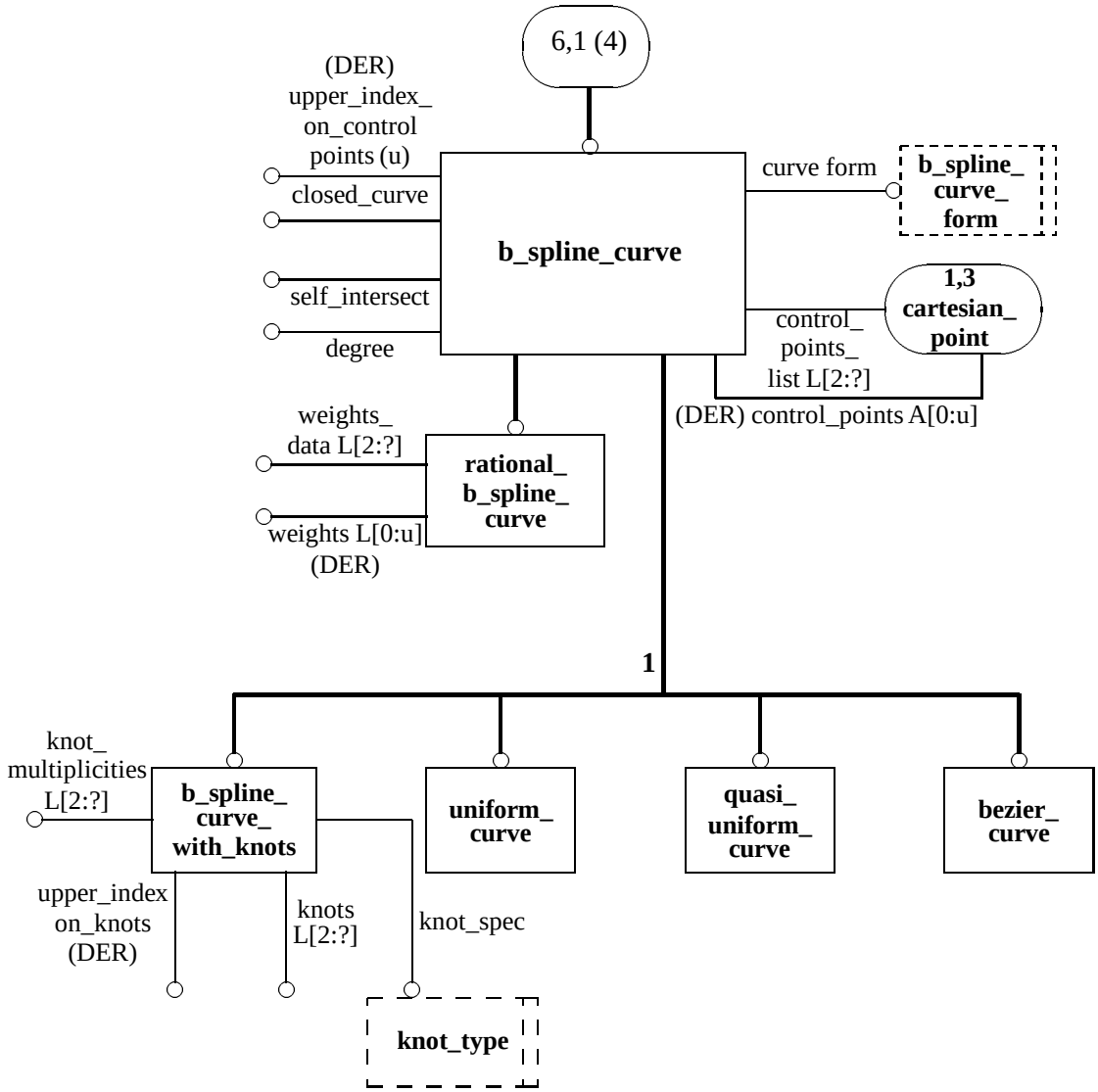


Figure H.6 – AIM EXPRESS-G diagram b_spline_curve

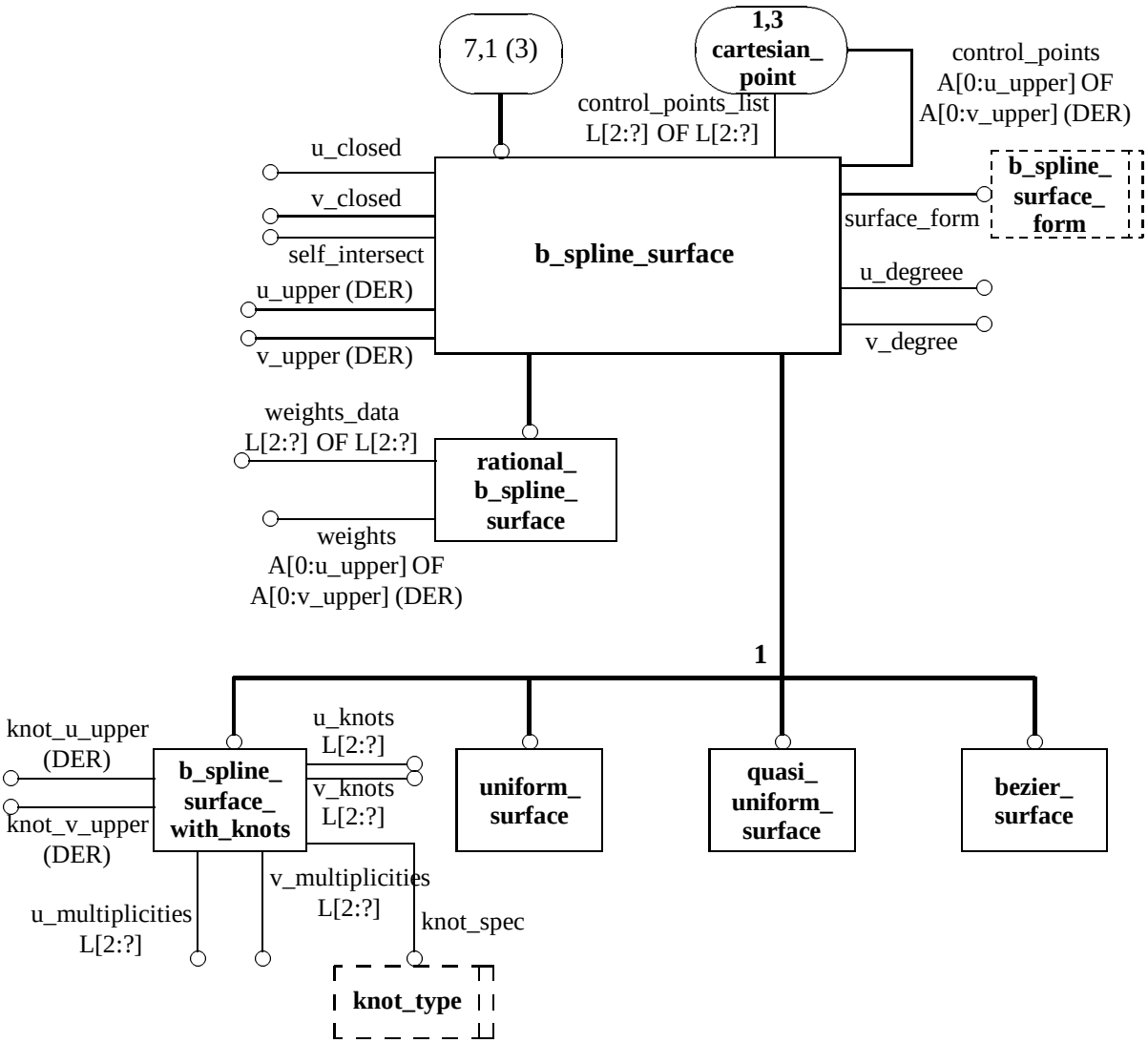
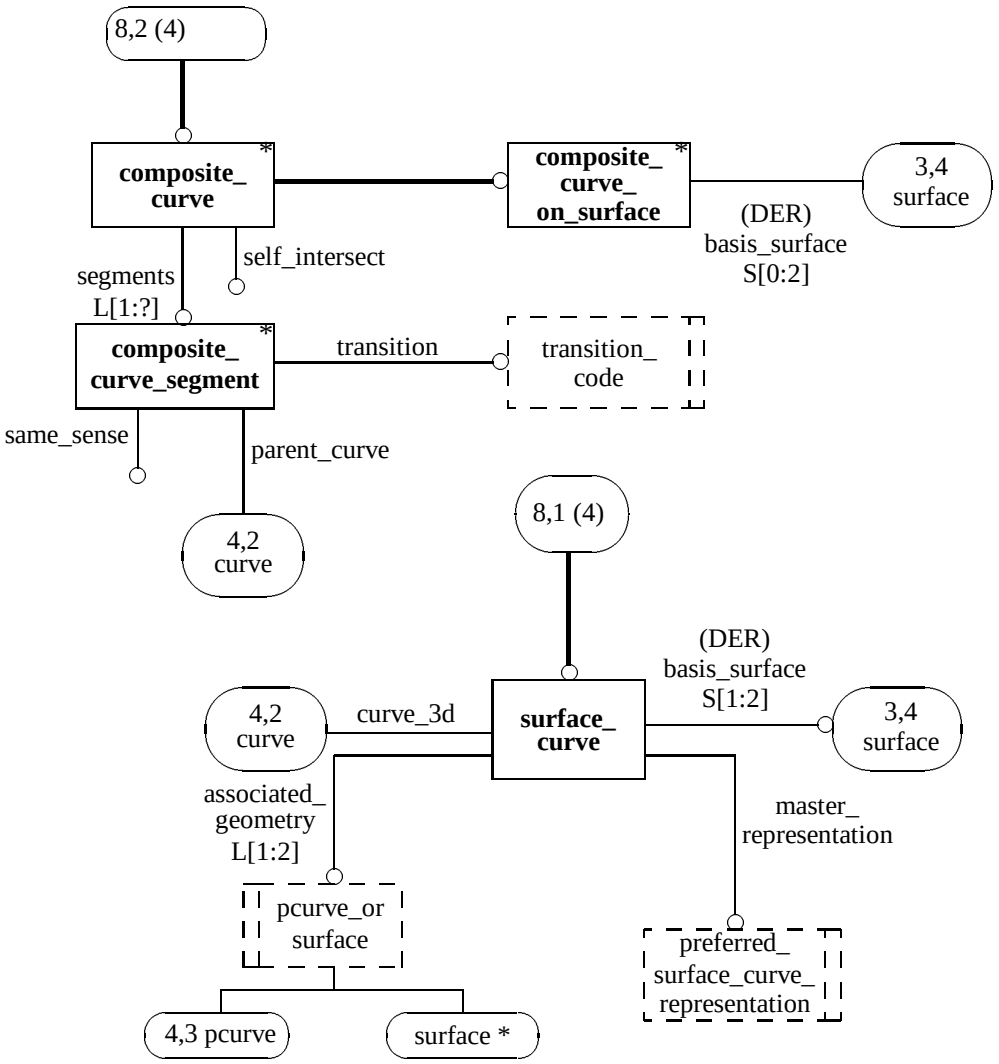


Figure H.7 – AIM EXPRESS-G diagram b_spline_surface



* note: excluded by rule on advanced_face

Figure H.8 – AIM EXPRESS-G diagram surface curves

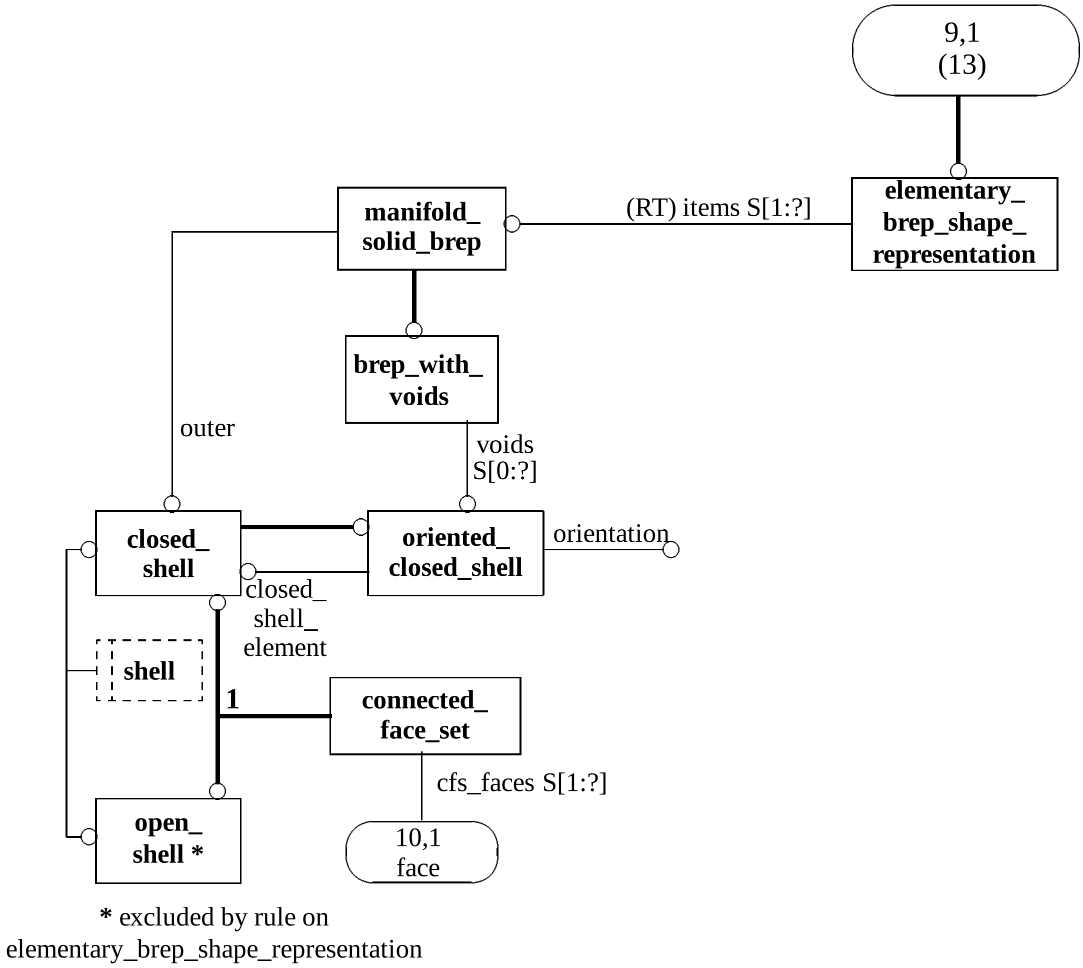


Figure H.9 – AIM EXPRESS-G diagram elementary B-rep

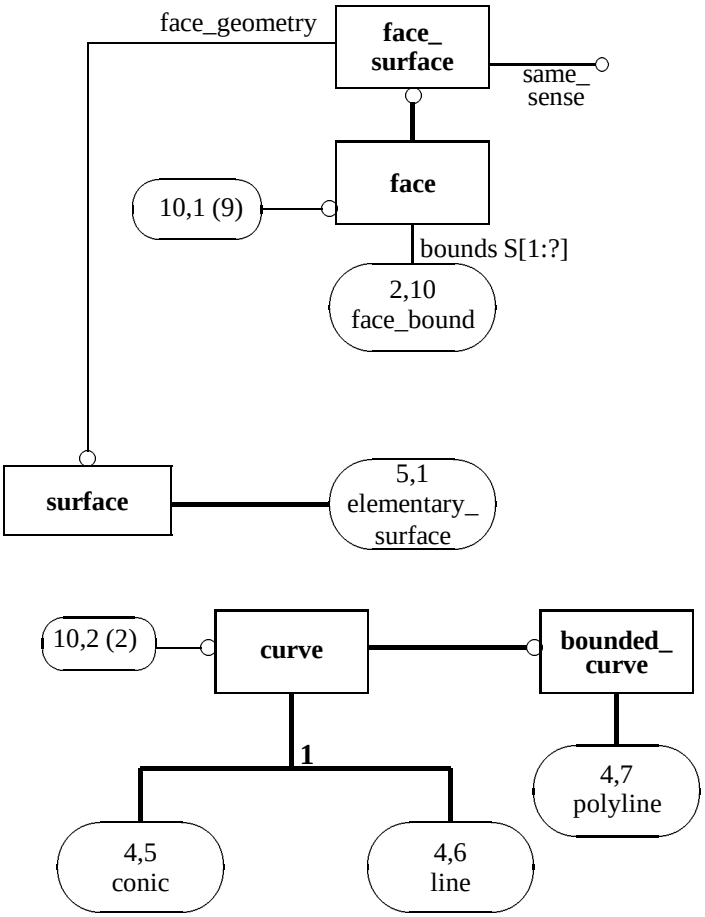


Figure H.10 – AIM EXPRESS-G diagram face and curve in elementary B-rep

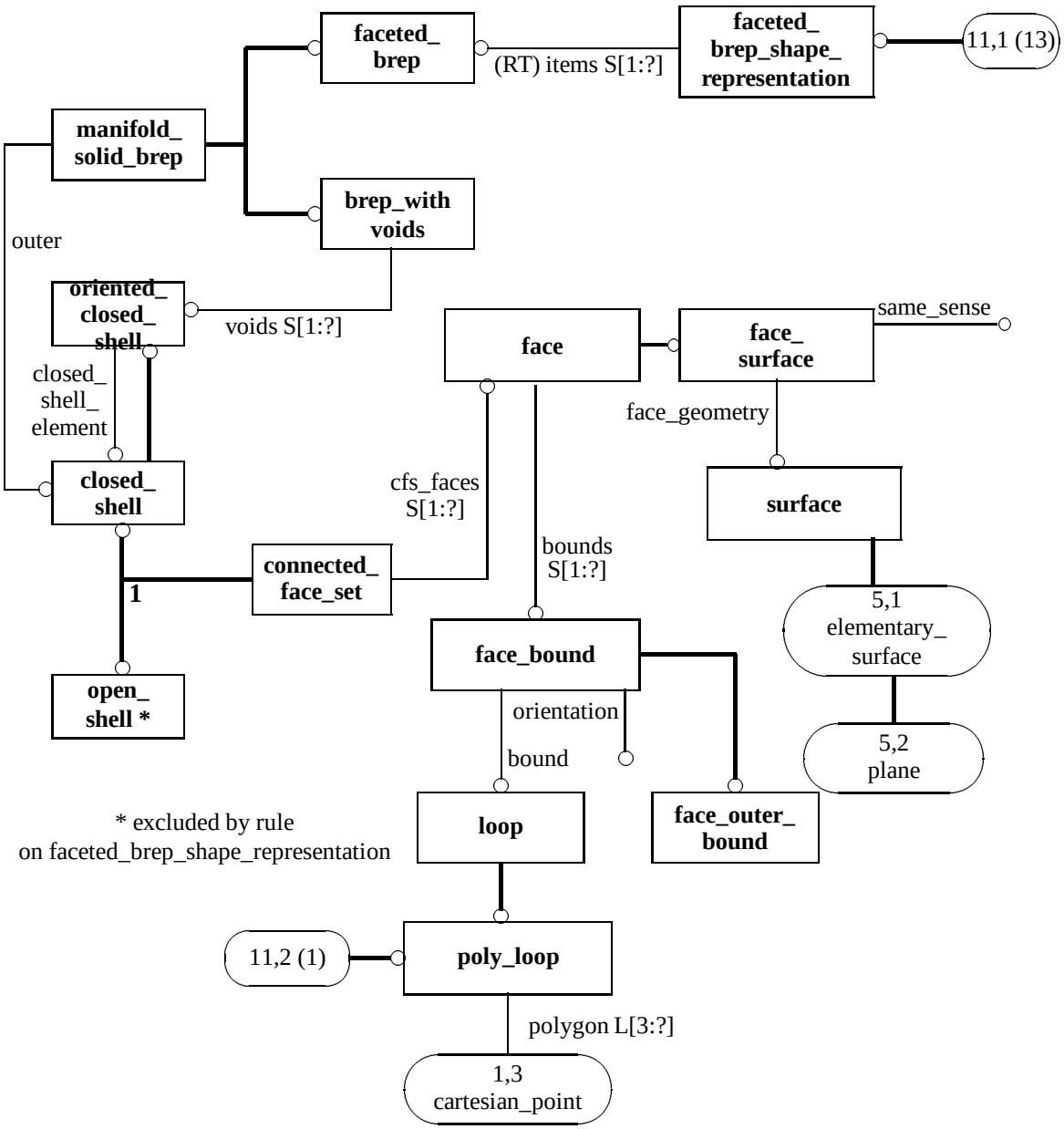


Figure H.11 – AIM EXPRESS-G diagram faceted B-rep

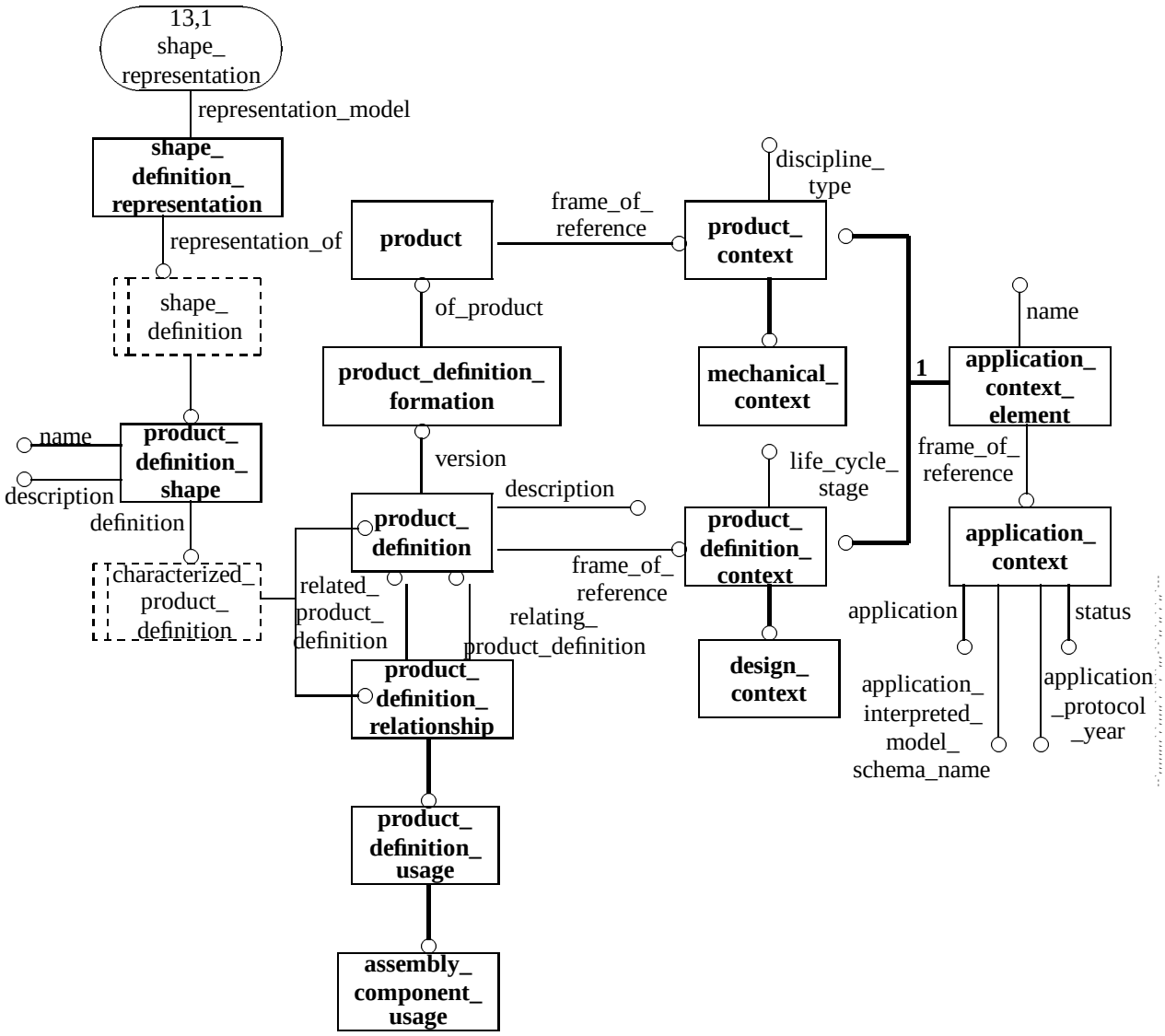


Figure H.12 – AIM EXPRESS-G diagram product structure

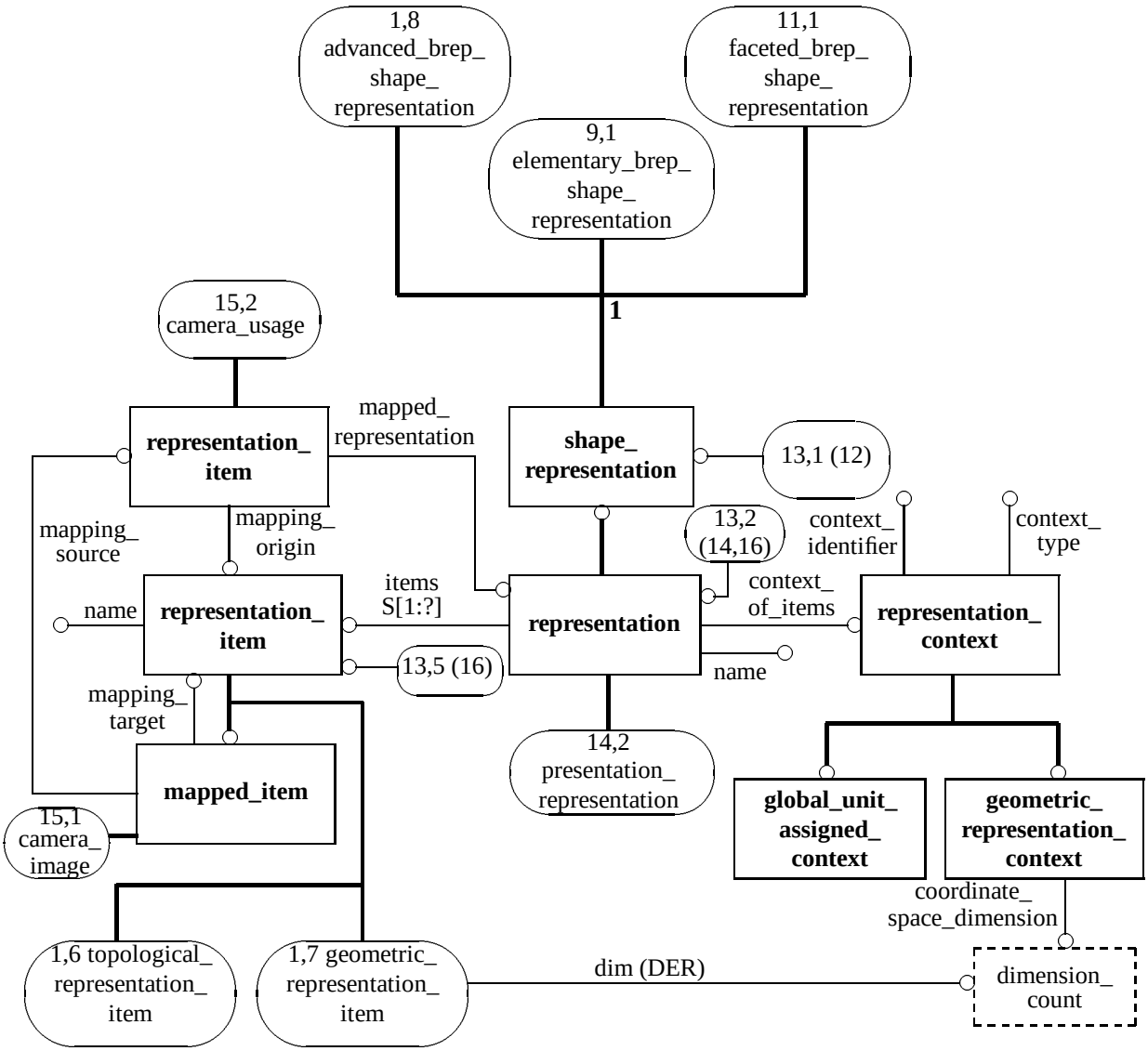


Figure H.13 – AIM EXPRESS-G diagram product structure continued

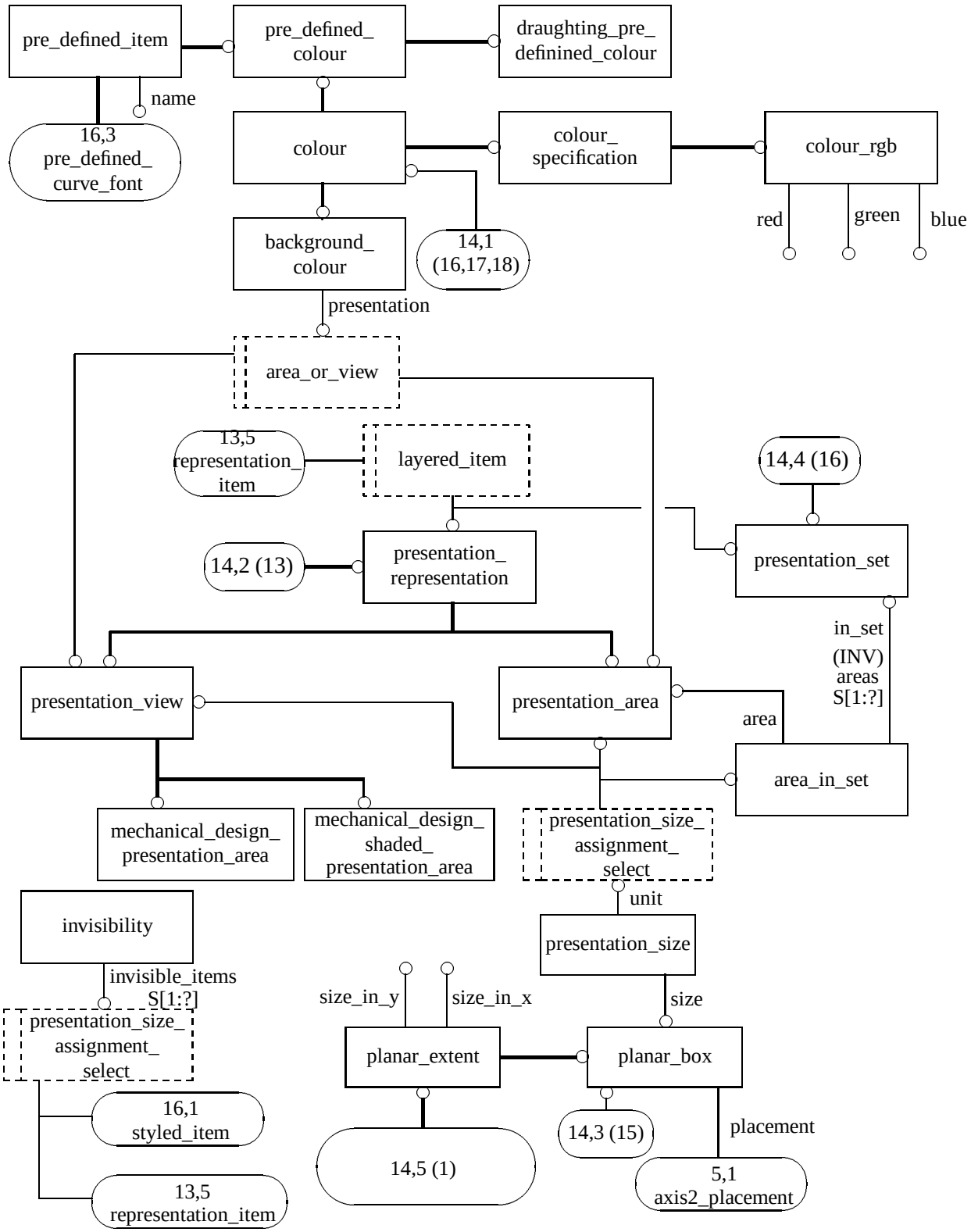


Figure H.14 – AIM EXPRESS-G diagram visual presentation

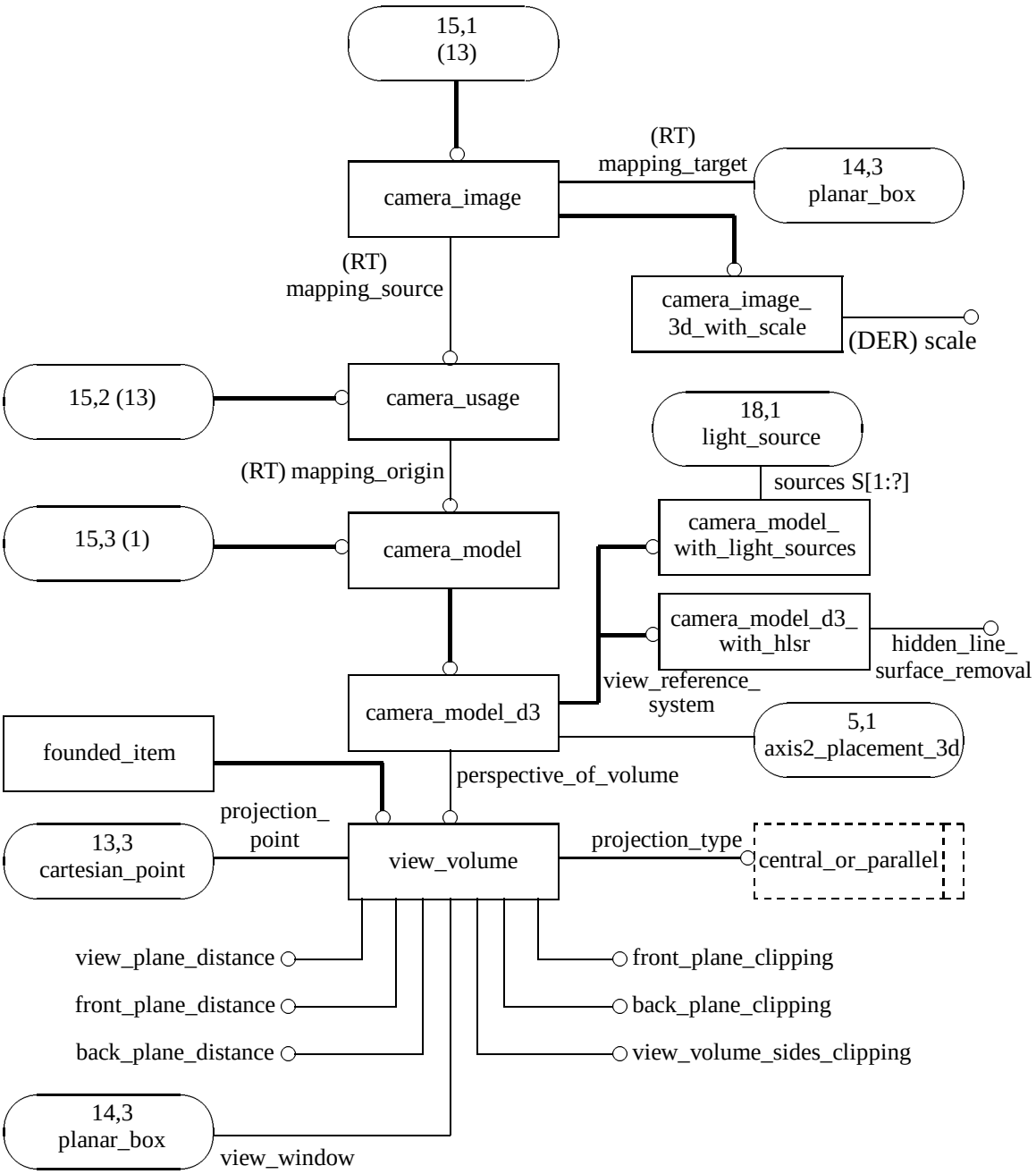


Figure H.15 – AIM EXPRESS-G diagram camera model and projection

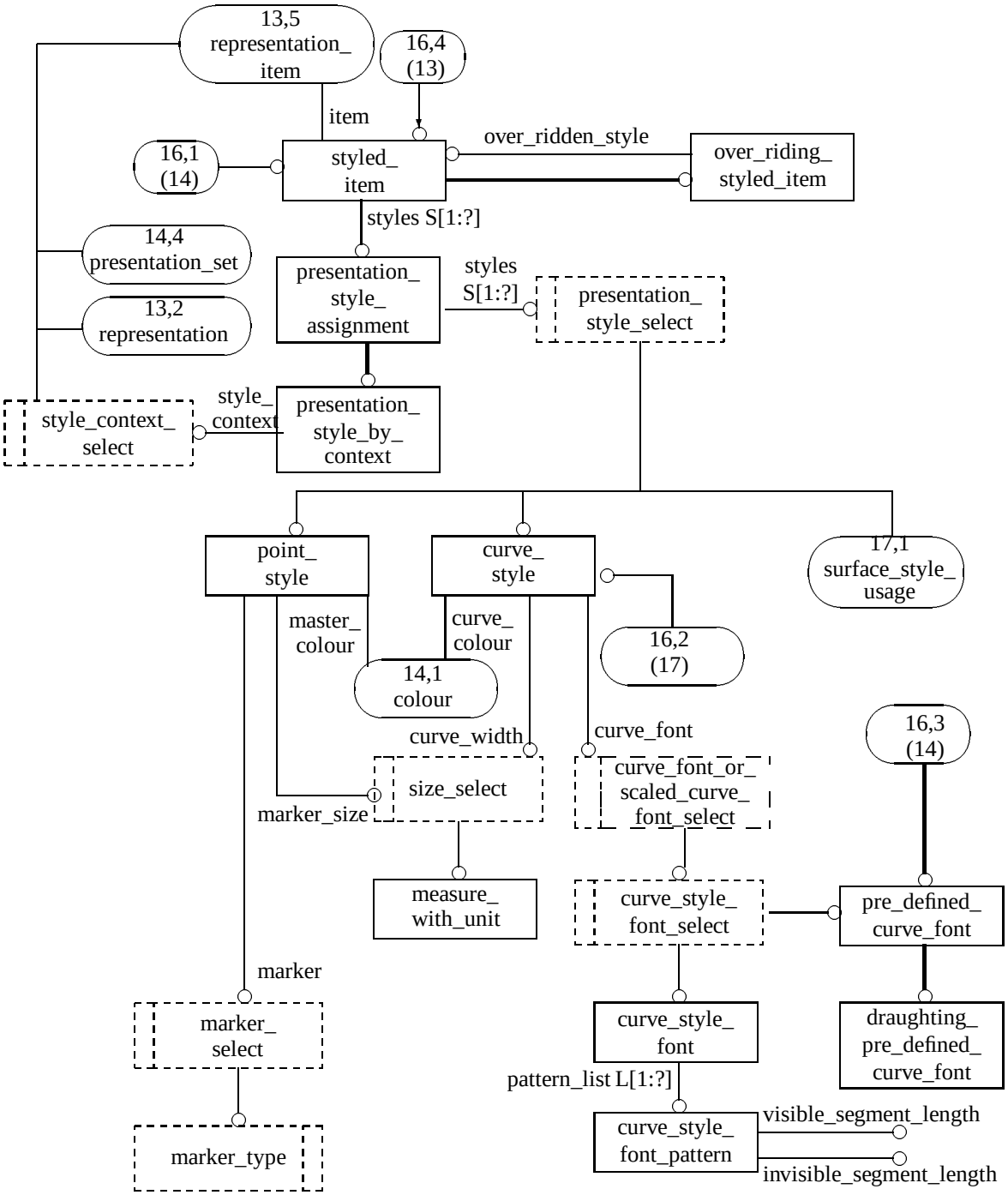


Figure H.16 – AIM EXPRESS-G diagram point and curve styles

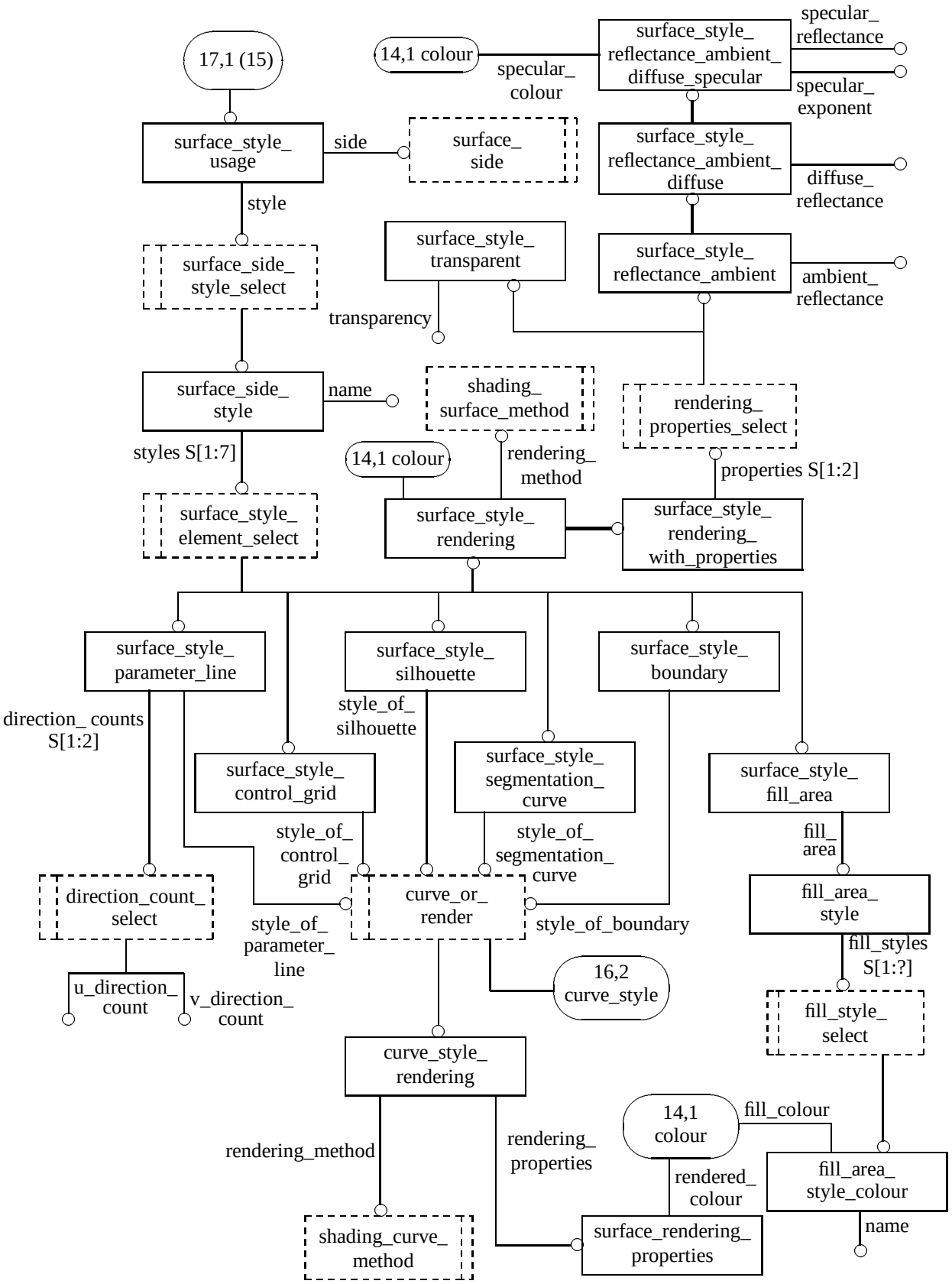


Figure H.17 – AIM EXPRESS-G diagram surface styles

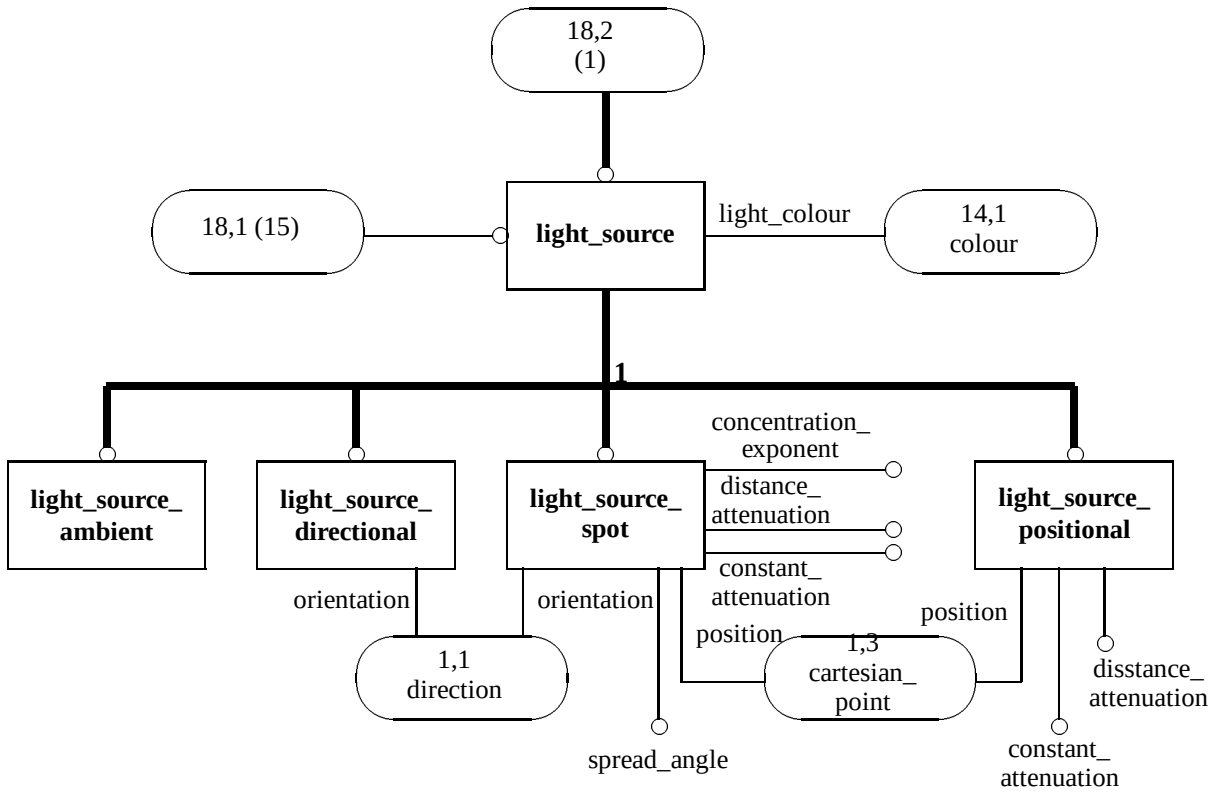


Figure H.18 – AIM EXPRESS-G diagram visual presentation concluded

Annex J
(informative)

Computer interpretable listing

This annex provides a listing of the complete EXPRESS schema specified in annex A of this part of ISO 10303 without comments or other explanatory text. It also provides a listing of the EXPESS entity names and corresponding short names as specified in annex b of this part of ISO 10303. The content of this annex is provided only in computer-interpretable form and can be found at the following URLs:

– Short names: <http://www.tc184-sc4.org/Short_Names/>

– EXPRESS: <http://www.tc184-sc4.org/EXPRESS/>

If there is difficulty accessing these sites contact ISO Central Secretariat or contact the ISO TC 184/SC4 Secretariat directly at: sc4sec@tc184-sc4.org.

NOTE 1 The information provided in computer interpretable form at the above URLs is informative. The information that is contained in the body of this part of ISO 10303 is normative.

Annex K
(informative)

Technical discussions

K.1 Geometric shape description alternatives

The different geometric shape descriptions in Figure F.1 distinguish themselves in model criteria like the quality of the completeness, conciseness, degrees of freedom, and complexity. Dependent upon the application, one or another geometric shape representation will satisfy the application requirements. In this application protocol only volume-based design using B-rep models is supported.

K.1.1 Volume-based model design

Volume-based design is applied to the design process for parts which have complex topology, as well as complex geometry.

Volume-based design is applied in industry using two practical solid modelling techniques. These are:

A1: Boundary representation solid modelling (B-rep).This includes faceted or polyhedron B-rep models.

A2: Constructive Solid Geometry modelling (CSG).

A B-rep solid model can capture the complete topological description of different shape aspects of a part. It contains geometric surfaces and curves of its boundaries in an explicit representation. The individual elements of a B-rep model may be associated with additional model information, such as dimensional tolerance values.

The CSG solids contain boolean composition trees of basic volume primitives. The explicit topology of a part, modelled with CSG can only be obtained when the CSG tree is evaluated.

Dual representation using both B-rep and CSG models is beyond the scope of this part of ISO 10303.

K.2 Known issues

The following technical issues arose during the development of this application protocol.

K.2.1 Face outer bound

For some types of face surface geometry (e.g., a cylindrical surface), the face may have two or more boundaries without the possibility of identifying an ‘outer boundary’. Possible resoloutions which were considered to this issue were to totally exclude all usage of the specialised **face_outer_bound** subtype, to make the usage entirely optional, or, to require this designation when appropriate.

Resolution:
In cases where the face surface is planar the outer bound can always be identified and this AP therefore requires the usage of **face_outer_bound** in the definition of all faces of a **faceted_brep**. For other types of B-rep, the conformance requirements should specify the designation of an outer bound when appropriate.

K.2.2 Complexity of geometry

Part 42 defines many types of curve and surface; not all of these are implemented in the current generation of 3D modelling systems. What criteria should be used in selecting curve and surface types for B-rep level-3 implementations?

Resolution:
In order to facilitate exchange between the largest number of modelling systems the selection was based upon the set of curve and surface types which permit the precise representation of complex shapes but which are directly defined. B-spline curves and surfaces are included but all forms of indirect definition are excluded. In particular offset curves, offset surfaces, parameter space curves, and intersection curves are excluded.

NOTE 1 Parameter space curves are introduced as a result of CD ballot.

K.2.3 Representation of faceted models

A faceted B-rep is a special case of the manifold solid B-rep in which all faces are planar. The possibility exists to represent such a model with the **manifold_solid_brep** entity which would require all vertices and edges to be explicitly defined. Alternatively the **faceted_brep** subtype could be used for this purpose, in which case all vertex and edge data is implicit.

Resolution:
Conformance class 1 (B-rep level 1) is defined using the **faceted_brep** subtype. As defined in the resource part, the definition of the plane defining the geometry of each face is optional, since this information can, in theory, be computed from the bounding **poly_loop**s. In practice numerical precision problems make the identification of the plane of the face difficult, and the **faceted_brep_shape_representation** entity defined in this Part of ISO 10303 requires a plane to be defined for each face.

Bibliography

[1] ISO/CD 10303-205, *Industrial automation systems and integration — Product data representation and exchange — Part 205: Application protocol: Mechanical design using surface representation*

[2] ISO/TS 10303-304, *Industrial automation systems and integration — Product data representation and exchange — Part 304: Abstract test suite: Mechanical design using boundary representation*

[3] *Initial Graphics Exchange Specification (IGES) version 5.1;* National Institute of Standards and Technology, 1991

[4] IDEF0 *Federal Information Processing Standards Publication 183, Integration Definition for Function Modelling (IDEF0)* FIPS PUB 183, National Institute of Standards and Technology, December 1993

[5] BARTELS, BEATTY and BARSKY. *'Splines in Computer Graphics and Geometric Modelling';* Morgan Kaufman, 1987

[6] FARIN, G. *'Curves and Surfaces for Computer Aided Geometric Design';* Academic Press, 1988

[7] SCHLECHTENDAHL, E. G. (Ed.), *'Specification of a CAD*I neutral file for CAD Geometry, Version 3.3';* ISBN 3-540-50392-7

[8] SCHLECHTENDAHL, E.G; WEICK, W. *'Esprit Contributions to the Exchange of CAD models';* CADCAM Yearbook 1990

[9] WEICK, W *'Application Protocol for mechanical design using B-reps, CADEX version 4.0';* January 1992

[10] WILSON, P. R. *'Euler Formulas and Geometric Modeling';* IEEE Computer Graphics & Applications, Vol 5, No 8, pp. 24-36, August 1985

Index

ICS 25.040.40

Price based on 258 pages