INTERNATIONAL STANDARD

ISO 10303-225

First edition
1999-12-01

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

Part 225: Application protocol: Building elements using explicit shape representation

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

Partie 225: Protocole d'application: Éléments de construction utilisant la représentation des formes explicites

Reference number
ISO 10303-225:1999(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

Page

Annexes

Figures

Tables

Foreword

The International Organization for Standardization (ISO) 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.

Draft International Standards adopted by 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.

International Standard ISO 10303-225 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC4, *Industrial data*.

ISO 10303 consists of the following parts under the general title *Industrial automation systems and integration - Product data representation and exchange*:

— Part 1, Overview and fundamental principles;

— Part 11, Description methods: The EXPRESS language reference manual;

— Part 12, Description method: The EXPRESS-I language reference manual;

— Part 21, Implementation methods: Clear text encoding of the exchange structure;

— Part 22, Implementation method: Standard data access interface specification;

— Part 23, Implementation method: C++ language binding to the standard data access interface;

— Part 24, Implementation method: C language binding to the standard data access interface;

— Part 26, Implementation method: Interface definition language binding to the standard data access;

— Part 31, Conformance testing methodology and framework: General concepts;

— Part 32, Conformance testing methodology and framework: Requirements on testing laboratories and clients;

— Part 34, Conformance testing methodology and framework: Abstract test methods;

— Part 35, Conformance testing methodology and framework: Abstract test methods for SDAI implementations;

— Part 41, Integrated generic resources: Fundamentals of product description and support;

— Part 42, Integrated generic resources: Geometric and topological representation;

— Part 43, Integrated generic resources: Representation structures;

— Part 44, Integrated generic resources: Product structure configuration;

— Part 45, Integrated generic resource: Materials;

— Part 46, Integrated generic resources: Visual presentation;

— Part 47, Integrated generic resource: Shape variation tolerances;

— Part 49, Integrated generic resource: Process structure and properties;

— Part 101, Integrated application resource: Draughting;

— Part 104, Integrated application resource: Finite element analysis;

— Part 105, Integrated application resource: Kinematics;

— Part 106, Integrated application resource: Building construction core model;

— Part 201, Application protocol: Explicit draughting;

— Part 202, Application protocol: Associative draughting;

— Part 203, Application protocol: Configuration controlled design;

— Part 204, Application protocol: Mechanical design using boundary representation;

— Part 205, Application protocol: Mechanical design using surface representation;

— Part 207, Application protocol: Sheet metal die planning and design;

— Part 208, Application protocol: Life cycle management - Change process;

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

— Part 210, Application protocol: Electronic assembly, interconnet, and packaging design;

— Part 212, Application protocol: Electrotechnical design and installation

— Part 213, Application protocol: Numerical control process plans for machined parts;

— Part 214, Application protocol: Core data for automotive design;

— Part 215, Application protocol: Ship arrangement;

— Part 216, Application protocol: Ship moulded forms;

— Part 217, Application protocol: Ship piping;

— Part 218, Application protocol: Ship structures;

— Part 221, Application protocol: Functional data and their schematic representation for process plant;

— Part 222, Application protocol: Exchange of product data for composite structures;

— Part 223, Application protocol: Exchange of design and manufacturing product information for casting parts;

— Part 224, Application protocol: Mechanical product definition for process plans using machining features;

— Part 225, Application protocol: Building elements using explicit shape representation;

— Part 226, Application protocol: Ship mechanical systems;

— Part 227, Application protocol: Plant spatial configuration;

— Part 229, Application protocol: Exchange of design and manufacturing product information for forged parts;

— Part 230, Application protocol: Building structural frame: Steelwork;

— Part 231, Application protocol: Process engineering data: Process design and process specification of major equipment;

— Part 232, Application protocol: Technical data packaging core information and exchange;

— Part 301, Abstract test suite: Explicit draughting;

— Part 302, Abstract test suite: Associative draughting;

— Part 303, Abstract test suite: Configuration controlled design;

— Part 304, Abstract test suite: Mechanical design using boundary representation;

— Part 305, Abstract test suite: Mechanical design using surface representation;

— Part 307, Abstract test suite: Sheet metal die planning and design;

— Part 308, Abstract test suite: Life cycle management - Change process;

— Part 309, Abstract test suite: Composite and metallic structural analysis and related design;

— Part 310, Abstract test suite: Electronic assembly, interconnect, and packaging design;

— Part 312, Abstract test suite: Electrotechnical design and installation;

— Part 313, Abstract test suite: Numerical control process plans for machined parts;

— Part 314, Abstract test suite: Core data for automotive mechanical design;

— Part 315, Abstract test suite: Ship arrangement;

— Part 316, Abstract test suite: Ship moulded forms;

— Part 317, Abstract test suite: Ship piping;

— Part 318, Abstract test suite: Ship structures;

— Part 321, Abstract test suite: Functional data and their schematic representation for process plant;

— Part 322, Abstract test suite: Exchange of product data for composite structures;

— Part 323, Abstract test suite: Exchange of design and manufacturing product information for casting parts;

— Part 324, Abstract test suite: Mechanical product definition for process plans using machining features;

— Part 325, Abstract test suite: Building elements using explicit shape representation;

— Part 326, Abstract test suite: Ship mechanical systems;

— Part 327, Abstract test suite: Plant spatial configuration;

— Part 329, Abstract test suite: Exchange of design and manufacturing product information for forged parts;

— Part 330, Abstract test suite: Building structural frame: Steelwork;

— Part 331, Abstract test suite: Process engineering data: Process design and process specification of major equipment;

— Part 332, Abstract test suite: Technical data packaging core information and exchange;

— Part 501, Application interpreted construct: Edge-based wireframe;

— Part 502, Application interpreted construct: Shell-based wireframe;

— Part 503, Application interpreted construct: Geometrically bounded 2D wireframe;

— Part 504, Application interpreted construct: Draughting annotation;

— Part 505, Application interpreted construct: Drawing structure and administration;

— Part 506, Application interpreted construct: Draughting elements;

— Part 507, Application interpreted construct: Geometrically bounded surface;

— Part 508, Application interpreted construct: Non-manifold surface;

— Part 509, Application interpreted construct: Manifold surface;

— Part 510, Application interpreted construct: Geometrically bounded wireframe;

— Part 511, Application interpreted construct: Topologically bounded surface;

— Part 512, Application interpreted construct: Faceted boundary representation;

— Part 513, Application interpreted construct: Elementary boundary representation;

— Part 514, Application interpreted construct: Advanced boundary representation;

— Part 515, Application interpreted construct: Constructive solid geometry;

— Part 517, Application interpreted construct: Mechanical design geometric presentation;

— Part 518, Application interpreted construct: Mechanical design shaded representation.

The structure of this International Standard is described in ISO 10303-1. The numbering of the parts of the International Standard reflects its structure:

— Parts 11 to 12 specify the description methods,

— Parts 21 to 26 specify the implementation methods,

— Parts 31 to 35 specify the conformance testing methodology and framework,

— Parts 41 to 49 specify the integrated generic resources,

— Parts 101 to 106 specify the integrated application resources,

— Parts 201 to 232 specify the application protocols,

— Parts 301 to 332 specify the abstract test suites, and

— Parts 501 to 518 specify the application interpreted constructs.

Should further parts be published, they will follow the same numbering pattern.

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

Introduction

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

This International Standard is organized as a series of parts, each published separately. The parts of ISO 10303 fall into one of the following series: description methods, integrated resources, application interpreted constructs, application protocols, abstract test suites, implementation methods, and conformance testing. The series are described in ISO 10303-1. This part of ISO 10303 is a member of the application protocol series.

This part of ISO 10303 specifies an application protocol (AP) for the exchange of building element shape, property, and spatial configuration information between architecture, engineering, and construction (AEC) application systems and related systems using explicit three-dimensional shape representations.

Many different institutions are engaged in the design, erection, management, and maintenance of buildings and constructions. As these institutions shift toward the increased application of three-dimensional Computer-Aided Design (CAD) modelling systems, data exchange is of crucial importance for all participants in the design and construction process. A fundamental component of the data exchange is the shape and arrangement of the elements of which a building is composed. This part of ISO 10303 specifies a standard which allows the exchange of this data between systems.

Figure 1 contains the data planning model that provides a high level description of the requirements for this application protocol, as well as the relationships between the basic data components.

The planning model illustrates that a building complex consists of buildings and a site shape; a building consists of building sections; and a building section is composed of building items. Each building item is composed of shape component that may be explicitly represented by a faceted, an elementary, or an advanced shape. A faceted shape consists of lines and planes. An elementary shape consists of lines, planes, circles, ellipses, hyperbolas, parabolas, b-spline curves, cylindrical surfaces, conical surfaces, toroidal surfaces, and spherical surfaces. An advanced shape can consist of any kind of curve or surface defined in ISO 10303-42. Figure 2 illustrates the nested relationship between these shapes; the elements that compose a faceted shape are a subset of those that compose an elementary shape, and the elements that compose an elementary shape are a subset of those that compose an advanced shape. Both building elements and shape components can be further described by properties and classifications.

This AP does not address functional or performance characteristics of building systems or the information required for design analysis such as structural or thermal analysis.

This application protocol defines the context, scope, and information requirements for the exchange of building elements using explicit shape representation and specifies the integrated resources necessary to satisfy these requirements.

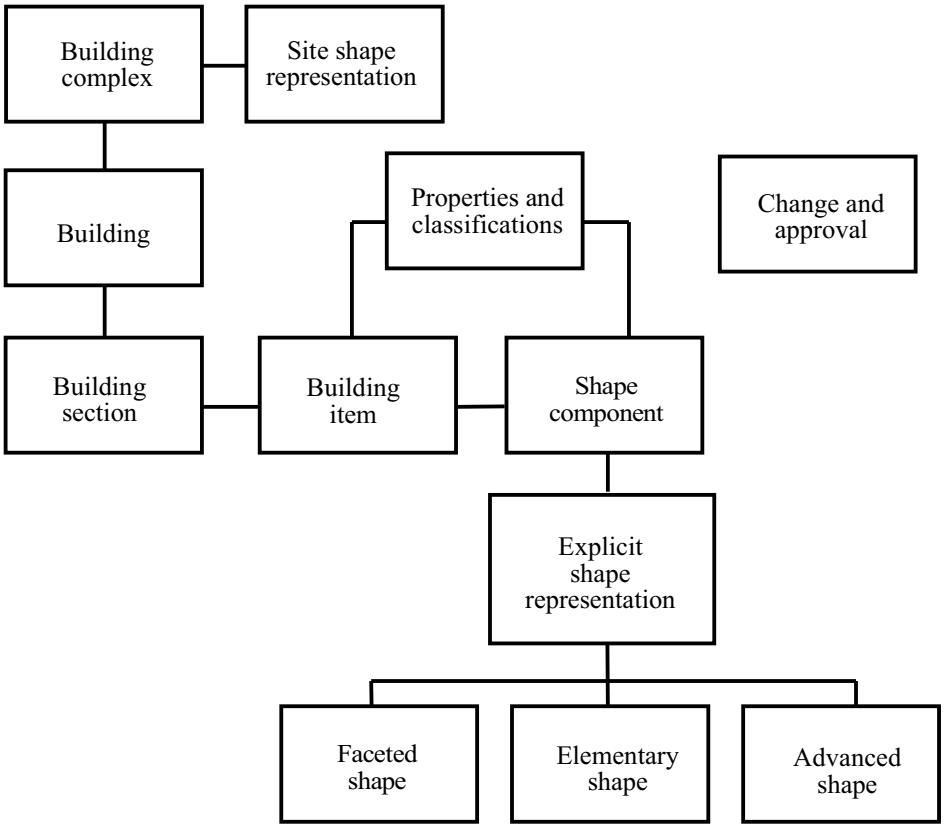


Figure 1 - Data planning model

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

Clause 1 defines the scope of the application protocol and summarizes the functionality and data covered by the AP. Clause 3 lists the words defined in this part of ISO 10303 and gives pointers to word defined elsewhere. An application activity model that is the basis for the definition of the scope is provided in annex F. The information requirements of the application are specified in clause 4 using terminology appropriate to the application. A graphical representation of the information requirements, referred to as the application reference model, is given in annex G.

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

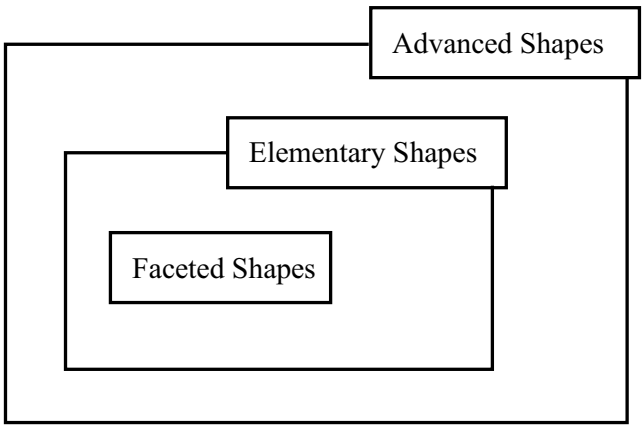


Figure 2 - Nested relationship of shape representation functionality

Industrial automation systems and integration — Product data representation and exchange — Part 225: Application protocol: Building elements using explicit shape representation

1 Scope

This part of ISO 10303 specifies the use of the integrated resources necessary for the scope and information requirements for the exchange of building element shape, property, and spatial configuration information between application systems with explicit shape representations. Building elements are those physical things of which a building is composed, such as structural elements, enclosing and separating elements, service elements, fixtures and equipment, and spaces.

NOTE 1 - See 3.3.1.3 for definitions of these terms and concepts.

Building element shape, property, and spatial configuration information requirements can be used at all stages of the life cycle of a building, including the design process, construction, and maintenance. Building element shape, property, and spatial configuration information requirements specified in this part of ISO 10303 support the following activities:

— concurrent design processes or building design iterations;

— integration of building structure designs with building systems designs to enable design analysis;

— building design visualization;

— specifications for construction and maintenance;

— analysis and review.

NOTE 2 - "Support" of these activities does not imply satisfaction. Satisfaction of the information requirements for construction, for example, would require a complete building design. This part of ISO 10303 only satisfies a portion of information requirements for this activity. See 3.3.1.1.

EXAMPLE 1 - A design analysis function combines the building structure design with building service systems designs (for systems such as heating, ventilation, and air condition (HVAC) and piping) to check for interferences of the structural elements with piping and air conditioning elements.

NOTE 3 - The application activity model in annex F provides a graphical representation of the processes and information flows that are the basis for the definition of the scope of this part of ISO 10303.

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

— explicit representation of the three-dimensional shape of building elements using boundary representation (B-rep) solid models, swept solid models, or constructive solid geometry (CSG) models.

— the spatial configuration of building elements that comprise the assembled building;

— building structures that represent physically distinct buildings that are part of a single building complex;

— non-structural elements that enclose a building or separate areas within a building;

— the shape and arrangement of equipment and service elements that provide services to a building;

EXAMPLE 2 - Service elements include items such as plumbing, ductwork, and conduits. Equipment includes items such as compressors, furnaces, or water heaters.

— the shape and arrangement of fixtures in a building;

EXAMPLE 3 - Fixtures include items such as furniture and installed items like doorknobs.

— specification of spaces and levels;

EXAMPLE 4 - Spaces include rooms, accesses, and hallways. Levels include concepts such as floors and mezzanines of a building.

— the shape of the site on which the building will be erected;

— specification of properties of building elements, including material composition;

— specification of classification information;

EXAMPLE 5 - Elements may be classified for reasons which include cost analysis, acoustics, or safety.

— association of properties and classification information to building elements;

— changes to building element shape, property, and spatial configuration information;

— association of approvals with building element shape, property, and spatial configuration information;

— as-built record of the building.

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

— 2D shape representation and draughting presentation;

— the contents of building standards;

— implicit representation of building elements through selection of standard parameters;

— structural analysis of building structures, including loads, connections, and material properties required for analysis;

— thermal analysis of buildings;

— the assembly process, joining methods, and detailed connectivity of building elements;

— building maintenance history, requirements, and instructions;

— approval, revision, versioning, and design change histories;

— building elements without explicit shape representation;

— bills of quantities.

NOTE 4 - In industries other than AEC, bills of quantities are often referred to as bills of material.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO 10303. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO 10303 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 8824-1:1994 *Information technology - Open systems interconnection - Abstract syntax notation one (ASN.1) - Part 1: Specification of basic notation.*

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

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

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

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

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

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

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

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

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

3 Definitions and abbreviations

3.1 Terms defined in ISO 10303-1

This part of ISO 10303 makes use of the following terms defined in ISO 10303-1:

— application;

— application activity model (AAM);

— application interpreted model (AIM);

— application object;

— application protocol (AP);

— application reference model (ARM);

— conformance testing;

— implementation method;

— integrated resource;

— PICS proforma;

— product;

— product data;

— unit of functionality (UoF).

3.2 Terms defined in ISO 10303-31

This part of ISO 10303 makes use of the following terms defined in ISO 10303-31:

— abstract test suite (ATS);

— conformance class;

— implementation under test (IUT);

— protocol information and conformance statement (PICS).

3.3 Other definitions

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

3.3.1 building design: the information that constitutes a specification sufficient for the evaluation, construction, and maintenance of a building.

3.3.2 building component: named features that represent functional characteristics of the building element.

EXAMPLE 6 - A beam may be composed of additive components such as a shaft and brackets and negative subtractive components such as notches. The components are treated as independent items for design changes and manipulation, such as for moving the notch on the beam.

3.3.3 enclosing and separating elements: physical, non-structural building elements. Enclosing elements are attached to the exterior of a building for weather protection or appearance. Separating elements divide interior spaces creating compartmentation of the structure.

NOTE - Structural elements may serve enclosing or separating functions. Enclosing and separating elements are distinguished by the fact that they do not carry loads. Enclosing elements may also serve separating functions.

3.3.4 explicit shape representation: the geometric data that describes or defines the three-dimensional volumetric boundaries (i.e., the shape) of a building element.

3.3.5 fixtures and equipment: discrete building element that are installed within a building to provide a function or service.

EXAMPLE 7 - Doors, windows, and furniture are examples of fixtures and equipment.

3.3.6 service elements: components of systems that provide services to a building.

EXAMPLE 8 - Plumbing, power, heating, and cooling are services typical in buildings.

3.3.7 spaces: areas or volumes of a building formed by structural, enclosing, or separating element.

EXAMPLE 9 - Spaces include interior and exterior areas and volumes such as rooms, hallways, clearance spaces, loading docks, offices, and entrances.

3.3.8 spatial configuration: the position and orientation of a building element within the spatial or geometric context of an aggregate object which is comprised of the elements.

EXAMPLE 10 - One may refer to the "spatial configuration" of the building elements within a building section, for example.

3.3.9 structural elements: load bearing elements of a building.

EXAMPLE 11 - Load bearing elements of a building include things like walls, floors, columns, beams, and foundations.

3.4 Abbreviations

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

AAM	Application Activity Model
AEC	Architecture, Engineering, and Construction
AIC	Application Interpreted Construct
AIM	Application Interpreted Model
AP	Application Protocol
ARM	Application Reference Model
ATS	Abstract Test Suite
B-rep	Boundary representation
CAD	Computer-Aided Design/Draughting
GIS	Geographic Information System
CSG	Constructive Solid Geometry
HVAC	Heating, Ventilation, and Air Conditioning
PICS	Protocol Implementation and Conformance Statement
UoF	Unit of Functionality
URL	Uniform Resource Locator

4 Information requirements

This clause specifies the information required for the exchange of building element shape, property, and spatial configuration information between application systems using explicit shape 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 terminology of the subject area of this application protocol.

NOTES

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

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

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.

There are two fundamental assumptions that underlie the requirements in this clause. The first assumption is this application protocol will be used for data exchange by basic AEC CAD systems. The second assumption is that the geometry requirements for these application systems will be met by ISO 10303-42; detailed requirements for geometry, therefore, are not included in this part of ISO 10303.

4.1 Units of functionality

This subclause specifies the units of functionality for the building elements using explicit shape representation application protocol. This part of ISO 10303 specifies the following units of functionality:

— advanced_geometric_representation;

— building_component;

— building_composition;

— building_items;

— design_administration;

— elementary_csg_representation;

— elementary_geometric_representation;

— faceted_csg_representation;

— faceted_geometric_representation;

— property_and_classification;

— space_boundary_representation.

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 advanced_geometric_representation UoF

The advanced_geometric_representation UoF are geometric elements that consist of any kind of curve or surface. They are used to represent the complete shape and component shapes of building elements.

The following application objects are used by the advanced_geometric_representation UoF:

— Advanced_b_rep;

— Advanced_curve;

— Advanced_face_with_thickness.

4.1.2 building_component UoF

The building_component UoF represents the component shapes that may be combined to define the shape of a building element. The building component may correspond to the main shape of a building element or any of several positive or negative additions to the main shape. Many building components correspond to common terms such as shaft, bracket, door opening, window, and recess.

EXAMPLE 12 - A block can be used to represent the shape of the shaft as a structural component of a column. Another shape can be used to represent the shape of a positive addition on top of the shaft. A block may be used to represent a recess as a negative component of the column. Figure 3 illustrates the use of components in defining a shape.

The following application objects are used by the building_component UoF:

— Building_element_component;

— Component_location_in_element;

— Component_shape;

— Negative_component;

— Opening;

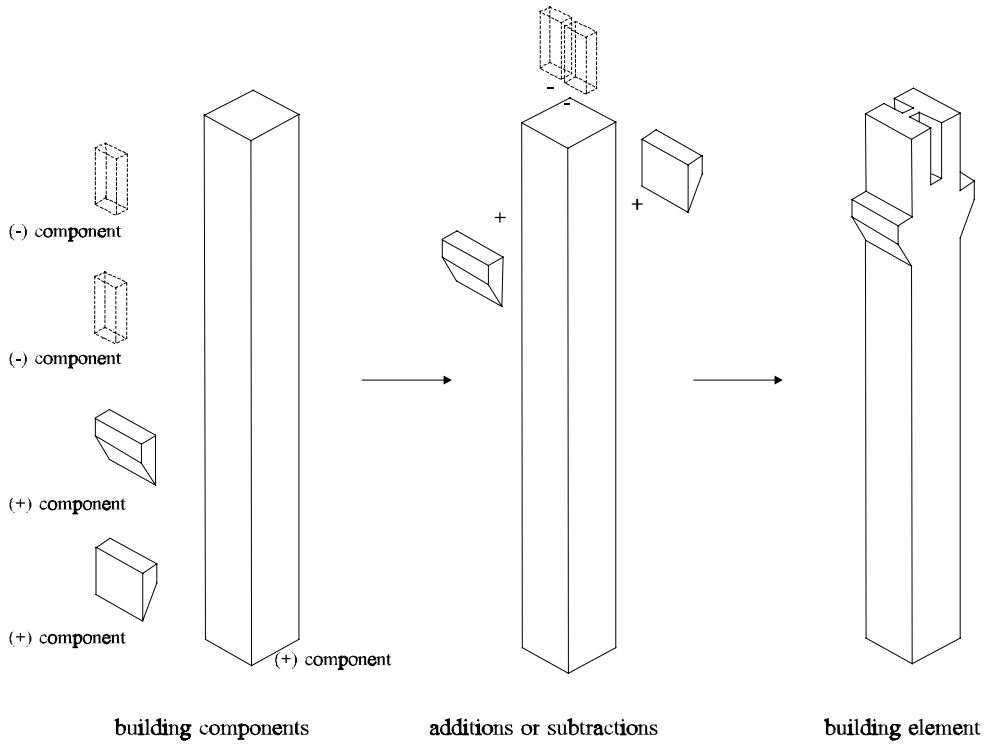


Figure 3 - Building components

— Positive_component;

— Recess.

4.1.3 building_composition UoF

The building_composition UoF represents the collection of building items and assemblies that specifies the arrangement of items within a building complex. A building complex is composed of buildings; a building is decomposed into building sections; each building section is composed of building elements, such as structural elements, enclosing elements, service elements, fixtures, equipment, or spaces. Assemblies or groups of building items can be defined.

The following application objects are used by the building_composition UoF:

— Building;

— Building_complex;

— Building_item;

— Building_level;

— Building_position_in_complex;

— Building_section;

— Gis_position;

— Item_assembly;

— Item_group;

— Item_position_in_section;

— Item_proximity_relationship;

— Level_position_in_section;

— Placement;

— Section_position_in_building;

— Simple_curve;

— Sublevel.

4.1.4 building_items UoF

The building_items UoF represents the discrete, large-scale, shape-related aspects of a building or things in the building. These aspects may be physical items that contribute to the form or function of the building, or may be abstract spaces that describe the usage of the building. A building item is an aspect of a building which has shape and is commonly used in discourse concerning the form or function of a building.

The following application objects are used by the building_items UoF:

— Building_element;

— Building_item;

— Component_shape;

— Component_shape_representation;

— Fixture_equipment_element;

— Service_element;

— Space;

— Structure_enclosure_element.

4.1.5 design_administration UoF

The design_administration UoF represents the management of building element shape, property, and spatial configuration information by specific individual(s). The management includes the approval, modification, acceptance, or confirmation of the building element shape, property, and spatial configuration information. Approvals and change requests are based upon the evaluation criteria derived from the perspective of a given functional discipline. The individual performing the evaluation is recognized, acknowledged, and accepted as qualified to perform the evaluation. The modifications may be necessary to ensure that the building element shape, property, and spatial configuration information meets certain requirements. Change requests can be initiated by almost any person involved in the design process of a building.

EXAMPLE 13 - The structural engineer reviews and evaluates the building design information with respect to structural performance and approves it if it meets the requirements specified in building design standards.

The following application objects are used by the design_administration UoF:

— Approval;

— Building_document_reference;

— Building_item_identification;

— Change_request.

4.1.6 elementary_csg_representation UoF

The elementary_csg_representation UoF consists of CSG primitives which are bounded by simple geometric surfaces such as cylinders, cones, torii, and spheres. The primitives are used to represent the complete shapes and component shapes of the building elements; they can be combined using boolean operations.

The following application objects are used by the elementary_csg_representation UoF:

— Elementary_face_with_thickness;

— Right_circular_cylinder;

— Solid_of_linear_extrusion;

— Solid_of_revolution;

— Trimmed_sphere;

— Trimmed_torus;

— Truncated_cone.

4.1.7 elementary_geometric_representation UoF

The elementary_geometric_representation UoF are geometric curves (such as lines and circles) and surfaces (such as planes, cylinders and cones) that are used to represent the complete shapes and component shapes of the building elements.

NOTE - An elementary_geometric_representation does not include advanced surfaces such as b-splines.

The following application objects are used by the elementary_geometric_representation UoF:

— Elementary_b_rep;

— Elementary_curve.

4.1.8 faceted_csg_representation UoF

The faceted_csg_representation UoF consists of CSG primitive elements that are bounded by planar surfaces. The primitives are used to represent the complete shapes and component shapes of building elements; they can be combined using boolean operations.

The following application objects are used by the faceted_csg_representation UoF:

— Block;

— Truncated_pyramid.

4.1.9 faceted_geometric_representation UoF

The faceted_geometric_representation UoF are geometric elements that consist of straight line segments and elements that are bounded by planar faces. It is a representation that may consist of:

— straight lines;

— planes bounded by straight lines;

— volumes bounded by planes.

They are used to represent the component shapes of Building_item objects.

The following application objects are used by the faceted_geometric_representation UoF:

— Facet_trigon;

— Faceted_b_rep;

— Faceted_curve;

— Faceted_face_with_thickness;

— Faceted_surface_representation;

— Point;

— Point_and_line_representation;

— Polyline;

— Site_shape_representation.

4.1.10 property_and_classification UoF

The property_and_classification UoF represents non-shape information that is associated with building elements and components. The information augments the shape information and is used to complete the specification of the building element, and enables the element to be produced.

> EXAMPLE 14 - Simple material information, such as concrete type, is a property of a wall or a column that may be specified for the approval or construction of a building. Such Properties are commonly specified as grades, ratings, or parameters that are documented in national or international codes and standards.

The following application objects are used by the property_and_classification UoF:

— Item_classification;

— Property.

4.1.11 space_boundary_representation UoF

The space_boundary_representation UoF represents the shape of spaces within a building. Face and shell representations completely represent the shape of the space.

The following application objects are used by the space_boundary_representation UoF:

— Advanced_shell;

— Elementary_shell;

— Faceted_shell;

— Ground_face.

4.2 Application objects

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

4.2.1 Advanced_b_rep

An Advanced_b_rep is a boundary representation solid that is composed of faces with underlying surfaces that may be one of the following: plane, cylindrical surface, conical surface, spherical surface, toroidal surface, or b-spline surface.

NOTE - Compare with the definition of Elementary_b_rep (see 4.2.21) and Faceted_b_rep (see 4.2.26).

4.2.2 Advanced_curve

An Advanced_curve is a one-dimensional path through three-dimensional space that may be mathematically described with one of the following types of curve: line, polyline, circle, ellipse, hyperbola, parabola, or b-spline curve.

NOTE - Compare with definitions of Elementary_curve (see 4.2.22) and Faceted_curve (see 4.2.27).

4.2.3 Advanced_face_with_thickness

An Advanced_face_with_thickness is a three-dimensional shape that has two parallel surfaces of one of the following types: plane, cylindrical surface, conical surface, spherical surface, toroidal surface, or b-spline surface. The surfaces have identical but arbitrary boundaries and are separated by a normal distance that is very small in comparison to the overall dimensions of the two surfaces.

NOTE - Compare with the definition of Elementary_face_with_thickness (see 4.2.23) and Faceted_face_-with_thickness (see 4.2.28).

4.2.4 Advanced_shell

An Advanced_shell is a contiguous, closed collection of topological faces with underlying surfaces that may be of one the following types: plane, cylindrical surface, conical surface, spherical surface, toroidal surface, or b-spline surface.

4.2.5 Approval

An Approval is the acknowledgement that a building element shape, property, and spatial configuration information meets evaluation criteria established from a certain functional perspective. The evaluation and subsequent approval or disapproval is performed by an individual or organization that is recognized and accepted as capable of making the evaluation.

EXAMPLE 15 - An architect has the approval responsibility for ensuring that the building meets community design standards.

The data associated with an Approval are the following:

— approver;

— date;

— purpose;

— status.

4.2.5.1 approver

The approver specifies the individual or organization who made the approval.

4.2.5.2 date

The date specifies the date that the approval was made.

4.2.5.3 purpose

The purpose specifies the functional role or objective of the evaluation or assessment performed in order to make the approval.

4.2.5.4 status

The status specifies the state of the approval.

EXAMPLE 16 - "Approved" and "Not approved" are examples of status.

4.2.6 Block

A Block is a three-dimensional, right, rectangular solid primitive. The size and shape of a Block is described by three real values, which represent the length, width, and height dimensions of the Block.

4.2.7 Building

A Building is a man-made structure erected to house, enclose, protect, or serve one or more human enterprises or endeavours.

The data associated with a Building are the following:

— address;

— description;

— name;

— owner;

— status.

4.2.7.1 address

The address specifies the alphanumeric designation used to identify the location of the building in accordance with local postal codes. The address need not be specified for a particular Building.

4.2.7.2 description

The description specifies a textual explanation of the purpose or function of the building.

4.2.7.3 name

The name specifies a descriptive title assigned to the building.

4.2.7.4 owner

The owner specifies the individual or organization who by legal acknowledgement possesses the building.

4.2.7.5 status

The status specifies a label indicating that the information associated with a building is at or has achieved a certain level of life-cycle maturity.

4.2.8 Building_complex

A Building_complex is a collection of buildings located on a site.

The data associated with a Building_complex are the following:

— description;

— global_position;

— name;

— owner;

— surrounding_grounds_shape.

4.2.8.1 description

The description specifies a textual explanation of the purpose or function of the building_complex.

4.2.8.2 global_position

The global_position specifies the information necessary for positioning the Building_complex on the earth. See 4.3.1 for the application assertion.

4.2.8.3 name

The name specifies a descriptive title assigned to the Building_complex.

4.2.8.4 owner

The owner specifies the individual or organization who by legal acknowledgement possesses the Building_complex.

4.2.8.5 surrounding_grounds_shape

The surrounding_grounds_shape specifies a geometric description that represents the topography of the site on which a Building_complex is located. The surrounding_grounds_shape need not be specified for a particular Building_complex. See 4.3.2 for the application assertion.

4.2.9 Building_document_reference

A Building_document_reference is the identification of a document and a section within the document that pertains to a Building_item (see 4.2.12) or a Building_element_component (see 4.2.11).

The data associated with a Building_document_reference are the following:

— document_type;

— identifier;

— item_in_document.

4.2.9.1 document_type

The document_type specifies a label that specifies a descriptive label that indicates the purpose, function, or role of the document.

4.2.9.2 identifier

The identifier specifies a designation that uniquely identifies a particular Building_document_reference.

4.2.9.3 item_in_document

The item_in_document specifies a paragraph, section, or component of the document.

4.2.10 Building_element

A Building_element is a type of Building_item (see 4.2.12) that is a physical part of a building or building system that has a characteristic function. A Building_element may be one of the following: a Structure_enclosure_element (see 4.2.57), a Service_element (see 4.2.51), or a Fixture_equipment_-element (see 4.2.31).

EXAMPLE 17 - "Characteristic functions" include purposes such as enclosing, supporting, furnishing, and providing a service.

The data associated with a Building_element are the following:

— additions_and_subtractions;

— main_component.

4.2.10.1 additions_and_subtractions

The additions_and_subtractions specifies a collection of Building_element_components objects (see 4.2.11) each of which has an explicit three-dimensional shape; the Building_element_components are added to or subtracted from each other to form the shape of a Building_element. The additions_and_-subtractions need not be specified for a particular Building_element. See 4.3.3 for the application assertion.

NOTE - The aggregate is ordered because the implied boolean operations of addition and subtraction of the positive and negative components are order dependent. Different orderings would result in different shapes for the Building_element.

4.2.10.2 main_component

The main_component specifies the primary shape component of a Building_element. See 4.3.4 for the application assertion.

NOTE - This shape is modified through the additions_and_subtractions to give the final shape of the Building_element.

4.2.11 Building_element_component

A Building_element_component is a basic compositional constituent of the shape of Building_element (see 4.2.10) object that can imply a functional role and a stereotypical shape. A Building_element_-component is either a Positive_component (see 4.2.46) or a Negative_component (see 4.2.40).

EXAMPLE 18 - A Structure_enclosure_element such as column consists of the component's shaft, brackets, and openings.

The data associated with a Building_element_component are the following:

— approval_information;

— component_characterization;

— component_class;

— description;

— document_reference;

— identifier;

— position;

— shape.

4.2.11.1 approval_information

The approval_information specifies a collection of approvals of the Building_element_component objects. The approval_information need not be specified for a particular Building_element_component. See 4.3.5 for the application assertion.

4.2.11.2 component_characterization

The component_characterization specifies a collection of properties of the Building_element_-component. The component_characterization need not be specified for a particular Building_element_-component. See 4.3.11 for the application assertion.

4.2.11.3 component_class

The component_class specifies a collection of designations; each designation classifies a Building_-element_component as a member of some predefined category. The component_class need not be specified for a particular Building_element_component. See 4.3.10 for the application assertion.

4.2.11.4 description

The description specifies a textual explanation of purpose or function of the Building_element_-component.

4.2.11.5 document_reference

The document_reference specifies a document that contains information pertinent to the Building_element_component. The document_reference need not be specified for a particular Building_element_component. See 4.3.6 for the application assertion.

4.2.11.6 identifier

The identifier specifies a Building_item_identification (see 4.2.13) that uniquely identifies a particular Building_element_component. See 4.3.7 for the application assertion.

4.2.11.7 position

The position specifies the location of a Building_element_component in a coordinate space common to other Building_element_component objects which comprise a Building_element (see 4.2.10). The positions of the Building_element_component objects are such that they form an assembly that represents the shape of the Building_element (see 4.2.10). See 4.3.8 for the application assertion.

4.2.11.8 shape

The shape specifies a shape model that defines a volume that represents shape of the Building_element_component. See 4.3.9 for the application assertion.

4.2.12 Building_item

A Building_item is a part of a building that has a characteristic function. A Building_item is either a Building_element (see 4.2.10) or a Space (see 4.2.56).

The data associated with a Building_item are the following:

— approval_information;

— description;

— document_reference;

— identifier;

— item_characterization;

— item_class;

— level_assignment;

— status.

4.2.12.1 approval_information

The approval_information specifies a collection of approvals of the Building_item. The approval_-information need not be specified for a particular Building_item. See 4.3.12 for the application assertion.

4.2.12.2 description

The description specifies a textual explanation of the purpose or function of the Building_item.

4.2.12.3 document_reference

The document_reference specifies a document that contains information pertinent to the Building_item. The document_reference need not be specified for a particular Building_item. See 4.3.13 for the application assertion.

4.2.12.4 identifier

The identifier specifies a Building_item_identification (see 4.2.13) that uniquely identifies a particular Building_item. See 4.3.14 for the application assertion.

4.2.12.5 item_characterization

The item_characterization specifies a collection of properties of the Building_item. An item_-characterization need not be specified for a particular Building_item. See 4.3.17 for the application assertion.

4.2.12.6 item_class

The item_class specifies a collection of designations; each designation classifies a Building_item as a member of some predefined category. The item_class need not be specified for a particular Building_item. See 4.3.16 for the application assertion.

4.2.12.7 level_assignment

The level_assignment specifies the Building_level (see 4.2.14) objects to which a Building_item is assigned. The level_assignment need not be specified for a particular Building_item. See 4.3.15 for the application assertion.

4.2.12.8 status

The status specifies a label indicating that the information associated with a Building_item is at or has achieved a certain level of life-cycle maturity.

4.2.13 Building_item_identification

A Building_item_identification is an alphanumeric designation that uniquely identifies a Building_-section (see 4.2.16), a Building_item (see 4.2.12), a Building_element_component (see 4.2.11), or a Building_level (see 4.2.14). The designation is unique and persistent across all systems that use the building information and throughout the duration of a building construction project.

The data associated with a Building_item_identification are the following:

— administrator;

— item_identifier;

— project.

NOTE - The designation that is unique is the combination of the item_identifier, the administrator assigning the identifier, and the project in which the assignment is made.

4.2.13.1 administrator

The administrator specifies the individual or organization that assigned the identifier to or is responsible for the item.

4.2.13.2 item_identifier

The item_identifier specifies an alphanumeric string assigned to the item.

4.2.13.3 project

The project specifies the building construction project in which the item must be uniquely distinguishable from other items.

4.2.14 Building_level

A Building_level is a part of a Building that consists of one or more areas accessible through primarily horizontal access paths. A Building_level may be a Sublevel (see 4.2.58) or may have Sublevels.

NOTES

1 - A Building can be thought of as being composed of multiple levels, often called "floors" or "storeys". An access to different areas within one level is achieved primarily through horizontal movement; access between Building_levels is achieved through primarily vertical movement. Figure 4 illustrates building levels.

2 - Building_level is the same thing as "storey" as defined in ISO/CD 4157-2. **[3]**

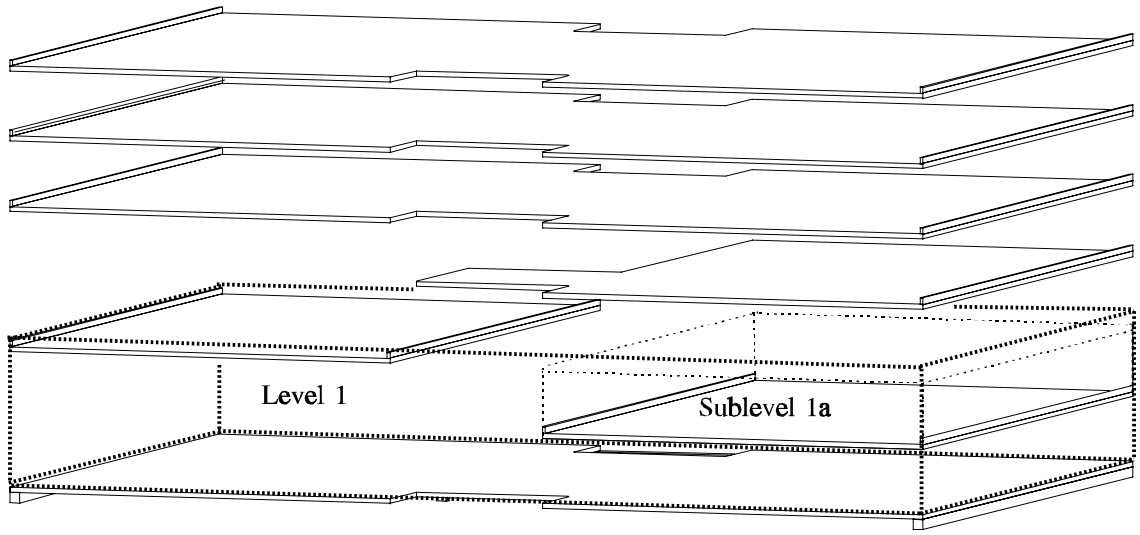


Figure 4 - Building levels

The data associated with a Building_level are the following:

— identifier;

— level_characterization;

— level_class;

— name;

— space_shapes.

4.2.14.1 identifier

The identifier specifies a Building_item_identification (see 4.2.13) that uniquely identifies a particular Building_level. See 4.3.18 for the application assertion.

4.2.14.2 level_characterization

The level_characterization specifies a collection of properties of the Building_level. The level_-characterization need not be specified for a particular Building_level. See 4.3.20 for the application assertion.

4.2.14.3 level_class

The level_class specifies a collection of designations; each designation classifies a Building_level as a member of some predefined category. The level_class need not be specified for a particular Building_level. See 4.3.19 for the application assertion.

4.2.14.4 name

The name specifies a descriptive title of the Building_level.

4.2.14.5 space_shapes

The space_shapes specifies a collection of Ground_face objects (see 4.2.33), Faceted_shell objects (see 4.2.29), Elementary_shell objects (see 4.2.24), or Advanced_shell objects (see 4.2.4) that bound or delimit a volume of space that represents or encloses a Building_level. The space_shapes need not be specified for a particular Building_level. See 4.3.21, 4.3.22, 4.3.23 and 4.3.24 for the application assertion.

4.2.15 Building_position_in_complex

A Building_position_in_complex is a combination of a Building_complex (see 4.2.8) and positioning information that enables a Building (see 4.2.7) to be positioned within the spatial context of the Building_complex. The aggregation of all Building_position_in_complex objects with the same Building_complex as a context constitutes the buildings that comprise the Building_complex.

The data associated with a Building_position_in_complex are the following:

— location;

— positioned_building;

— positioned_within.

4.2.15.1 location

The location specifies the positioning and orienting information required to place a building within the spatial context of the Building_complex. See 4.3.27 for the application assertion.

4.2.15.2 positioned_building

The positioned_building specifies the Building that is to be placed within the spatial context of the Building_complex. See 4.3.25 for the application assertion.

4.2.15.3 positioned_within

The positioned_within specifies the Building_complex that establishes a spatial context in which buildings are positioned. See 4.3.26 for the application assertion.

4.2.16 Building_section

A Building_section is a logical segment of a building. Logical segments of a building are defined based on local coordinate systems for functional or definitional convenience.

> EXAMPLE 19 - Buildings are often designed as individual sections that are integrated into a single structure at a later point in time. The wings of a hospital, library, or office complex are typical building sections. Figure 5 illustrates three building sections; a usage scenario might be that a new section is added every other year to satisfy the spatial needs of a growing company.

The data associated with a Building_section are the following:

— description;

— identifier;

— name;

— status.

4.2.16.1 description

The description specifies a textual explanation of the purpose or function of the Building_section.

4.2.16.2 identifier

The identifier specifies a Building_item_identification (see 4.2.13) that uniquely identifies a particular Building_section. See 4.3.28 for the application assertion.

4.2.16.3 name

The name specifies a descriptive title of the Building_section.

4.2.16.4 status

The status specifies a label indicating that the information associated with a Building_section is at or has achieved a certain level of life-cycle maturity.

4.2.17 Change_request

A Change_request is a modification to some aspect of a building element shape, property, and spatial configuration information, a request for a change to a building element shape, property, and spatial configuration information, or the documentation of a change that has been incorporated.

The data associated with a Change_request are the following:

— approval_information;

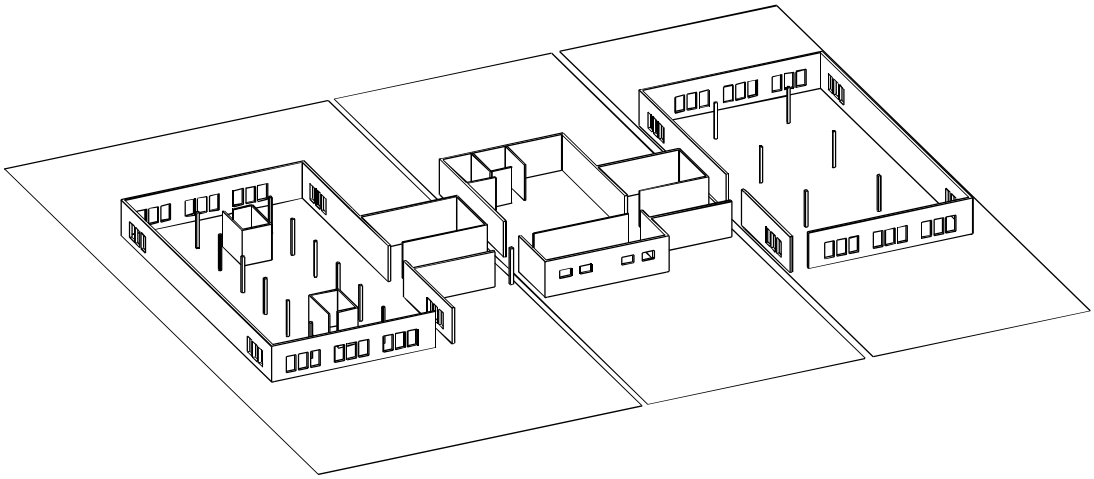


Figure 5 - Building section

— change_from;

— change_to;

— description;

— reason;

— request_date;

— requestor;

— responsibility;

— solution;

— status.

4.2.17.1 approval_information

The approval_information specifies a collection of authorizations to incorporate a requested change into a building element shape, property, and spatial configuration information or the status of the authorization. See 4.3.29 for the application assertion.

4.2.17.2 change_from

The change_from specifies the identifier of some aspect of a building element shape, property, or spatial configuration information that is deemed unsatisfactory. See 4.3.32 for the application assertion.

4.2.17.3 change_to

The change_to specifies the Building_item (see 4.2.12) or the Building_element_component (see 4.2.11) that represents the new condition for some aspect of a building item which is proposed as a replacement for the change_from condition. The change_to need not be specified for a particular Change_request. See 4.3.31 and 4.3.30 for the application assertions.

4.2.17.4 description

The description specifies a request that a change be made or provides an explanation of a change to the building element shape, property, and spatial configuration information.

4.2.17.5 reason

The reason specifies a textual explanation of a problem with the existing building element shape, property, and spatial configuration information and the rationale for changing the design to eliminate the problem.

4.2.17.6 request_date

The request_date specifies the calendar day on which the change is requested.

4.2.17.7 requestor

The requestor specifies the individual or organization that requests that a change to the building element shape, property, and spatial configuration information be made.

4.2.17.8 responsibility

The responsibility specifies the individual or organization that requested the change or is responsible for executing the change.

4.2.17.9 solution

The solution may specify a textual explanation of the action that has been taken with respect to the requested change. The solution need not be specified for a particular Change_request.

4.2.17.10 status

The status specifies a label indicating that the information associated with a Change_request is at or has achieved a certain level of life-cycle maturity.

4.2.18 Component_location_in_element

A Component_location_in_element is the information necessary to position a Building_element_-component (see 4.2.11) within the coordinate space of the Building_element (see 4.2.10) of which it is a part.

4.2.19 Component_shape

A Component_shape is the shape of a Building_element_component (see 4.2.11).

The data associated with a Component_shape are the following:

— representations.

The representations specifies the collection of geometric information that represents the shape of the Building_element_component.

NOTE - The shape of a building_element_component may be represented by one or more shape models. If multiple models are used, they should all represent the same shape; they may represent different views of the same shape, e.g., envelope, outline, detailed.

4.2.20 Component_shape_representation

A Component_shape_representation represents a single, cohesive, explicit representation of the shape of a single Component_shape (see 4.2.19).

NOTE - In use, an instance of this entity may correspond to a geometric model that defines the shape of an object.

The data associated with a Component_shape_representation are the following:

— representation_element;

— representation_type.

4.2.20.1 representation_element

The representation_element specifies the Advanced_b_rep (see 4.2.1), the Advanced_curve (see 4.2.2), the Advanced_face_with_thickness (see 4.2.3), the Block (see 4.2.6), the Elementary_b_rep (see 4.2.21), the Elementary_curve (see 4.2.22), the Elementary_face_with_thickness (see 4.2.23), the Faceted_b_rep (see 4.2.26), the Faceted_curve (see 4.2.27), the Faceted_face_with_thickness (see 4.2.28), the Right_circular_cylinder (see 4.2.49), the Solid_of_linear_extrusion (see 4.2.54), the Solid_-of_revolution (see 4.2.55), the Trimmed_sphere (see 4.2.59), the Trimmed_torus (see 4.2.60), the Truncated_cone (see 4.2.61), or the Truncated_pyramid (see 4.2.62) that is the Component_shape_-representation. See 4.3.34, 4.3.35, 4.3.36, 4.3.37, 4.3.38, 4.3.39, 4.3.40, 4.3.42, 4.3.43, 4.3.41, 4.3.44, 4.3.45, 4.3.46, 4.3.47, 4.3.48, 4.3.49, and 4.3.50 for the application assertions.

4.2.20.2 representation_type

The representation_type specifies the level of detail or precision of the geometric representation. The value of the representation_type is either user supplied or one of a set of predefined values.

NOTE 1 - A user supplied value may be "analysis".

The value of the representation_type attribute may be one of the following:

— detail;

— envelope;

— outline.

NOTE 2 - See 4.2.20.2.1 - 4.2.20.2.3 for the definition of each permissible value for opening_type.

NOTE 3 - Figure 6 illustrates examples of detail, outline, and envelope shapes for an air conditioning unit. Detail shapes are usually used to produce visually accurate pictures. Outline shapes are usually used for rapid display refresh during interactive design. Envelope shapes are usually used for clash detection.

4.2.20.2.1 detail: a three-dimensional spatial volume that corresponds to the complete surface representation (or part of a complete representation) of a Component_shape (see 4.2.19).

4.2.20.2.2 envelope: a simple three-dimensional spatial volume that completely encloses or bounds a Component_shape (see 4.2.19) and may also enclose usage or work spaces.

4.2.20.2.3 outline: a simple three-dimensional spatial volume that encloses a Component_shape (see 4.2.19).

4.2.21 Elementary_b_rep

An Elementary_b_rep is a boundary representation solid model that is composed of faces with underlying surfaces that may be one of the following: plane, cylindrical surface, conical surface, spherical surface, or toroidal surface.

NOTE - Compare with the definition of Advanced_b_rep (see 4.2.1) and Faceted_b_rep (see 4.2.26).

4.2.22 Elementary_curve

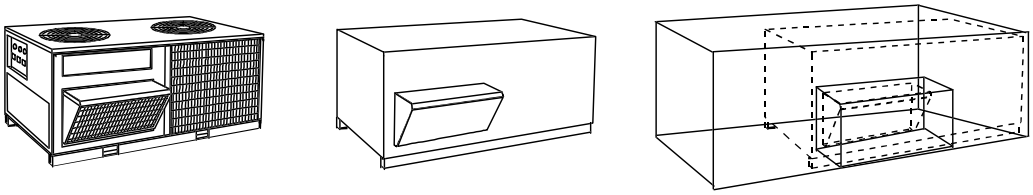
An Elementary_curve is a one-dimensional path through three-dimensional space that may be mathematically described with one of the following types of curve: line, polyline, circle, ellipse, hyperbola, or parabola.

NOTE - Compare with the definition of Advanced_curve (see 4.2.2) and Faceted_curve (see 4.2.22).

4.2.23 Elementary_face_with_thickness

An Elementary_face_with_thickness is a three-dimensional shape that has two parallel surfaces of one of the following types: plane, cylindrical surface, conical surface, spherical surface, or toroidal surface. The surfaces have identical but arbitrary boundaries and are separated by a normal distance that is very small in comparison to the overall dimensions of the two surfaces.

NOTE - Compare with the definition of Advanced_face_with_thickness (see 4.2.3) and Faceted_face_with_-thickness (see 4.2.23).



Detail Outline Envelope

Figure 6 - Detail, outline and envelope shapes

EXAMPLE 20 - The shape of cladding, glazing, and other thin, laminar structures can be most easily represented with a face_with_thickness.

4.2.24 Elementary_shell

An Elementary_shell is a contiguous, closed collection of topological faces with underlying surfaces that may be of one the following types: plane, cylindrical surface, conical surface, spherical surface, or toroidal surface.

4.2.25 Facet_trigon

A Facet_trigon is a triangular, planar surface element with straight-line boundaries.

The data associated with a Facet_trigon are the following:

— border.

The border specifies a collection of three Points (see 4.2.43) that establishes the linear boundaries of the Facet_trigon. Three unique pairs of points can be formed from the three points of the trigon; if more than one Facet_trigon exists, then at least one pair of points shall be shared by an adjacent Facet_trigon such that the points and the line implied by the points constitute a shared boundary of the trigons. See 4.3.51 for the application assertions.

4.2.26 Faceted_b_rep

A Faceted_b_rep is a boundary representation solid that is composed of planar, polygonal faces.

4.2.27 Faceted_curve

A Faceted_curve is a one-dimensional path through three-dimensional space that may be mathematically described with one of the following types of curve: line or polyline.

NOTE - Compare with the definition of Advanced_curve (see 4.2.2) and Elementary_curve (see 4.2.22).

4.2.28 Faceted_face_with_thickness

A Faceted_face_with_thickness is a three-dimensional shape that has two parallel, planar surfaces. The surfaces have identical but arbitrary boundaries composed of line or polylines and are separated by a normal distance that is very small in comparison to the overall dimensions of the two surfaces.

NOTE - Compare with the definition of Advanced_face_with_thickness (see 4.2.3) and Elementary_face_-with_thickness (see 4.2.23).

4.2.29 Faceted_shell

A Faceted_shell is a contiguous, closed collection of topological faces in which each face is planar and has a polygonal boundary.

4.2.30 Faceted_surface_representation

A Faceted_surface_representation is a type of Site_shape_representation (see 4.2.53). A Faceted_-surface_representation is an open surface that is composed of a connected set of planar, triangular faces.

The data associated with a Faceted_surface_representation are the following:

— facets.

The facets specifies the collection of planar, triangular faces that represent the contour of the terrain on which a building is or will be located. See 4.3.52 for the application assertion.

4.2.31 Fixture_equipment_element

A Fixture_equipment_element is a type of Building_element (see 4.2.10) that is a solid, tangible part of a building placed or installed in the building. It is a standalone element that provides a service without complex interdependence on other building elements.

EXAMPLE 21 - Doors, furniture, window shades, hand railing, fireplace mantlepiece, installed shelving, ornamental molding.

The data associated with a Fixture_equipment_element are the following:

— functional_type.

The functional_type specifies a descriptive label that indicates the function of the element. The value is either user defined or predefined.

The predefined value of the functional_type attribute is one of the following:

— ceiling;

— covering element;

— door;

— floor covering;

— furniture;

— wall covering;

— window.

NOTE - See 4.2.31.1 - 4.2.31.7 for the definition of each permissible value for functional_type.

4.2.31.1 ceiling: the upper interior surface of a room.

4.2.31.2 covering element: material that is applied to a surface of a Building_item (see 4.2.12) to protect, insulate, or decorate.

4.2.31.3 door: a movable element that is used to seal an accessway.

4.2.31.4 floor covering: a covering element that is applied to an lower interior surface of a room.

4.2.31.5 furniture: movable articles used in living or working activities that are not implements of production.

4.2.31.6 wall covering: a covering element that is applied to a surface of a wall.

4.2.31.7 window: a framed structure which is spanned with glass and mounted in a window opening; it may be mounted to permit opening and closing.

4.2.32 Gis_position

A Gis_position is the positioning and orientation information necessary for transforming coordinate values between a local coordinate space and the global coordinate space of earth. Transformation procedures depend upon the GIS coordinate system. Each Gis_position object designates the global position and orientation of either a Building_complex (see 4.2.8) or a Site_shape_representation (see 4.2.53).

The data associated with a Gis_position are the following:

— height;

— scale;

— system;

— x_axis_delta_x;

— x_axis_delta_y;

— x_coordinate;

— y_coordinate;

— zone.

4.2.32.1 height

The height specifies the distance above sea level or reference level in the GIS coordinate system.

4.2.32.2 scale

The scale specifies a transformation factor applied to the conversion of point coordinates between a local coordinate system and a GIS coordinate system. The precise application of the transformation will depend on the GIS system.

4.2.32.3 system

The system specifies the identifier of the GIS system being used.

EXAMPLE 22 - Gauss-Krueger, Universal Transverse Mercator (UTM), and State Plane are examples of GIS systems used for global positioning.**[2]**

4.2.32.4 x_axis_delta_x

The x_axis_delta_x specifies the abscissa value of the end point of a vector indicating the positive x axis of GIS coordinate space in the local coordinate system.

4.2.32.5 x_axis_delta_y

The x_axis_delta_y specifies the ordinate value of the end point of a vector indicating the orientation or the positive x axis of GIS coordinate space in the local coordinate system.

EXAMPLE 23 - The GIS coordinate system XY00 has an origin at the intersection of the equator and the Greenwich meridian; the x axis of the coordinate system runs East (positive) and West (negative); the y axis runs North (positive) and South (negative); the positive z axis is up. An x_axis_delta_x of 1.0 and x_axis_delta_y of 1.0 indicates x axis of the GIS coordinate space makes a +45° angle with respect to the x axis of the local coordinate; if the local coordinate space were superimposed on the GIS coordinate space, the positive x axis of the local coordinate system would point in a South-East direction (-45°).

4.2.32.6 x_coordinate

The x_coordinate specifies the distance from the y axis of the coordinate space defined by the GIS system and zone.

4.2.32.7 y_coordinate

The y_coordinate specifies the distance from the x axis of the coordinate space defined by the GIS system and zone.

4.2.32.8 zone

The zone specifies a subdivision of the earth's surface based on the GIS system.

EXAMPLE 24 - The Gauss-Krueger GIS system subdivides the earth into 120 zones that are 3° in longitudinal width. Each zone is identified as 3°, 6°, 9°, etc., from the Greenwich meridian.

4.2.33 Ground_face

A Ground_face is a planar bounded surface that represents a floor space within a building.

4.2.34 Item_assembly

An Item_assembly is an assemblage of Building_item objects (see 4.2.10) or Item_assembly objects that are combined to form a larger object.

NOTES

1 - An Item_assembly is an aggregation of components of a building that is defined at an intermediate level between the individual piece-part elements of a building (e.g., a beam) and a major subdivision of a building (e.g., a building section or level).

2 - The components in an Item_assembly are physically positioned within the spatial coordinate system of the assembly. Contrast this with Item_group (see 4.2.36), which is simply a logical collection of Building_-item objects.

EXAMPLE 25 - Trusses and frames are typical element assemblies for structural elements. A pipeline is a typical example of an element assembly for building services equipment.

The data associated with an Item_assembly are the following:

— approval_information;

— assembly_characterization;

— assembly_class;

— assembly_type;

— components;

— description;

— identifier;

— name.

4.2.34.1 approval_information

The approval_information specifies a collection of approvals of the Item_assembly objects. The approval_information need not be specified for a particular Item_assembly. See 4.3.53 for the application assertion.

4.2.34.2 assembly_characterization

The assembly_characterization specifies a collection of properties of the Item_assembly. The assembly_characterization need not be specified for a particular Item_assembly. See 4.3.57 for the application assertion.

4.2.34.3 assembly_class

The assembly_class specifies a collection of designations; each designation classifies an Item_assembly as a member of some predefined category. The assembly_class need not be specified for a particular Item_assembly. See 4.3.56 for the application assertion.

4.2.34.4 assembly_type

The assembly_type specifies a descriptive label that indicates the form or function of the assembly. The value is either user defined or predefined.

EXAMPLE 26 - Figure 7 illustrates several examples of element assemblies.

The predefined value of the assembly_type attribute is one of the following:

— roof;

— stairway;

— vertical passage enclosure.

NOTE 2 - See 4.2.34.4.1 - 4.2.34.4.3 for the definition of each permissible value for assembly_type.

4.2.34.4.1 roof: the covering or upper part of a building that has a primary function of protecting the building from weather.

4.2.34.4.2 stairway: one or more flights of stairs used to pass from one floor of a building to another. The flights of stairs are often separated by landings and the stairway includes the supporting structure.

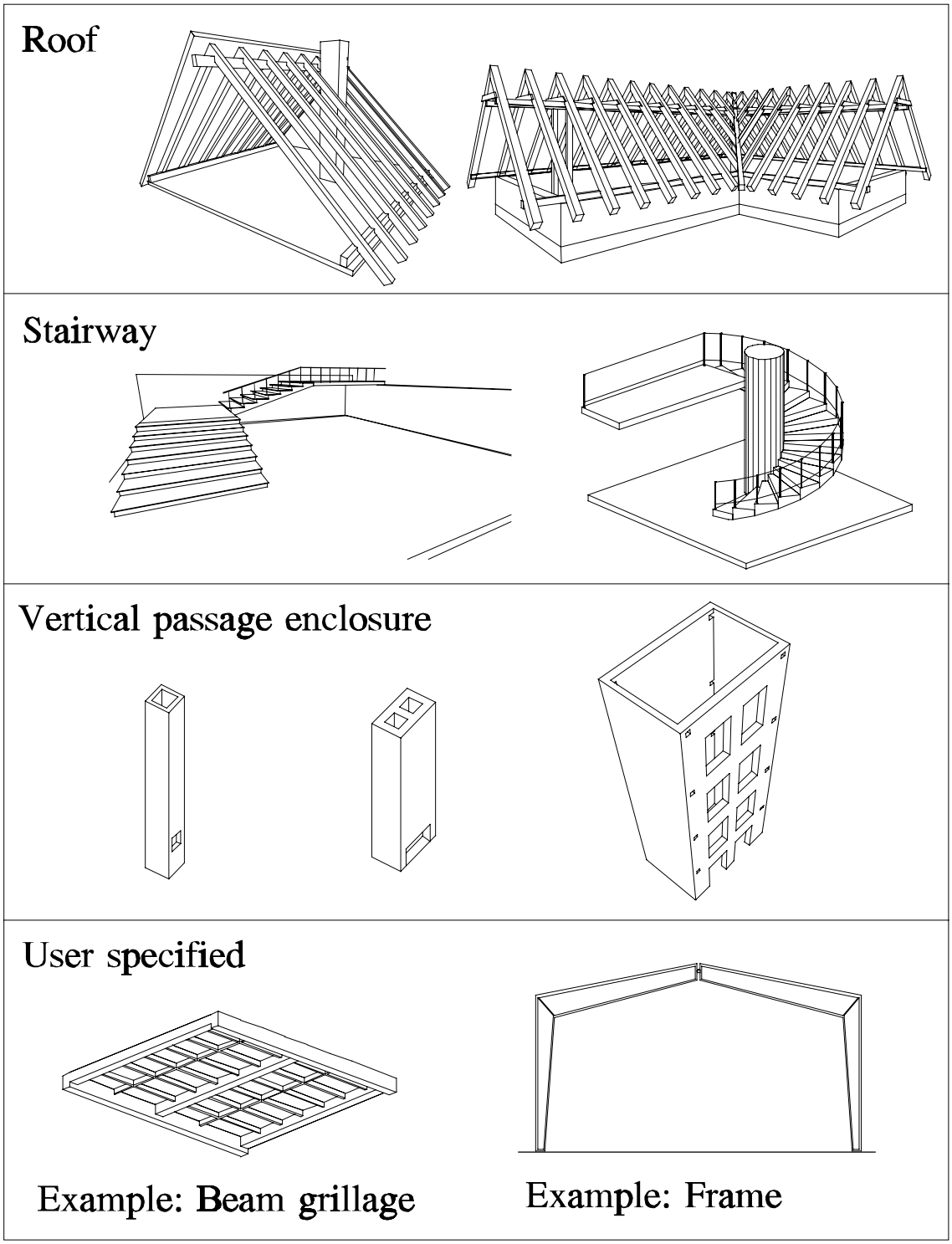


Figure 7 - Kinds of element assemblies

4.2.34.4.3 vertical passage enclosure: the structure that surrounds, frames, or encases a vertical opening through the floors of a building.

4.2.34.5 components

The components specifies the collection of Building_item (see 4.2.12) objects or Item_assembly (see 4.2.34) objects that comprise the Item_assembly. See 4.3.54, 4.3.55 for the application assertions.

4.2.34.6 description

The description specifies a textual explanation of the purpose or function of the Item_assembly.

4.2.34.7 identifier

The identifier specifies a designation that uniquely identifies a particular Item_assembly.

4.2.34.8 name

The name specifies a descriptive, alphanumeric label that is used to refer to the assembly.

4.2.35 Item_classification

An Item_classification is a designation that categorizes a Building_item (see 4.2.12) or a Building_-element_component (see 4.2.11). The classification can be based on national or international standards or a project specific classification system.

NOTE - Organizations involved in a building project should identify and agree to the set of classification standards to be used prior to the initiation of the project.

EXAMPLE 27 - Table 1 shows examples of classifications.

Table 1 - Examples of building element classifications

name	notation	description
DIN_276	462	escalators, moving pavements
AU_CCS	237_36	escalators
EPIC_F	J13	escalators
NL_SFB	662	transports; escalators, conveyors
UK_BCIS_B	5J2	escalators

The data associated with an Item_classification are the following:

— description;

— name;

— notation;

— table.

4.2.35.1 description

The description specifies textual explanation of the classification assigned to an item or component.

4.2.35.2 name

The name specifies an alphanumeric identifier that designates the standard or specification used to classify the item or component.

4.2.35.3 notation

The notation specifies an alphanumeric designation of a class within a table in the standard.

4.2.35.4 table

The table specifies a subsection of the standard identified by name.

4.2.36 Item_group

An Item_group is a logical collection of Building_item objects (see 4.2.12) or Item_group objects.

NOTE - There is no meaning associated with the group other than the fact that it is a group. In particular, there is no physical relationship implied among the members of the group.

The data associated with an Item_group are the following:

— description;

— identifier;

— members;

— name.

4.2.36.1 description

The description specifies a textual explanation of the purpose or function of the Item_group.

4.2.36.2 identifier

The identifier specifies a designation that uniquely identifies a particular Item_group among all Item_-groups.

4.2.36.3 members

The members specifies the collection of Building_item (see 4.2.12) objects or Item_group (see 4.2.36) objects that comprise the Item_group. See 4.3.58, 4.3.59 for the application assertions.

4.2.36.4 name

The name specifies a descriptive, alphanumeric label that is used to refer to the group.

4.2.37 Item_position_in_section

An Item_position_in_section is a combination of a Building_section (see 4.2.16) and positioning information that enables the shape of a Building_item (see 4.2.12) to be positioned within the spatial context of a Building_section. The aggregation of all Item_position_in_section objects with the same Building_section as a context constitutes the items that comprise the Building_section.

The data associated with an Item_position_in_section are the following:

— location;

— positioned_item;

— positioned_within;

— reference_curves.

4.2.37.1 location

The location specifies the positioning and orienting information required to place a Building_item (see 4.2.12) within the spatial context of the Building_section (see 4.2.16). See 4.3.63 for the application assertion.

4.2.37.2 positioned_item

The positioned_item specifies the Building_item (see 4.2.12) that is to be placed within the spatial context of the Building_section (see 4.2.16). See 4.3.60 for the application assertion.

4.2.37.3 positioned_within

The positioned_within specifies the Building_section (see 4.2.16) which establishes a spatial context in which items of the section are positioned. See 4.3.61 for the application assertion.

4.2.37.4 reference_curves

The reference_curves specifies the Simple_curve (see 4.2.52) that serves as a positioning reference for Building_items (see 4.2.12) within a Building_section (see 4.2.16). The reference_curves in a Building_section are always co-planar and lie in the X-Y plane. See 4.3.62 for the application assertion.

EXAMPLE 28 - Figure 8 illustrates examples of reference curves.

4.2.38 Item_proximity_relationship

An Item_proximity_relationship is an association between a Building_item (see 4.2.12) and a group of other Building_item objects that are near it.

The data associated with an Item_proximity_relationship are the following:

— item;

— items_in_proximity;

— relationship_type.

4.2.38.1 item

The item specifies the Building_item (see 4.2.12) near which other Building_item objects are positioned. See 4.3.64 for the application assertion.

4.2.38.2 items_in_proximity

The items_in_proximity specifies the group of Building_item (see 4.2.12) objects that are near the item. See 4.3.64 for the application assertion.

4.2.38.3 relationship_type

The relationship_type specifies a descriptive label of proximity. The value is either user defined or predefined.

The predefined value of the relationship_type attribute is one of the following:

— penetration;

— space to element;

— touch.

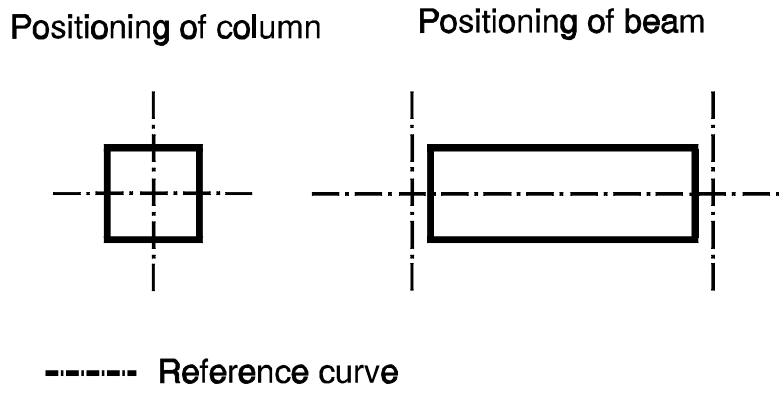


Figure 8 - Examples of reference curves

NOTE - See 4.2.38.3.1 - 4.2.38.3.3 for the definition of each permissible value for relationship_type.

4.2.38.3.1 penetration: the condition in which the items_in_proximity penetrate or "go through" the item.

4.2.38.3.2 space to element: the condition in which the items_in_proximity bound the space.

4.2.38.3.3 touch: the condition in which the items_in_proximity are in contact with the item.

4.2.39 Level_position_in_section

A Level_position_in_section is a combination of a Building_section (see 4.2.16) and positioning information that enables a Building_level (see 4.2.14) to be positioned within the spatial context of a Building_section. The aggregation of all Level_position_in_section objects with the same Building_-section as a context constitutes the levels that comprise the Building_section.

The data associated with a Level_position_in_section are the following:

— location;

— positioned_level;

— positioned_within.

4.2.39.1 location

The location specifies the positioning and orienting information required to place a level within the spatial context of the Building_section (see 4.2.16). See 4.3.67 for the application assertion.

4.2.39.2 positioned_level

The positioned_level specifies the Building_level (see 4.2.14) that is to be placed within the spatial context of the Building_section (see 4.2.16). See 4.3.65 for the application assertion.

4.2.39.3 positioned_within

The positioned_within specifies the Building_section (see 4.2.16) that establishes a spatial context in which levels of the building are positioned. See 4.3.66 for the application assertion.

4.2.40 Negative_component

A Negative_Component is a type of Building_element_component (see 4.2.11) that is a portion of a Building_element (see 4.2.10) and represents a volume of space; it is a volume that is empty space rather than solid material. A Negative_component may be one of the following: a Recess (see 4.2.48) or an Opening (see 4.2.41).

NOTE - The shape of a Negative_component is of concern only where it intersects a Positive_component.

EXAMPLE 29 - A Negative_component in a wall could be a door opening. The relationship between the Negative_component (the door opening) and the Positive_component (the wall) is the same as if the shape of the Negative_component were subtracted using a solid boolean operation (a CSG solid modelling operation) from the shape of the Positive_component.

4.2.41 Opening

An Opening is a type of Negative_component (see 4.2.40) that passes through a Building_element (see 4.2.10) and serves a functional purpose.

EXAMPLE 30 - An opening would be left in a wall or slab for plumbing or air conditioning ducts. Figure 9 illustrates several opening types.

The data associated with an Opening are the following:

— opening_type.

The opening_type specifies a descriptive label that indicates the form or function of the Opening. The value is either user defined or predefined.

The predefined value of the opening_type attribute is one of the following:

— door opening;

— service opening;

— window opening.

NOTE 2 - See 4.2.41.1 - 4.2.41.3 for the definition of each permissible value for opening_type.

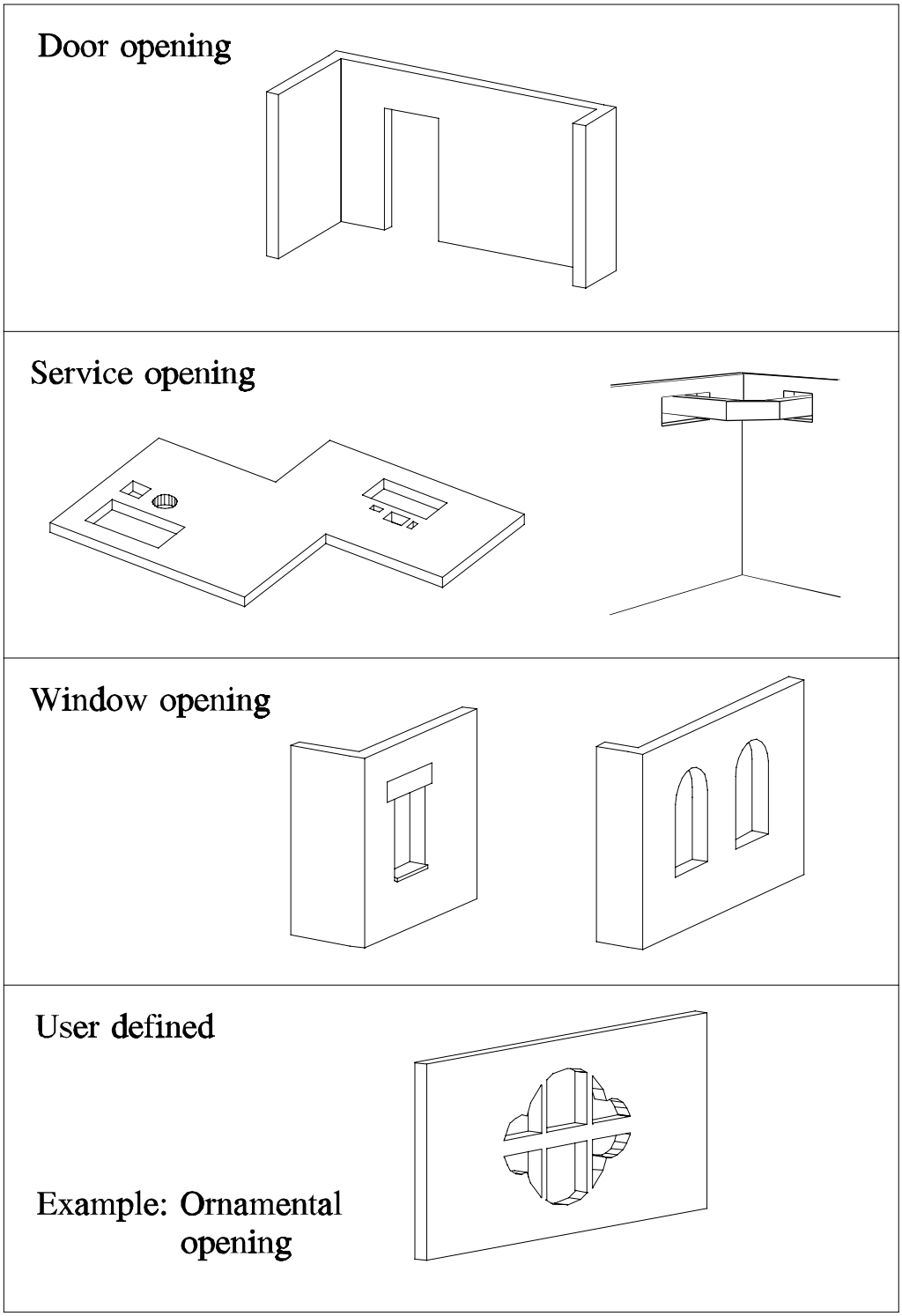


Figure 9 - Kinds of openings

4.2.41.1 door opening: an opening in a wall that is intended to provide access from one side of the wall to the other and may be filled with a door.

4.2.41.2 service opening: an opening in a wall, floor, or roof that is intended to allow service elements to pass through the wall, floor, or roof.

4.2.41.3 window opening: an opening in a wall or roof of a building to admit light and usually to provide visual access through the wall or roof.

4.2.42 Placement

A Placement is a transformation mechanism that positions the geometric definition of a shape within the spatial context of another geometric definition.

4.2.43 Point

A Point is a set of three rational values that specify a location in a three-dimensional cartesian space.

4.2.44 Point_and_line_representation

A Point_and_line_representation is a type of Site_shape_representation (see 4.2.53). A Point_and_-line_representation is a representation of the shape of the surrounding grounds of a building. It consists of a collection of points representing the topography of the site upon which a building is or will be located.

NOTE - Points are commonly gathered through a survey of an actual building site.

The data associated with a Point_and_line_representation are the following:

— survey_points.

The survey_points specifies a collection of cartesian points that comprise the Point_and_line_-representation. The data points are gathered through a surveying mechanism. See 4.3.68 for the application assertion.

4.2.45 Polyline

A Polyline is a curve that consists of individual, straight-line segments.

The data associated with a Polyline are the following:

— path.

The path specifies a collection of points that represent contiguous, straight line elements. See 4.3.69 for the application assertion.

4.2.46 Positive_component

A Positive_component is a type of Building_element_component (see 4.2.11) that represents a solid volume.

EXAMPLE 31 - A Positive_component in a wall could be a window sill. The relationship between the Positive_component representing the sill and the Positive_component representing the wall is the same as if the shape of the positive_component representing the sill were added to the Positive_component representing the wall using a solid boolean operation (a CSG solid modelling operation).

4.2.47 Property

A Property is a descriptively-named, non-geometric characteristic of a Building_item (see 4.2.12), Building_level (see 4.2.14), or Building_element_component (see 4.2.11) that is associated with a value; the name and the value together specify the performance, behavior, or state of the element or component.

The data associated with a Property are the following:

— code_of_measurement;

— formula;

— name;

— property_type;

— value.

4.2.47.1 code_of_measurement

The code_of_measurement specifies the name of a standard or code that is used to calculate the value of a property. The code_of_measurement need not be specified for a particular Property.

4.2.47.2 formula

The formula specifies an equation that was used to calculate the value of the Property. The formula may be dependent upon the code_of_measurement but need not be. The formula need not be specified for a particular Property.

4.2.47.3 name

The name specifies a descriptive label for an aspect or characteristic of a Building_item (see 4.2.12) or Building_element_component (see 4.2.11) that can be represented with an alphanumeric value.

4.2.47.4 property_type

The property_type specifies a descriptive label that indicates the purpose or nature of the property. The value is either user defined or predefined.

NOTE 1 - The property_type is used to classify the property. Contrast property_type with the role that name plays with respect to Property.

The predefined value of the property_type attribute is one of the following:

— material;

— performance;

— physical;

— surface.

NOTE 2 - See 4.2.47.4.1 - 4.2.47.4.4 for the definition of each permissible pre-defined value for property_-type.

4.2.47.4.1 material: a property that specifies or describes the substances from which something is composed.

4.2.47.4.2 performance: a property that specifies or describes the behaviour of something.

4.2.47.4.3 physical: a property that specifies or describes a physical characteristic other than a material or a surface property of something.

4.2.47.4.4 surface: a property that specifies or describes the conditions or preparation of the surface of something.

4.2.47.5 value

The value specifies an alphanumeric symbol that describes or specifies the aspect or characteristic of a Building_item (see 4.2.12) or Building_element_component (see 4.2.11) designated by the name.

4.2.48 Recess

A Recess is a type of Negative_component (see 4.2.40) that does not pass through a Building_element (see 4.2.10).

EXAMPLE 32 - An alcove or niche in a wall are recesses.

4.2.49 Right_circular_cylinder

A Right_circular_cylinder is a three-dimensional, cylindrical, solid primitive with end surfaces that are planar and are perpendicular to the axis. The size and shape of a Right_circular_cylinder is completely described by two real values, which represent the radius and length of the cylinder.

4.2.50 Section_position_in_building

A Section_position_in_building is a combination of a Building (see 4.2.7) and positioning information that enables a Building_section (see 4.2.16) to be positioned within the spatial context of the Building. The aggregation of all Section_position_in_building objects with the same Building as a context constitutes the sections that comprise the Building.

The data associated with a Section_position_in_building are the following:

— location;

— positioned_section;

— positioned_within.

4.2.50.1 location

The location specifies the positioning and orienting information required to place a section within the spatial context of the building. See 4.3.72 for the application assertion.

4.2.50.2 positioned_section

The positioned_section specifies the Building_section that is to be placed within the spatial context of the Building. See 4.3.71 for the application assertion.

4.2.50.3 positioned_within

The positioned_within specifies the Building that establishes a spatial context in which sections of the building are positioned. See 4.3.70 for the application assertion.

4.2.51 Service_element

A Service_element is a type of Building_element (see 4.2.10). A Service_element is a solid, tangible part of a building that is a component in one or more systems installed in a building which provide a service to the building.

NOTE 1 - Building services do not include the structural system of a building, separation walls, or the cladding or glazing of the building.

EXAMPLE 33 - Building services include HVAC, plumbing, and electrical systems. A Service_element may therefore be a duct or pipe, a collection of ducts or pipes, or a piece of equipment like a compressor or pump. Figure 10 illustrates an HVAC system, which is a kind of building services equipment.

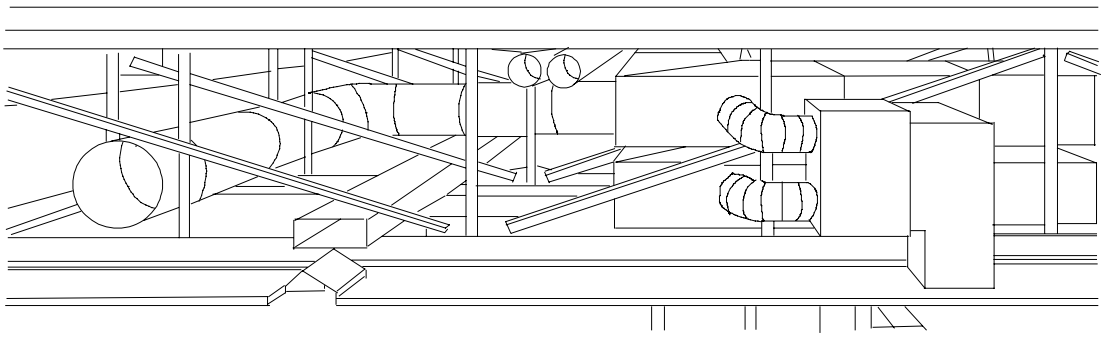


Figure 10 - Services element: HVAC system

The data associated with a Service_element are the following:

— functional_type.

The functional_type specifies a descriptive label that indicates the function of the element. The value is either user defined or predefined.

The predefined value of the functional_type attribute is one of the following:

— electrical system;

— HVAC system;

— plumbing system;

— transport system.

NOTE 2 - See 4.2.51.1 - 4.2.51.4 for the definition of each permissible value for functional_type.

4.2.51.1 electrical system: a network of electrical equipment and conducting wire that provides power or communication throughout a building complex.

4.2.51.2 HVAC system: a network of equipment and ducting that provides ventilation and heated or cooled air throughout a building complex.

4.2.51.3 plumbing system: a network of pipes, fixtures, and other apparatus that provide water, gas, or sewage removal services throughout a building complex.

4.2.51.4 transport system: mechanical apparatuses that moves articles or people from one point in a building complex to another.

4.2.52 Simple_curve

A Simple_curve is a bounded, straight line or circular arc.

4.2.53 Site_shape_representation

A Site_shape_representation is a representation of the shape of the surrounding grounds of a building. A Site_shape_representation is either a Faceted_surface_representation (see 4.2.30) or a Point_and_-line_representation (see 4.2.44).

The data associated with a Site_shape_representation are the following:

— breaklines;

— global_position.

4.2.53.1 breaklines

The breaklines specifies a collection of straight line segments that traverse a Site_shape_representation and establish boundaries that limit the interpolation of the terrain. The points that comprise the breaklines must be a subset of the points that comprise the representation. See 4.3.74 for the application assertion. The breaklines need not be specified for a particular Site_shape_representation.

NOTES

1 - Figure 11 illustrates the role that breaklines play in the interpolation of triangular facets representing the site shape. Figure 11(a) shows survey points of a site shape model without breaklines. Figure 11(b) shows the corresponding site shape model with interpolated triangular facets; there are no constraints concerning the arrangement of the triangular facets. Figure 11(c) shows the same survey points as figure 9(a) with breaklines. The breaklines represent constraints concerning the arrangement of the triangular facets; no triangular facet shall be created that cuts across a breakline. This is illustrated in figures 9(b) and 9(d); the areas within the dashed circles differ with respect to the constraint imposed by the breakline.

2 - Breaklines are used when there are sharp edges in the site, such as a cliff, that restrict the interpolation of the site shape.

4.2.53.2 global_position

The global_position specifies the information necessary for positioning the Site_shape_representation on the earth. See 4.3.73 for the application assertion.

4.2.54 Solid_of_linear_extrusion

A Solid_of_linear_extrusion is a solid geometric model that is formed by sweeping a two-dimensional shape along a linear path through three-dimensional space.

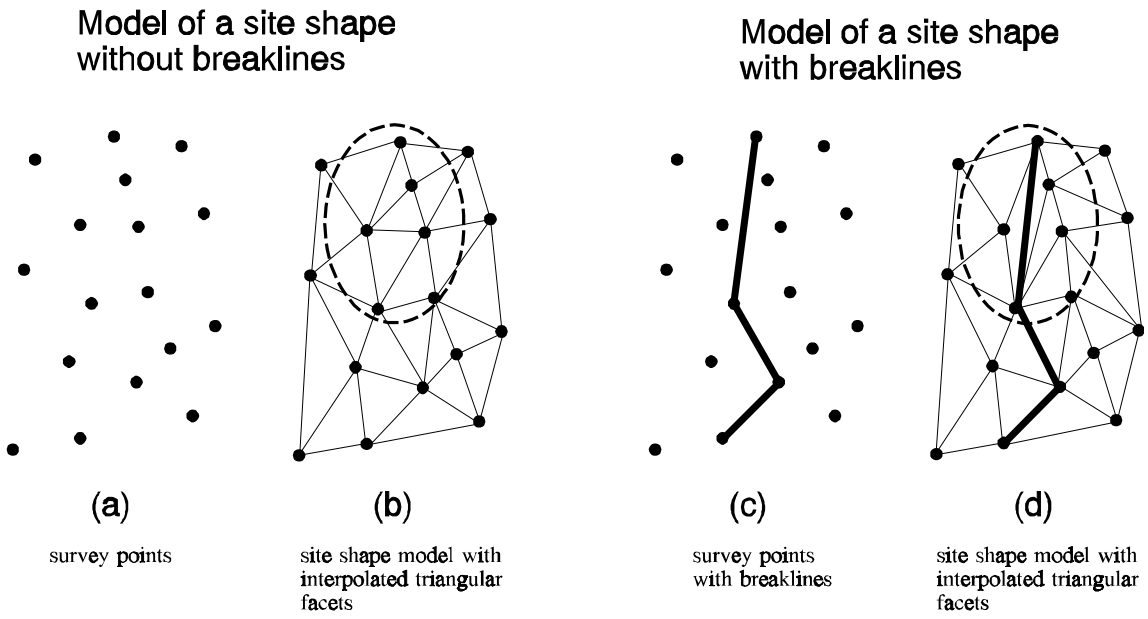


Figure 11 - Breaklines and survey points in site_shape_representation

4.2.55 Solid_of_revolution

A Solid_of_revolution is a solid geometric model that is formed by sweeping a two-dimensional shape through an angle about an axis.

4.2.56 Space

A Space is a type of Building_item (see 4.2.12). A Space is an intangible, volumetric region of a building that represents the portions of a building usable by the building occupants or building service equipment. Spaces are used for planning, describing, and specifying the use of a building. Spaces are created in buildings by the erection of the surrounding walls, floors, and ceilings.

EXAMPLE 34 - Rooms, hallways, and service shafts are examples of spaces. Figure 12 illustrates spaces.

The data associated with a Space are the following:

— space_shapes.

The space_shapes specifies a collection of Advanced_shell objects (see 4.2.4), Elementary_shell objects (see 4.2.24), Faceted_shell objects (see 4.2.29), or Ground_face objects (see 4.2.33) that bound or delimit a volume of space that represents or encloses a Space. See 4.3.75, 4.3.76, 4.3.77, and 4.3.78 for the application assertion.

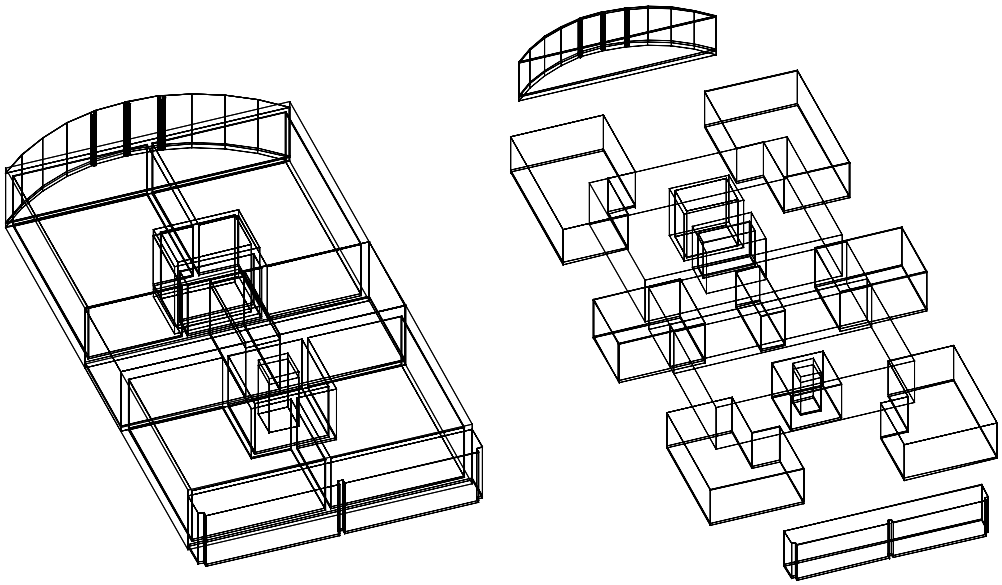


Figure 12 - Spaces

4.2.57 Structure_enclosure_element

A Structure_enclosure_element is a type of Building_element (see 4.2.10) that is an identifiable part of a building that contributes to the basic form or function of a building.

EXAMPLE 35 - Functional roles include characteristics such as load bearing behavior, space separating traits, or protection. These functional roles contribute to the overall function of the building.

The data associated with a Structure_enclosure_element are the following:

— funtional_type;

— load_bearing.

4.2.57.1 functional_type

The functional_type specifies a descriptive label that indicates the form or function of the element. See table 2 and 4.2.57.1.1 to 4.2.57.1.7 for restrictions on the types of geometric primitives that may be used as values for main_component (see 4.2.10) and additions_and_subtractions (see 4.2.10) of Building_element (see 4.2.10) based on functional_type. The value is either user defined or predefined.

NOTE 1 - Figure 13 illustrates several examples of Structure_enclosure_element functional_types.

The predefined value of the functional_type attribute is one of the following:

— beam;

— brace;

— column;

— floor;

— foundation;

— structural wire;

— wall.

NOTE 2 - See 4.2.57.1.1 - 4.2.57.1.7 for the definition of each permissible value for functional_type.

4.2.57.1.1 beam: a long structural element that bears loads perpendicular to its primary dimension, and is supported at different points along that dimension.

4.2.57.1.2 brace: a long structural element that bears compressive or tensile loads parallel to its primary dimension.

4.2.57.1.3 column: a tall, upright structural element that bears loads primarily parallel to its primary dimension.

4.2.57.1.4 floor: a flat, horizontal plate that bears loads primarily perpendicular to its surface. Floors directly bear the life loads of a building. A single floor may consist of several distinct sections.

4.2.57.1.5 foundation: the base structural element of a building upon which the building is constructed. It is usually built below ground and includes footings, cellar walls, and isolated column footings.

4.2.57.1.6 structural wire: a long, commonly cylindrical, structural element that usually bears tensile structural loads along its axis.

4.2.57.1.7 wall: a continuous, flat, usually vertical structural element that bears loads primarily parallel to its surface and encloses, divides, or bounds a space.

4.2.57.2 load_bearing

The load_bearing specifies whether the Structure_enclosure_element bears loads of the building. The value is one of the following: load bearing, load carrying, non-load carrying, or unknown.

4.2.58 Sublevel

A Sublevel is a type of Building_level (see 4.2.14) that is a constituent of another Building_level.

The data associated with a Sublevel are the following:

— belongs_to.

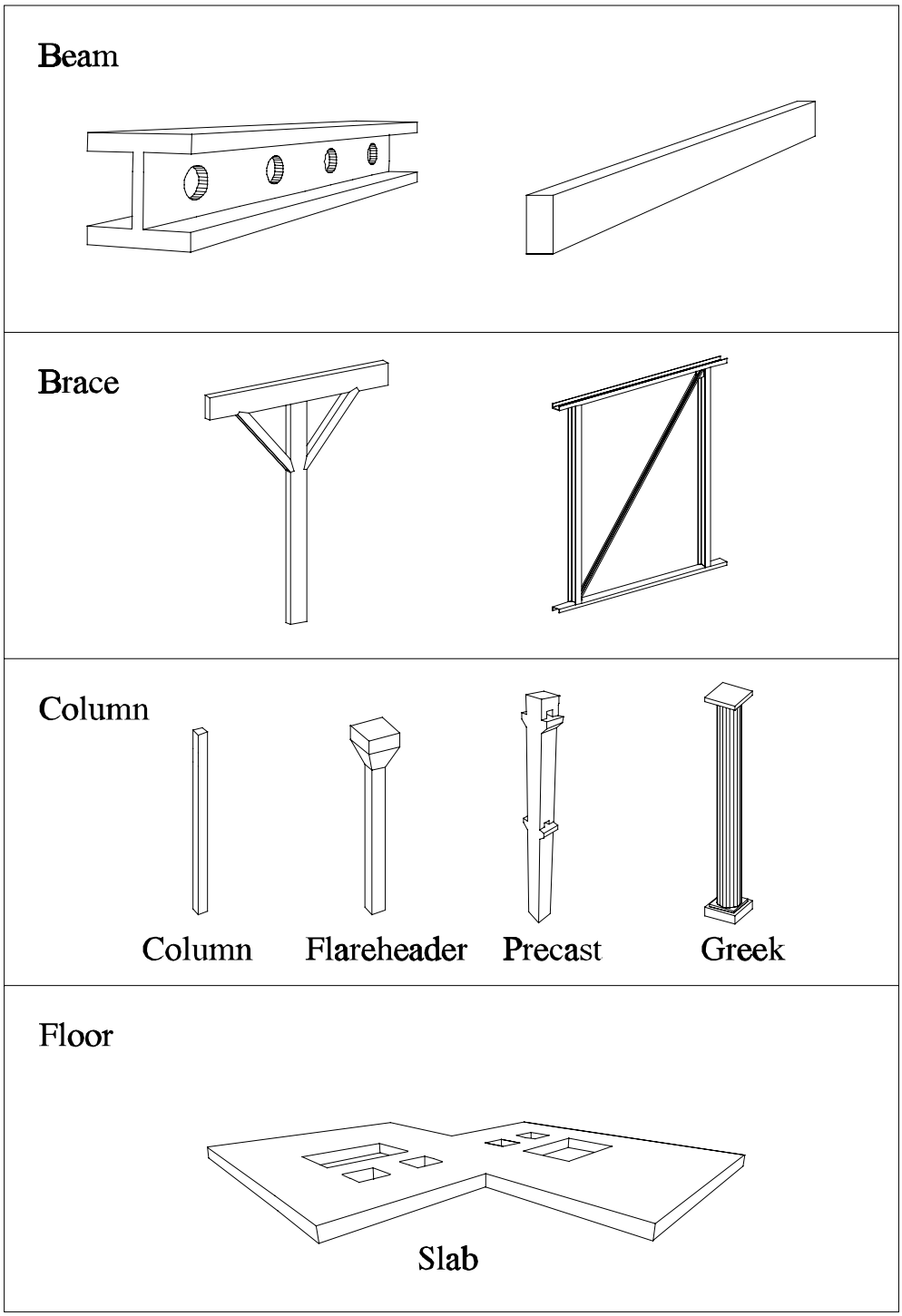


Figure 13 - Kinds of Structure_enclosure_elements

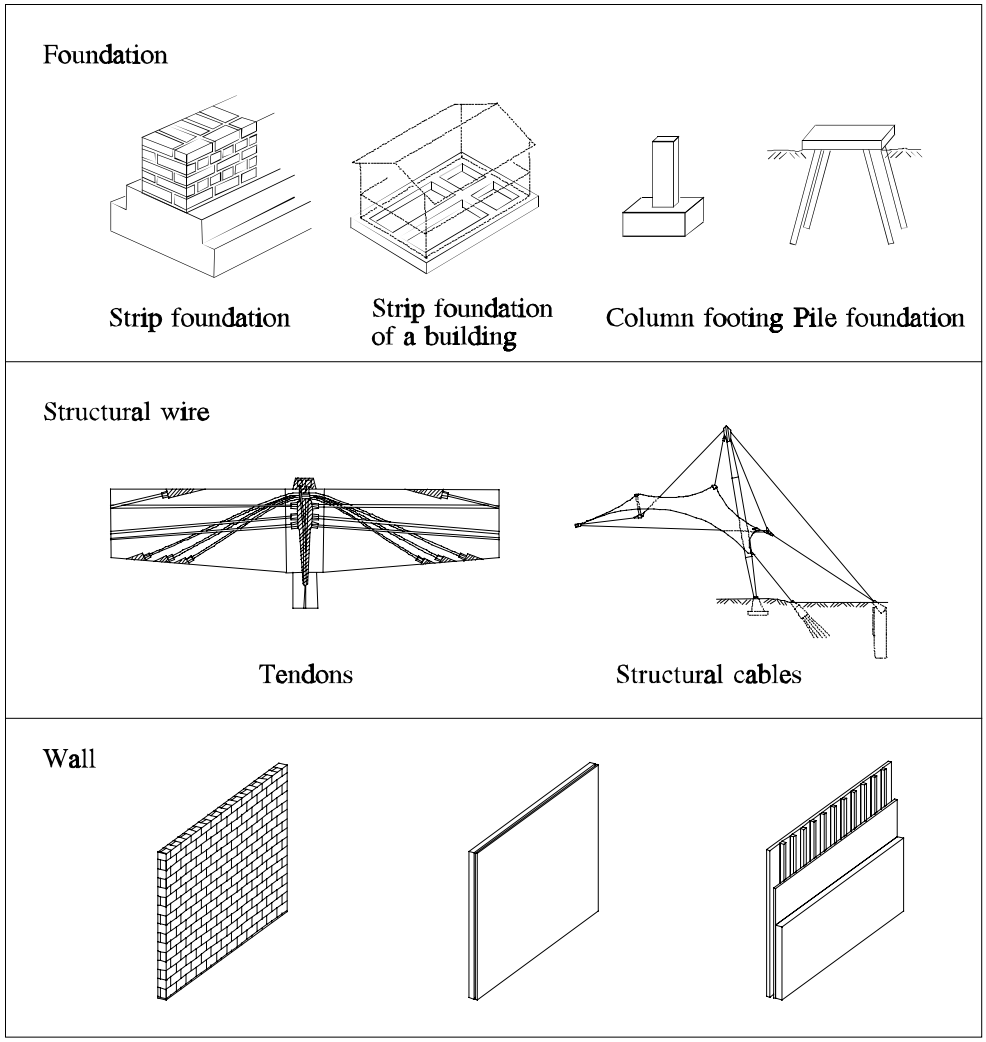


Figure 13 - Kinds of Structure_enclosure_elements (concluded)

The belongs_to specifies the Building_level of which the Sublevel is a part. See 4.3.79 for the application assertion.

4.2.59 Trimmed_sphere

A Trimmed_sphere is the portion of a sphere that remains when the sphere is intersected with a planar surface and one or the other of the resulting sphere sections are removed.

4.2.60 Trimmed_torus

A Trimmed_torus is the portion of a torus that remains when the part of the torus subtended by a plane angle at the primary axis of the torus is removed.

4.2.61 Truncated_cone

A Truncated_cone is a three-dimensional volume with parallel, coaxial, circular faces of differing radii that are connected with a smooth surface. The axis is perpendicular to the circular faces; it is a truncated right circular cone.

4.2.62 Truncated_pyramid

A Truncated_pyramid is a four-sided pyramid that has been truncated by a plane parallel to the base.

Table 2 - Shape component usage for Structure_enclosure_element

Structure_-enclosure_-element		Component	Block	Truncated_pyramid	Truncated_cone	Right_circular_-cylinder	Trimmed_torus	Trimmed_sphere	Solid_of_linear_-extrusion	Solid_of_revolution	Faceted_b_rep	Elementary_b_rep	Advanced_b_rep	Faceted_curve	Elementary_curve	Advanced_curve	Faceted_face_-with_thickness	Elementary_face_-with_thickness	Advanced_face_-with_thickness
Predefined	Beam	main_component	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes							
		additions_and_-subtractions	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes						
	Brace	main_component	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes							
		additions_and_-subtractions	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes						
	Column	main_component	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes							
		additions_and_-subtractions	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes						
	Floor	main_component	Yes						Yes	Yes	Yes								
		additions_and_-subtractions	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes	Yes						
	Founda-tion	main_component	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes	Yes						
		additions_and_-subtractions	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes						
	Structural wire	main_component												Yes	Yes	Yes			
		additions_and_-subtractions	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes			

Table 2 - Shape component usage for Structure_enclosure_element (concluded)

Structure_-enclosure_-element	Component	Block	Truncated_pyramid	Truncated_cone	Right_circular_-cylinder	Trimmed_torus	Trimmed_sphere	Solid_of_linear_-extrusion	Solid_of_revolution	Faceted_b_rep	Elementary_b_rep	Advanced_b_rep	Faceted_curve	Elementary_curve	Advanced_curve	Faceted_face_-with_thickness	Elementary_face_-with_thickness	Advanced_face_-with_thickness
Wall	main_component	Yes						Yes	Yes	Yes	Yes	Yes				Yes	Yes	Yes
	additions_and_-subtractions	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes	Yes				Yes	Yes	Yes
User Defined	main_component	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	additions_and_-subtractions	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

4.3 Application assertions

This subclause specifies the application assertions for the building elements using explicit representation application protocol. Application assertions specify the relationships between application objects, the cardinality of the relationships, and the rules associated required for the integrity and validity of the application objects and UoFs. The application assertions and their definitions are given below.

4.3.1 Building_complex to Gis_position

Each Building_complex has a global position specified by zero or one Gis_position. Each Gis_position specifies the global position of zero or one Building_complex.

4.3.2 Building_complex to Site_shape_representation

Each Building_complex has a surrounding grounds shape defined by zero or one Site_shape_-representation. Each Site_shape_representation defines the site shape of exactly one Building_complex object.

4.3.3 Building_element to Building_element_component

Each Building_element contains zero, one, or many Building_element_component objects that are added to or subtracted from each other to form the shape of the Building_element. Each Building_-element_component defines an addition to or subtraction from the shape of exactly one Building_-element object.

4.3.4 Building_element to Positive_component

Each Building_element has a main_component shape defined by exactly one Positive_component. Each Positive_component is the main component of exactly one Building_element.

4.3.5 Building_element_component to Approval

Each Building_element_component is approved by zero, one, or many Approval objects. Each Approval provides approval for zero, one, or many Building_element_component objects.

4.3.6 Building_element_component to Building_document_reference

Each Building_element_component references zero or one Building_document_reference. Each Building_document_reference provides information for zero, one, or many Building_element_-component objects.

4.3.7 Building_element_component to Building_item_identification

Each Building_element_component is uniquely identified by exactly one Building_item_identification object. Each Building_item_identification object uniquely identifies zero or one Building_element_-component.

4.3.8 Building_element_component to Component_location_in_element

Each Building_element_component has a position specified by exactly one Component_location_in_-element. Each Component_location_in_element specifies the position of zero, one, or many Building_-element_component objects.

4.3.9 Building_element_component to Component_shape

Each Building_element_component has a shape defined by exactly one Component_shape. Each Component_shape defines the shape of exactly one Building_element_component object.

4.3.10 Building_element_component to Item_classification

Each Building_element_component has classifications specified by zero, one, or many Item_-classification objects. Each Item_classification specifies the classification of zero, one, or many Building_element_component objects.

4.3.11 Building_element_component to Property

Each Building_element_component is characterized by zero, one, or many Property objects. Each Property characterizes zero, one, or many Building_element_component objects.

4.3.12 Building_item to Approval

Each Building_item is approved by zero, one, or many Approval objects. Each Approval provides approval for zero, one, or many Building_item objects.

4.3.13 Building_item to Building_document_reference

Each Building_item references zero or one Building_document_reference object. Each Building_-document_reference provides information about zero, one, or many Building_item objects.

4.3.14 Building_item to Building_item_identification

Each Building_item is uniquely identified by exactly one Building_item_identification object. Each Building_item_identification uniquely identifies zero or one Building_item.

4.3.15 Building_item to Building_level

Each Building_item is assigned to zero, one, or many Building_level objects. Each Building_level has assigned to it zero, one, or many Building_item objects.

4.3.16 Building_item to Item_classification

Each Building_item has classifications specified by zero, one, or many Item_classification objects. Each Item_classification specifies the classification of zero, one, or many Building_item objects.

4.3.17 Building_item to Property

Each Building_item is characterized by zero, one, or many Property objects. Each Property characterizes zero, one, or many Building_items objects.

4.3.18 Building_level to Building_item_identification

Each Building_level is uniquely identified by zero or one Building_item_identification. Each Building_item_identification identifies zero or one Building_level.

4.3.19 Building_level to Item_classification

Each Building_level has classifications specified by zero, one, or many Item_classification objects. Each Item_classification specifies the classification of zero, one, or many Building_level objects.

4.3.20 Building_level to Property

Each Building_level is characterized by zero, one, or many Property objects. Each Property characterizes zero, one, or many Building_level objects.

4.3.21 Building_level to Advanced_shell

Each Building_level has space shapes defined by zero or one Advanced_shell object. Each Advanced_-shell defines the space shape of exactly one Building_level object.

4.3.22 Building_level to Elementary_shell

Each Building_level has space shapes defined by zero or one Elementary_shell object. Each Elementary_shell defines the space shape of exactly one Building_level object.

4.3.23 Building_level to Faceted_shell

Each Building_level has space shapes defined by zero or one Faceted_shell object. Each Faceted_shell defines the space shape of exactly one Building_level object.

4.3.24 Building_level to Ground_face

Each Building_level has space shapes defined by zero or one Ground_face object. Each Ground_face defines the space shape of exactly one Building_level object.

4.3.25 Building_position_in_complex to Building

Each Building_position_in_complex has as a positioned building exactly one Building. Each Building is the positioned building in exactly one Building_position_in_complex object.

4.3.26 Building_position_in_complex to Building_complex

Each Building_position_in_complex is positioned within exactly one Building_complex. Each Building_complex contains zero, one, or many Building_position_in_complex objects.

4.3.27 Building_position_in_complex to Placement

Each Building_position_in_complex has a location specified by exactly one Placement. Each Placement specifies the location of zero, one, or many Building_position_in_complex objects.

4.3.28 Building_section to Building_item_identification

Each Building_section is uniquely identified by exactly one Building_item_identification object. Each Building_item_identification object uniquely identifies zero or one Building_section.

4.3.29 Change_request to Approval

Each Change_request is authorized by one, or more Approval objects. Each Approval provides authorization for zero, one, or many Change_request objects.

4.3.30 Change_request to Building_element_component

Each Change_request specifies zero or one Building_element_component as the proposed replacement of the change. Each Building_element_component is specified as the proposed replacement of zero, one, or many Change_request objects.

4.3.31 Change_request to Building_item

Each Change_request specifies zero or one Building_item as the proposed replacement of the change_-request. Each Building_item is specified as the proposed replacement of zero, one, or many Change_-request objects.

4.3.32 Change_request to Building_item_identification

Each Change_request specifies exactly one Building_item_identification which identifies a building item as an unsatisfactory condition. Each Building_item_identification identifies a building item that is specified as unsatisfactory by zero, one, or many Change_request objects.

4.3.33 Component_shape to Component_shape_representation

Each Component_shape is represented by one, or more Component_shape_representation objects. Each Component_shape_representation represents exactly one Component_shape object.

4.3.34 Component_shape_representation to Advanced_b_rep

Each Component_shape_representation contains zero or one Advanced_b_rep objects as an element of the representation. Each Advanced_b_rep is an element of exactly one Component_shape_-representation object.

4.3.35 Component_shape_representation to Advanced_curve

Each Component_shape_representation contains zero or one Advanced_curve objects as an element of the representation. Each Advanced_curve is an element of exactly one Component_shape_-representation object.

4.3.36 Component_shape_representation to Advanced_face_with_thickness

Each Component_shape_representation contains zero or one Advanced_face_with_thickness objects as an element of the representation. Each Advanced_face_with_thickness is an element of exactly one Component_shape_representation object.

4.3.37 Component_shape_representation to Block

Each Component_shape_representation contains zero or one more Block objects as an element of the representation. Each Block is an element of exactly one Component_shape_representation object.

4.3.38 Component_shape_representation to Elementary_b_rep

Each Component_shape_representation contains zero or one Elementary_b_rep objects as an element of the representation. Each Elementary_b_rep is an element of exactly one Component_shape_-representation object.

4.3.39 Component_shape_representation to Elementary_curve

Each Component_shape_representation contains zero or one Elementary_curve objects as an element of the representation. Each Elementary_curve is an element of exactly one Component_shape_-representation object.

4.3.40 Component_shape_representation to Elementary_face_with_thickness

Each Component_shape_representation contains zero or one Elementary_face_with_thickness objects as an element of the representation. Each Elementary_face_with_thickness is an element of exactly one Component_shape_representation object.

4.3.41 Component_shape_representation to Faceted_b_rep

Each Component_shape_representation contains zero or one Faceted_b_rep objects as an element of the representation. Each Faceted_b_rep is an element of exactly one Component_shape_representation object.

4.3.42 Component_shape_representation to Faceted_curve

Each Component_shape_representation contains zero or one Faceted_curve objects as an element of the representation. Each Faceted_curve objects is an element of exactly one Component_shape_-representation object.

4.3.43 Component_shape_representation to Faceted_face_with_thickness

Each Component_shape_representation contains zero or one Faceted_face_with_thickness objects as an element of the representation. Each Faceted_face_with_thickness is an element of exactly one Component_shape_representation object.

4.3.44 Component_shape_representation to Right_circular_cylinder

Each Component_shape_representation contains zero or one Right_circular_cylinder objects as an element of the representation. Each Right_circular_cylinder is an element of exactly one Component_-shape_representation object.

4.3.45 Component_shape_representation to Solid_of_linear_extrusion

Each Component_shape_representation contains zero or one Solid_of_linear_extrusion objects as an element of the representation. Each Solid_of_linear_extrusion is an element of exactly one Component_shape_representation object.

4.3.46 Component_shape_representation to Solid_of_revolution

Each Component_shape_representation contains zero or one Solid_of_revolution objects as an element of the representation. Each Solid_of_revolution is an element of exactly one Component_shape_-representation object.

4.3.47 Component_shape_representation to Trimmed_sphere

Each Component_shape_representation contains zero or one Trimmed_sphere objects as an element of the representation. Each Trimmed_sphere is an element of exactly one Component_shape_-representation object.

4.3.48 Component_shape_representation to Trimmed_torus

Each Component_shape_representation contains zero or one Trimmed_torus objects as an element of the representation. Each Trimmed_torus is an element of exactly one Component_shape_representation object.

4.3.49 Component_shape_representation to Truncated_cone

Each Component_shape_representation contains zero or one Truncated_cone objects as an element of the representation. Each Truncated_cone is an element of exactly one Component_shape_representation object.

4.3.50 Component_shape_representation to Truncated_pyramid

Each Component_shape_representation contains zero or one Truncated_pyramid objects as an element of the representation. Each Truncated_pyramid is an element of exactly one Component_shape_-representation object.

4.3.51 Facet_trigon to Point

Each Facet_trigon has its border defined by three Point objects. Each Point defines the border of zero, one, or many Facet_trigon objects.

4.3.52 Faceted_surface_representation to Facet_trigon

Each Faceted_surface_representation has facets defined by one, or more Facet_trigon objects. Each Facet_trigon defines the facets of zero, one, or many Faceted_surface_representation objects.

4.3.53 Item_assembly to Approval

Each Item_assembly is approved by zero, one, or many Approval objects. Each Approval provides approval for zero, one, or more Item_assembly objects.

4.3.54 Item_assembly to Building_item

Each Item_assembly has components of zero, one, or many Building_item objects. Each Building_item is a component of zero or one Item_assembly objects.

4.3.55 Item_assembly to Item_assembly

Each Item_assembly has components of zero, one, or many Item_assembly objects. Each Item_-assembly is a component of zero or one Item_assembly object.

4.3.56 Item_assembly to Item_classification

Each Item_assembly has classifications specified by zero, one, or many Item_classification objects. Each Item_classification specifies the classification of zero, one, or many Item_assembly objects.

4.3.57 Item_assembly to Property

Each Item_assembly is characterized by zero, one, or many Property objects. Each Property characterizes zero, one, or many Item_assembly objects.

4.3.58 Item_group to Building_item

Each Item_group has members of zero, one, or many Building_item objects. Each Building_item is a member of zero, one, or many Item_group objects.

4.3.59 Item_group to Item_group

Each Item_group has members of zero, one, or many Item_group objects. Each Item_group is a member of zero, one, or many Item_group objects.

4.3.60 Item_position_in_section to Building_item

Each Item_position_in_section has as a positioned element exactly one Building_item. Each Building_-item is the positioned element exactly one Item_position_in_section object.

4.3.61 Item_position_in_section to Building_section

Each Item_position_in_section is positioned within exactly one Building_section. Each Building_-section has positioned in it zero, one, or many Item_position_in_section objects.

4.3.62 Item_position_in_section to Simple_curve

Each Item_position_in_section may have reference lines defined by exactly two or three Elementary_-curve objects. Each Elementary_curve is a reference line for zero, one, or many Item_position_in_-section objects.

4.3.63 Item_position_in_section to Placement

Each Item_position_in_section has a location specified by exactly one Placement. Each Placement specifies the location of zero, one, or many Item_position_in_section objects.

4.3.64 Item_proximity_relationship to Building_item

Each Item_proximity_relationship has an item specified by exactly one Building_item object. Each Building_item may an item in zero, one, or many Item_proximity_relationship objects.

Each Item_proximity_relationship has items in proximity to an item specified by one or more Building_item objects. Each Building_item may in proximity to an item as specified by zero, one, or many Item_proximity_relationship objects.

4.3.65 Level_position_in_section to Building_level

Each Level_position_in_section has as a positioned level in exactly one Building_level. Each Building_level is the positioned level in exactly one Level_position_in_section object.

4.3.66 Level_position_in_section to Building_section

Each Level_position_in_section is positioned within exactly one Building_section. Each Building_-section has positioned in it zero, one, or many Level_position_in_section objects.

4.3.67 Level_position_in_section to Placement

Each Level_position_in_section has a location specified by exactly one Placement. Each Placement specifies the location of zero, one, or many Level_position_in_section objects.

4.3.68 Point_and_line_representation to Point

Each Point_and_line_representation has survey points defined by one or more Point objects. Each Point defines a survey point for zero, one, or many Point_and_line_representation objects.

4.3.69 Polyline to Point

Each Polyline has its path defined by one or more Point objects. Each Point defines the path of zero, one, or many Polyline objects.

4.3.70 Section_position_in_building to Building

Each Section_position_in_building is positioned within exactly one Building. Each Building contains zero, one, or many Section_position_in_building objects.

4.3.71 Section_position_in_building to Building_section

Each Section_position_in_building has as a positioned section exactly one Building_section. Each Building_section is the positioned section in exactly one Section_position_in_building object.

4.3.72 Section_position_in_building to Placement

Each Section_position_in_building has a location specified by exactly one Placement. Each Placement specifies the location of zero, one, or many Section_position_in_building objects.

4.3.73 Site_shape_representation to Gis_position

Each Site_shape_representation has a global position specified by zero or one Gis_position. Each Gis_-position specifies the orientation of zero or one Site_shape_representation.

4.3.74 Site_shape_representation to Polyline

Each Site_shape_representation has breaklines defined by one or more Polyline objects. Each Polyline defines the breaklines of exactly one Site_shape_representation objects.

4.3.75 Space to Advanced_shell

Each Space has space shapes defined by zero or one Advanced_shell object. Each Advanced_shell defines the space shape of exactly one Space object.

4.3.76 Space to Elementary_shell

Each Space has space shapes defined by zero or one Elementary_shell object. Each Elementary_shell defines the space shape of exactly one Space object.

4.3.77 Space to Faceted_shell

Each Space has space shapes defined by zero or one Faceted_shell object. Each Faceted_shell defines the space shape of exactly one Space object.

4.3.78 Space to Ground_face

Each Space has space shapes defined by zero or one Ground_face object. Each Ground_face defines the space shape of exactly one Space object.

4.3.79 Sublevel to Building_level

Each Sublevel belongs to exactly one Building_level. Each Building_level may have zero, one, or more Sublevel objects.

5 Application interpreted model

5.1 Mapping table

This clause contains the mapping table that shows how each UoF and application object of this part of ISO 10303 (see clause 4) maps to one or more AIM constructs (see annex A). The mapping table is organized in five columns.

Column 1) Application element: Name of an application element as it appears in the application object definition in 4.2. Application object names are written in uppercase. Attribute names and assertions are listed after the application object to which they belong and are written in lower case.

Column 2) AIM element: Name of an AIM element as it appears in the AIM (see annex A), the term ‘IDENTICAL MAPPING’, or the term ‘PATH’. AIM entities are written in lower case. Attribute names of AIM entities are referred to as <entity name>.<attribute name>. The mapping of an application element may result in several related AIM elements. Each of these AIM elements requires a line of its own in the table. The term ‘IDENTICAL MAPPING’ indicates that both application objects of an application assertion map to the same AIM element. The term ‘PATH’ indicates that the application assertion maps to the entire reference path.

Column 3) Source: For those AIM elements that are interpreted from the integrated resources or the application interpreted constructs (AICs), 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.

NOTE - Entities or types that are defined within the integrated resources have an AIC as the source reference if the use of the entity or type for the mapping is within the scope of the AIC.

Column 4) Rules: One or more numbers may be given that refer to rules that apply to the current AIM element or reference path. For rules that are derived from relationships between application objects, the same rule is referred to by the mapping entries of all the involved AIM elements. The expanded names of the rules are listed after the table.

Column 5) Reference path: To describe fully the mapping of an application object, it may be necessary to specify a reference path through several related AIM elements. The reference path column documents the role of an AIM element relative to the AIM element in the row succeeding it. Two or more such related AIM elements define the interpretation of the integrated resources that satisfies the requirement specified by the application object. For each AIM element that has been created for use within this part of ISO 10303, a reference path up to its supertype from an integrated resource is specified.

For the expression of reference paths and the relationships between AIM elements the following notational conventions apply:

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

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

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

d) <>: enclosed section constrains at one or more required reference path;

e) ||: enclosed section constrains the supertype entity;

f) ->: attribute references the entity or select type given in the following row;

g) <-: entity or select type is referenced by the attribute in the following row;

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

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

j) =>: entity is a supertype of the entity given in the following row;

k) <=: entity is a subtype of the entity given in the following row;

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

m) \: the reference path expression continues on the next line.

Table 3 - Mapping table for advanced_geometric_representation UoF

Application element	AIM element	Source	Rules	Reference path
ADVANCED_B_REP	advanced_brep_building_shape_-representation	225	3, 6	advanced_brep_building_shape_representation <= shape_representation
ADVANCED_CURVE	advanced_wire_shape_representation	225	3, 6	advanced_wire_shape_representation <= shape_representation {shape_representation <= representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition -> property_definition property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition [product_definition => structure_enclosure_element] [product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category product_category.name = 'structural wire'] }

Table 3 - Mapping table for advanced_geometric_representation UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
ADVANCED_FACE_-WITH_THICKNESS	advanced_face_with_thickness_shape_-representation	225	3, 6, 9	advanced_face_with_thickness_shape_representation <= [solid_model] [shape_representation {shape_representation <= representation representation.items[i] ->> ({representation_item representation_item.name = 'thickness'} representation_item => measure_representation_item <= measure_with_unit => length_measure_with_unit) (representation_item => topological_representation_item => face => face_surface => advanced_face)}]

Table 4 - Mapping table for building_component UoF

Application element	AIM element	Source	Rules	Reference path
BUILDING_ELEMENT_- COMPONENT	[(negative_component) (positive_component)] [(shape_representation) (csg_solid) (boolean_operand)]	225 225 41 42 42		(negative_component <=) (positive_component <=) shape_aspect
description	shape_aspect.description	41		(negative_component <=) (positive_component <=) shape_aspect shape_aspect.description
building_element_- component to approval (as approval_information)	PATH			(negative_component building_design_approval_item = negative_component) (positive_component building_design_approval_item = positive_component) building_design_approval_item <- building_design_approval.items[i] building_design_approval
building_element_- component to building_- document_reference (as document_reference)	PATH			(negative_component building_document_item = negative_component) (positive_component building_document_item = positive_component) building_document_item <- building_document_reference.items[i] building_document_reference
building_element_- component to building_- item_identification (as identifier)	PATH			(negative_component identified_item = negative_component) (positive_component identified_item = positive_component) identified_item <- building_item_identification_assignment.item building_item_identification_assignment

Table 4 - Mapping table for building_component UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_element_- component to component_location_in_- element (as position)	PATH		3, 5	shape_representation <= representation <- representation_map.mapped_representation representation_map <- mapped_item.mapping_source mapped_item mapped_item.mapping_target
building_element_- component to component_shape (as shape)	PATH			(negative_component <=) (positive_component <=) shape_aspect shape_definition = shape_aspect shape_definition characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition
building_element_- component to item_classification (as component_class)	PATH		8	(negative_component building_component_classification_item = negative_component) (positive_component building_component_classification_item = positive_component) building_component_classification_item <- building_component_classification_assignment.items[i] [building_component_classification_assignment] [building_component_classification_assignment <= group_assignment group_assignment.assigned_group ->group => building_component_classification_group building_document_item = building_component_classification_group building_document_item <- building_document_reference.items[i] building_document_reference]

Table 4 - Mapping table for building_component UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_element_-component to property (as component_-characterization)	PATH			(negative_component <=) (positive_component <=) shape_aspect shape_definition = shape_aspect shape_definition characterized_definition = shape_definition characterized definition <- property_definition.definition property_definition
COMPONENT_-LOCATION_IN_-ELEMENT	mapped_item.mapping_target	43	5	{mapped_item.mapping_target mapped_item <= representation_item <- representation.items[i] {representation => shape_representation} representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition ->{property_definition => product_definition_shape} property_definition property_definition.definition ->characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition] product_definition => (building_element) (fixture_equipment_element) (service_element) (structure_enclosure_element)}
COMPONENT_SHAPE	property_definition	41		

Table 4 - Mapping table for building_component UoF (continued)

Application element	AIM element	Source	Rules	Reference path
component_shape to component_shape_-representation (as representations)	PATH		3	property_definition <- property_definition_representation.definition {property_definition_representation => shape_definition_representation} property_definition_representation property_definition_representation.used_representation -> representation => shape_representation => (advanced_brep_building_shape_representation) (advanced_csg_shape_representation) (advanced_face_with_thickness_shape_representation) (advanced_wire_shape_representation) (elementary_csg_shape_representation) (elementary_geometric_shape_representation) (elementary_face_with_thickness_shape_representation) (elementary_wire_shape_representation) (faceted_csg_shape_representation) (faceted_geometric_shape_representation) (faceted_face_with_thickness_shape_representation) (faceted_wire_shape_representation)
NEGATIVE_-COMPONENT	[negative_component] [(shape_representation) (csg_solid) (boolean_operand)]	225 41 42 42		negative_component <= shape_aspect
OPENING	opening	225		opening <= negative_component <= shape_aspect

Table 4 - Mapping table for building_component UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
opening_type	group.name	41	8	opening building_component_classification_item = opening building_component_classification_item <- building_component_classification_assignment.items[i] building_component_classification_assignment <= group_assignment.assigned_group ->{group => building_component_classification_group} group (group.name) (group.name = 'door opening') (group.name = 'service opening') (group.name = 'window opening')
POSITIVE_COMPONENT	[positive_component] [(shape_representation) (csg_solid) (boolean_operand)]	225 41 42 42		positive_component <= shape_aspect
RECESS	recess	225		recess <= negative_component <= shape_aspect

Table 5 - Mapping table for building_composition UoF

Application element	AIM element	Source	Rules	Reference path
BUILDING	building	225		building <= product_definition
address	descriptive_representation_-item.description	45		building <= product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation ->representation representation.items[i] ->{representation_item representation_item.name = 'building address'}representation_item => descriptive_representation_item descriptive_representation_item.description
description	product_definition.description	41		building <= product_definition product_definition.description
name	product.name	41		building <= product_definition product_definition.formation ->product_definition_formation product_definition_formation.of_product ->product product.name

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
owner	(person) (organization)	41 41		(building building_design_person_item = building building_design_person_item <- building_design_person_assignment.items[i] building_design_person_assignment <= person_assignment {person_assignment.role ->person_role person_role.name = ‘owner’} person_assignment.assigned_person ->person) (building building_design_organization_item = building building_design_organization_item <- building_design_organization_assignment.items[i] building_design_organization_assignment <= organization_assignment {organization_assignment.role ->organization_role organization_role.name = ‘owner’} organization_assignment.assigned_organization ->organization)
status	product.description	41		building <= product_definition product_definition.formation ->product_definition_formation product_definition_formation.of_product ->product product.description

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
BUILDING_COMPLEX	building_complex	225	1, 4	building_complex <= product_definition {product_definition product_definition.frame_of_reference ->[product_definition_context product_definition_context.life_cycle_stage = ‘design’] [product_definition_context <= application_context_element application_context_element.frame_of_reference ->application_context application_context.application = ‘building shape composition’]}
description	product.description	41		building_complex <= product_definition product_definition.formation ->product_definition_formation product_definition_formation.of_product ->product product.description
name	product.name	41		building_complex <= product_definition product_definition.formation ->product_definition_formation product_definition_formation.of_product ->product product.name

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
owner	(person) (organization)	41 41		(building_complex building_design_person_item = building_complex building_design_person_item <- building_design_person_assignment.items[i] building_design_person_assignment <= person_assignment {person_assignment.role ->> person_role person_role.name = 'owner'} person_assignment.assigned_person ->> person) (building_complex building_design_organization_item = building_complex building_design_organization_item <- building_design_organization_assignment.items[i] building_design_organization_assignment <= organization_assignment {organization_assignment.role ->> organization_role organization_role.name = 'owner'} organization_assignment.assigned_organization ->> organization)

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_complex to gis_position (as global_position)	PATH		3	building_complex <= product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition {property_definition property_definition.name = 'global position'} property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation {representation representation.name = 'gis position'}
building_complex to site_shape_representation (as surrounding_-grounds_shape)	PATH		3	building_complex <= product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => site <= characterized_object characterized_definition = characterized_object characterized_definition <- property_definition.definition property_definition property_definition.used_representation -> representation => shape_representation => site_representation

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
BUILDING_ITEM	(building_element) (fixture_equipment_element) (service_element) (space_element) (structure_enclosure_element)	225 225 225 225 225		(building_element <=) (fixture_equipment_element <=) (service_element <=) (space_element <=) (structure_enclosure_element <=) product_definition
description	product_definition.description	41		(building_element <=) (fixture_equipment_element <=) (service_element <=) (space_element <=) (structure_enclosure_element <=) product_definition product_definition.description
status	product.description	41		(building_element <=) (fixture_equipment_element <=) (service_element <=) (space_element <=) (structure_enclosure_element <=) product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product product.description

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_item to approval (as approval_information)	PATH			(building_element building_design_approval_item = building_element) (fixture_equipment_element building_design_approval_item = fixture_equipment_element) (service_element building_design_approval_item = service_element) (space_element building_design_approval_item = space_element) (structure_enclosure_element building_design_approval_item = structure_enclosure_element) building_design_approval_item <- building_design_approval.items[i] building_design_approval
building_item to building_document_-reference (as document_reference)	PATH			(building_element building_document_item = building_element) (fixture_equipment_element building_document_item = fixture_equipment_element) (service_element building_document_item = service_element) (space_element building_document_item = space_element) (structure_enclosure_element building_document_item = structure_enclosure_element) building_document_item <- building_document_reference.items[i] building_document_reference

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_item to building_item_- identification (as identifier)	PATH			(building_element identified_item = building_element) (fixture_equipment_element identified_item = fixture_equipment_element) (service_element identified_item = service_element) (space_element identified_item = space_element) (structure_enclosure_element identified_item = structure_enclosure_element) identified_item <- building_item_identification_assignment.item building_item_identification_assignment
building_item to building_level (as level_assignment)	PATH			(building_element <=) (fixture_equipment_element <=) (service_element <=) (space_element <=) (structure_enclosure_element <=) product_definition <- product_definition_relationship.related_product_definition product_definition_relationship product_definition_relationship.relating_product_definition -> product_definition => building_level

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_item to item_classification (as item_class)	PATH			(building_element <=) (fixture_equipment_element <=) (service_element <=) (space_element <=) (structure_enclosure_element <=) product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= [product_category] [product_category building_document_item = product_category building_document_item <- building_document_reference.items[i] building_document_reference]
building_item to property (as item_characterization)	PATH			(building_element <=) (fixture_equipment_element <=) (service_element <=) (space_element <=) (structure_enclosure_element <=) product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition
BUILDING_LEVEL	building_level	225		building_level <= product_definition
name	product_definition.description	41		building_level <= product_definition product_definition.description

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_level to advanced_shell (as space_shapes)	PATH		3	building_level <= product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition {property_definition => product_definition_shape} property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation => shape_representation => advanced_space_boundary_shape_representation
building_level to building_item_-identification (as identifier)	PATH			building_level identified_item = building_level identified_item <- building_item_identification_assignment.item building_item_identification_assignment

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_level to elementary_shell (as space_shapes)	PATH		3	building_level <= product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition {property_definition => product_definition_shape} property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation => shape_representation => elementary_space_boundary_shape_representation
building_level to faceted_shell (as space_shapes)	PATH		3	building_level <= product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition {property_definition => product_definition_shape} property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation => shape_representation => faceted_space_boundary_shape_representation

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_level to ground_face (as space_shapes)	PATH		3	building_level <= product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition {property_definition => product_definition_shape} property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation => shape_representation => ground_face_space_boundary_shape_representation
building_level to item_classification (as level_class)	PATH			building_level <= product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= [product_category] [product_category building_document_item = product_category building_document_item <- building_document_reference.items[i] building_document_reference]

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_level to property (as level_-characterization)	PATH			building_level <= product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition
BUILDING_POSITION_-IN_COMPLEX	[shape_representation] [(mapped_item) (representation_relationship_with_-transformation)]	41 43 43	3, 5	

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_position_in_-complex to building (as positioned_building)	PATH			[shape_representation <= representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition -> property_definition property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition_relationship product_definition_relationship product_definition_relationship.related_product_definition ->] [(mapped_item mapped_item.mapping_source -> representation_map representation_map.mapped_representation ->) (representation_relationship_with_transformation <= representation_relationship representation_relationship.rep_2 ->) representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition -> property_definition property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition] product_definition => building

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_position_in_-complex to building_complex (as positioned_within)	PATH			[shape_representation <= representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition -> property_definition property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition_relationship product_definition_relationship product_definition_relationship.relating_product_definition ->] [(mapped_item <= representation_item <- representation.items[i]) (representation_relationship_with_transformation <= representation_relationship representation_relationship.rep_1 ->) representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition -> property_definition property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition] product_definition => building_complex

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_position_in_-complex to placement (as location)	PATH			[shape_representation <= representation representation.items[i] -> {representation_item representation_item.name = 'location'} representation_item => geometric_representation_item => placement] [(mapped_item mapped_item.mapping_target) (representation_relationship_with_transformation representation_relationship_with_transformation.transformation_operator -> transformation transformation = functionally_defined_transformation functionally_defined_transformation => cartesian_transformation_operator => cartesian_transformation_operator_3d)]
BUILDING_SECTION	building_section	225		building_section <= product_definition
description	product_definition.description	41		building_section <= product_definition product_definition.description
name	product.name	41		building_section <= product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product product.name

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
status	product.description	41		building_section <= product_definition product_definition.formation ->>br>product_definition_formation product_definition_formation.of_product ->>br>product product.description
building_section to building_item_-identification (as identifier)	PATH			building_section identified_item = building_section identified_item <- building_item_identification_assignment.item building_item_identification_assignment
GIS_POSITION	representation	43	3	{representation representation.name = ‘gis position’}
height	[measure_with_unit.value_component] [measure_with_unit.unit_component]	41 41		representation representation.items[i] ->>br>{representation_item representation_item.name = ‘height’} representation_item =>>br>measure_representation_item <= measure_with_unit [measure_with_unit.value_component] [measure_with_unit.unit_component]
scale	[measure_with_unit.value_component] [measure_with_unit.unit_component]	41 41		representation representation.items[i] ->>br>{representation_item representation_item.name = ‘scale’} representation_item =>>br>measure_representation_item <= measure_with_unit [measure_with_unit.value_component] [measure_with_unit.unit_component]

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
system	representation_context.context_type	43		representation representation.context_of_items -> representation_context representation_context.context_type
x_axis_delta_x	[measure_with_unit.value_component] [measure_with_unit.unit_component]	41 41		representation representation.items[i] -> {representation_item representation_item.name = ‘x-axis delta x’} representation_item => measure_representation_item <= measure_with_unit [measure_with_unit.value_component] [measure_with_unit.unit_component]
x_axis_delta_y	[measure_with_unit.value_component] [measure_with_unit.unit_component]	41 41		representation representation.items[i] -> {representation_item representation_item.name = ‘x-axis delta y’} representation_item => measure_representation_item <= measure_with_unit [measure_with_unit.value_component] [measure_with_unit.unit_component]
x_coordinate	[measure_with_unit.value_component] [measure_with_unit.unit_component]	41 41		representation representation.items[i] -> {representation_item representation_item.name = ‘x coordinate’} representation_item => measure_representation_item <= measure_with_unit [measure_with_unit.value_component] [measure_with_unit.unit_component]

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
y_coordinate	[measure_with_unit.value_component] [measure_with_unit.unit_component]	41 41		representation representation.items[i] ->{representation_item representation_item.name = ‘y coordinate’} representation_item => measure_representation_item <= measure_with_unit [measure_with_unit.value_component] [measure_with_unit.unit_component]
zone	descriptive_representation_item.-description	45		representation representation.items[i] ->{representation_item representation_item.name = ‘zone’} representation_item => descriptive_representation_item descriptive_representation_item.description
ITEM_ASSEMBLY	building_element_assembly	225		building_element_assembly <= product_definition
assembly_type	product_category.name	41		building_element_assembly <= product_definition product_definition.formation ->product_definition_formation product_definition_formation.of_product ->product <- product_related_product_category.products[i] product_related_product_category <=product_category (product_category.name) (product_category.name = ‘roof’) (product_category.name = ‘stairway’) (product_category.name = ‘vertical passage enclosure’)
description	product_definition.description	41		building_element_assembly <= product_definition product_definition.description

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
identifier	product_definition.id	41		building_element_assembly <= product_definition product_definition.id
name	product_definition.name	41		building_element_assembly <= product_definition product_definition.name
item_assembly to approval (as approval_information)	PATH			building_element_assembly building_design_approval_item = building_element_assembly building_design_approval_item <- building_design_approval.items[i] building_design_approval
item_assembly to building_item (as components)	PATH			building_element_assembly <= product_definition <- product_definition_relationship.relating_product_definition {product_definition_relationship => product_definition_usage => assembly_component_usage} product_definition_relationship product_definition_relationship.related_product_definition -> product_definition => (building_element) (fixture_equipment_element) (service_element) (space_element) (structure_enclosure_element)
item_assembly to item_assembly (as components)	PATH			building_element_assembly <= product_definition <- product_definition_relationship.relating_product_definition {product_definition_relationship => product_definition_usage => assembly_component_usage} product_definition_relationship product_definition_relationship.related_product_definition -> product_definition => building_element_assembly

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
item_assembly to item_classification (as assembly_class)	PATH			building_element_assembly <= product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= {product_category <- product_category_relationship.sub_category product_category_relationship * product_category_relationship.category -> product_category (product_category.name) (product_category.name = ‘roof’) (product_category.name = ‘stairway’)} [product_category] [product_category building_document_item = product_category building_document_item <- building_document_reference.items[i] building_document_reference]
item_assembly to property (as assembly_-characterization)	PATH			building_element_assembly <= product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition {property_definition property_definition.name = ‘assembly characterization’} property_definition
ITEM_GROUP	building_element_group	225		building_element_group <= product_definition

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
description	product_definition.description	41		building_element_group <= product_definition product_definition.description
identifier	product_definition.id	41		building_element_group <= product_definition product_definition.id
name	product_definition.name	41		building_element_group <= product_definition product_definition.name
item_group to building_item (as members)	PATH			building_element_group <= product_definition <- product_definition_relationship.relating_product_definition {product_definition_relationship product_definition_relationship.name = 'group member'} product_definition_relationship product_definition_relationship.related_product_definition -> product_definition => (building_element) (fixture_equipment_element) (service_element) (space_element) (structure_enclosure_element)
item_group to item_group (as members)	PATH			building_element_group <= product_definition <- product_definition_relationship.relating_product_definition {product_definition_relationship product_definition_relationship.name = 'group member'} product_definition_relationship product_definition_relationship.related_product_definition -> product_definition => building_element_group

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
ITEM_POSITION_IN_-SECTION	[shape_representation] [(mapped_item) (representation_relationship_with_-transformation)]	41 43 43	2, 3, 5	

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
item_position_in_- section to building_item (as positioned_item)	PATH			[shape_representation <= representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition -> property_definition property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition_relationship product_definition_relationship product_definition_relationship.related_product_definition ->] [(mapped_item mapped_item.mapping_source -> representation_map representation_map.mapped_representation ->) (representation_relationship_with_transformation <= representation_relationship representation_relationship.rep_2 ->) representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition -> property_definition property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition] product_definition => (building_element) (fixture_equipment_element) (service_element) (space_element) (structure_enclosure_element)]

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
item_position_in_- section to building_section (as positioned_within)	PATH			[shape_representation <= representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition ->> property_definition property_definition.definition ->> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition_relationship product_definition_relationship product_definition_relationship.relating_product_definition ->] [(mapped_item <= representation_item <- representation.items[i]) (representation_relationship_with_transformation <= representation_relationship representation_relationship.rep_1 ->) representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition ->> property_definition property_definition.definition ->> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition] product_definition => building_section

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
item_position_in_- section to simple_- curve (as reference_curves)	PATH			shape_representation => representation representation.items[i] ->> {representation_item representation_item.name = 'reference curve'} representation_item => geometric_representation_item => curve => bounded_curve => (trimmed_curve {trimmed_curve trimmed_curve.basis_curve ->> curve => conic => circle}) (polyline)
item_position_in_- section to placement (as location)	PATH			[shape_representation <= representation representation.items[i] ->> {representation_item representation_item.name = 'location'} representation_item => geometric_representation_item => placement] [(mapped_item mapped_item.mapping_target) (representation_relationship_with_transformation representation_relationship_with_transformation.transformation_operator ->> transformation transformation = functionally_defined_transformation functionally_defined_transformation => cartesian_transformation_operator => cartesian_transformation_operator_3d)]
ITEM_PROXIMITY_- RELATIONSHIP	product_definition_relationship	41		{product_definition_relationship product_definition_relationship.name = 'item proximity relationship'}

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
relationship_type	product_definition_- relationship.description	41		{(product_definition_relationship.description) (product_definition_relationship.description = ‘penetration’) (product_definition_relationship.description = ‘space to element’) (product_definition_relationship.description = ‘touch’)}
item_proximity_- relationship to building_- item (as item)	PATH			product_definition_relationship product_definition_relationship.related_product_definition ->product_definition => (building_element) (fixture_equipment_element) (service_element) (space_element) (structure_enclosure_element)
item_proximity_- relationship to building_- item (as items_in_proximity)	PATH			product_definition_relationship product_definition_relationship.relating_product_definition ->product_definition => (building_element) (fixture_equipment_element) (service_element) (space_element) (structure_enclosure_element)
LEVEL_POSITION_IN_- SECTION	[shape_representation] [(mapped_item) (representation_relationship_with_- transformation)]	41 43 43	3, 5	

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
level_position_in_- building to building_level (as positioned_level)	PATH			[shape_representation <= representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition ->property_definition property_definition.definition ->characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition_relationship product_definition_relationship product_definition_relationship.related_product_definition ->] [(mapped_item mapped_item.mapping_source ->representation_map representation_map.mapped_representation ->) (representation_relationship_with_transformation <=representation_relationship representation_relationship.rep_2 ->) representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition ->property_definition property_definition.definition ->characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition] product_definition => building_level

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
level_position_in_section to building_section (as positioned_within)	PATH			[shape_representation <= representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition ->) property_definition property_definition.definition ->) characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition_relationship product_definition_relationship product_definition_relationship.relating_product_definition ->] [(mapped_item <= representation_item <- representation.items[i]) (representation_relationship_with_transformation <= representation_relationship representation_relationship.rep_1 ->) representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition ->) property_definition property_definition.definition ->) characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition] product_definition => building_section

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
level_position_in_- building to placement (as location)	PATH			[shape_representation <= representation representation.items[i] ->{representation_item representation_item.name = 'location'} representation_item => geometric_representation_item => placement] [(mapped_item mapped_item.mapping_target) (representation_relationship_with_transformation representation_relationship_with_transformation.transformation_operator ->transformation transformation = functionally_defined_transformation functionally_defined_transformation => cartesian_transformation_operator => cartesian_transformation_operator_3d)]
PLACEMENT	(placement) (mapped_item.mapping_target) (cartesian_transformation_operator_3d)	42 43 42	5	
SECTION_POSITION_- IN_BUILDING	[shape_representation] [(mapped_item) (representation_relationship_with_- transformation)]	41 43 43	3, 5	

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
section_position_in_- building to building (as positioned_within)	PATH			[shape_representation <= representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition ->) property_definition property_definition.definition ->) characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition_relationship product_definition_relationship product_definition_relationship.relating_product_definition ->] [(mapped_item <= representation_item <- representation.items[i]) (representation_relationship_with_transformation <= representation_relationship representation_relationship.rep_1 ->) representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition ->) property_definition property_definition.definition ->) characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition] product_definition => building

Table 5 - Mapping table for building_composition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
section_position_in_- building to building_section (as positioned_section)	PATH			[shape_representation <= representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition ->> property_definition property_definition.definition ->> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition_relationship product_definition_relationship product_definition_relationship.related_product_definition ->] [(mapped_item mapped_item.mapping_source -> representation_map representation_map.mapped_representation ->) (representation_relationship_with_transformation <= representation_relationship representation_relationship.rep_2 ->) representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition ->> property_definition property_definition.definition ->> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition] product_definition => building_section

Table 5 - Mapping table for building_composition UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
section_position_in_- building to placement (as location)	PATH			[shape_representation <= representation representation.items[i] ->{representation_item representation_item.name = 'location'} representation_item => geometric_representation_item => placement] [(mapped_item mapped_item.mapping_target) (representation_relationship_with_transformation representation_relationship_with_transformation.transformation_operator ->transformation transformation = functionally_defined_transformation functionally_defined_transformation => cartesian_transformation_operator => cartesian_transformation_operator_3d)]
SIMPLE_CURVE	(trimmed_curve) (polyline)	42 42		({trimmed_curve trimmed_curve.basis_curve ->curve => conic => circle})
SUBLEVEL	building_level	225		building_level <= product_definition
sublevel to building_level (as belongs_to)	PATH			building_level <= product_definition <- product_definition_relationship.related_product_definition product_definition_relationship {product_definition_relationship.name = 'sublevel'} product_definition_relationship.relating_product_definition ->product_definition => building_level

Table 6 - Mapping table for building_items UoF

Application element	AIM element	Source	Rules	Reference path
BUILDING_ELEMENT	[(building_element) (fixture_equipment_element) (service_element) (structure_enclosure_element)] [(shape_representation) (csg_solid)]	225 225 225 225 41 42		(building_element <=) (fixture_equipment_element <=) (service_element <=) (structure_enclosure_element <=) product_definition

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_element to building_element_-component (as additions_and_-subtractions)	PATH			[(building_element <=) (fixture_equipment_element <=) (service_element <=) (structure_enclosure_element <=) product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape shape_aspect => (negative_component) (positive_component)] [(shape_representation <= representation representation.items[i] ->) representation_item => mapped_item mapped_item.mapping_source -> representation_map representation_map.mapped_representation -> representation => shape_representation) (csg_solid csg_solid.tree_root_expression -> csg_select csg_select = boolean_result boolean_result (boolean_result.first_operand ->) (boolean_result.second_operand ->) boolean_operand)]

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_element to positive_component (as main_component) #1: There are additions and subtractions. #2: There are no additions and subtractions.	#1: ([PATH] [PATH]) #2: ([PATH] [IDENTICAL MAPPING])			#1, #2: [(building_element <=) (fixture_equipment_element <=) (service_element <=) (structure_enclosure_element <=) product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape <- shape_aspect.of_shape {shape_aspect shape_aspect.description = 'main component'} shape_aspect => positive_component] #1: [csg_solid csg_solid.tree_root_expression -> csg_select csg_select = boolean_result boolean_result boolean_result.first_operand -> boolean_operand]
BUILDING_ITEM	(building_element) (fixture_equipment_element) (service_element) (space_element) (structure_enclosure_element)	225 225 225 225 225		(building_element <=) (fixture_equipment_element <=) (service_element <=) (space_element <=) (structure_enclosure_element <=) product_definition

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
description	product_definition.description	41		(building_element <=) (fixture_equipment_element <=) (service_element <=) (space_element <=) (structure_enclosure_element <=) product_definition product_definition.description
status	product.description	41		(building_element <=) (fixture_equipment_element <=) (service_element <=) (space_element <=) (structure_enclosure_element <=) product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product product.description
building_item to approval (as approval_information)	PATH			(building_element building_design_approval_item = building_element) (fixture_equipment_element building_design_approval_item = fixture_equipment_element) (service_element building_design_approval_item = service_element) (space_element building_design_approval_item = space_element) (structure_enclosure_element building_design_approval_item = structure_enclosure_element) building_design_approval_item <- building_design_approval.items[i] building_design_approval

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_item to building_document_-reference (as document_reference)	PATH			(building_element building_document_item = building_element) (fixture_equipment_element building_document_item = fixture_equipment_element) (service_element building_document_item = service_element) (space_element building_document_item = space_element) (structure_enclosure_element building_document_item = structure_enclosure_element) building_document_item <- building_document_reference.items[i] building_document_reference
building_item to building_item_-identification (as identifier)	PATH			(building_element identified_item = building_element) (fixture_equipment_element identified_item = fixture_equipment_element) (service_element identified_item = service_element) (space_element identified_item = space_element) (structure_enclosure_element identified_item = structure_enclosure_element) identified_item <- building_item_identification_assignment.item building_item_identification_assignment

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_item to building_level (as level_assignment)	PATH			(building_element <=) (fixture_equipment_element <=) (service_element <=) (space_element <=) (structure_enclosure_element <=) product_definition <- product_definition_relationship.related_product_definition product_definition_relationship product_definition_relationship.relating_product_definition -> product_definition => building_level
building_item to item_classification (as item_class)	PATH			(building_element <=) (fixture_equipment_element <=) (service_element <=) (space_element <=) (structure_enclosure_element <=) product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= [product_category] [product_category building_document_item = product_category building_document_item <- building_document_reference.items[i] building_document_reference]

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
building_item to property (as item_characterization)	PATH			(building_element <=) (fixture_equipment_element <=) (service_element <=) (space_element <=) (structure_enclosure_element <=) product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition
COMPONENT_SHAPE	property_definition	41		
component_shape to component_shape_-representation (as representations)	PATH		3	property_definition <- property_definition_representation.definition {property_definition_representation => shape_definition_representation} property_definition_representation property_definition_representation.used_representation -> representation => shape_representation => (advanced_brep_building_shape_representation) (advanced_csg_shape_representation) (advanced_face_with_thickness_shape_representation) (advanced_wire_shape_representation) (elementary_csg_shape_representation) (elementary_geometric_shape_representation) (elementary_face_with_thickness_shape_representation) (elementary_wire_shape_representation) (faceted_csg_shape_representation) (faceted_geometric_shape_representation) (faceted_face_with_thickness_shape_representation) (faceted_wire_shape_representation)

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
COMPONENT_SHAPE_- REPRESENTATION	(advanced_brep_building_shape_- representation) (advanced_csg_shape_representation) (advanced_face_with_thickness_shape_- representation) (advanced_wire_shape_representation) (elementary_csg_shape_representation) (elementary_geometric_shape_- representation) (elementary_face_with_thickness_- shape_representation) (elementary_wire_shape_representation) (faceted_csg_shape_representation) (faceted_geometric_shape_- representation) (faceted_face_with_thickness_shape_- representation) (faceted_wire_shape_representation)	225 225 225 225 225 225 225 225 225 225 225 225 225	3, 6, 9	(advanced_brep_building_shape_representation <=) (advanced_csg_shape_representation <=) (advanced_wire_shape_representation <=) (elementary_csg_shape_representation <=) (elementary_geometric_shape_representation <=) (elementary_wire_shape_representation <=) (faceted_csg_shape_representation <=) (faceted_geometric_shape_representation <=) (faceted_wire_shape_representation <=) shape_representation (advanced_face_with_thickness_shape_representation <=) (elementary_face_with_thickness_shape_representation <=) (faceted_face_with_thickness_shape_representation <=) [shape_representation] [solid_model]

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
representation_type	representation.name	43		(advanced_brep_building_shape_representation <=) (advanced_csg_shape_representation <=) (advanced_face_with_thickness_shape_representation <=) (advanced_wire_shape_representation <=) (elementary_csg_shape_representation <=) (elementary_geometric_shape_representation <=) (elementary_face_with_thickness_shape_representation <=) (elementary_wire_shape_representation <=) (faceted_csg_shape_representation <=) (faceted_geometric_shape_representation <=) (faceted_face_with_thickness_shape_representation <=) (faceted_wire_shape_representation <=) shape_representation <= representation (representation.name) (representation.name = ‘detail’) (representation.name = ‘envelope’) (representation.name = ‘outline’)
component_shape_-representation to advanced_b_rep (as representation_-element)	IDENTICAL MAPPING			
component_shape_-representation to advanced_curve (as representation_-element)	IDENTICAL MAPPING			

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
component_shape_- representation to advanced_face_- with_thickness (as representation_- element)	(IDENTICAL MAPPING) (PATH)			advanced_csg_shape_representation <= shape_representation <= representation representation.items[i] ->> representation_item => geometric_representation_item => solid_model => csg_solid csg_solid.tree_root_expression ->> csg_select csg_select = boolean_result boolean_result {boolean_result *} (boolean_result.first_operand ->) (boolean_result.second_operand ->) boolean_operand boolean_operand = solid_model solid_model => advanced_face_with_thickness_shape_representation

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
component_shape_- representation to block (as representation_- element)	PATH			faceted_csg_shape_representation <= shape_representation <= representation representation.items[i] ->> representation_item => geometric_representation_item => solid_model => csg_solid csg_solid.tree_root_expression ->> csg_select (csg_select = csg_primitive) (csg_select = boolean_result boolean_result {boolean_result *} (boolean_result.first_operand ->) (boolean_result.second_operand ->) boolean_operand boolean_operand = csg_primitive) csg_primitive csg_primitive = block block
component_shape_- representation to elementary_b_rep (as representation_- element)	IDENTICAL MAPPING			
component_shape_- representation to elementary_curve (as representation_- element)	IDENTICAL MAPPING			

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
component_shape_- representation to elementary_face_with_- thickness (as representation_- element)	(IDENTICAL MAPPING) (PATH)			elementary_csg_shape_representation <= shape_representation <= representation representation.items[i] ->> representation_item => geometric_representation_item => solid_model => csg_solid csg_solid.tree_root_expression ->> csg_select csg_select = boolean_result boolean_result {boolean_result *} (boolean_result.first_operand ->) (boolean_result.second_operand ->) boolean_operand boolean_operand = solid_model solid_model => elementary_face_with_thickness_shape_representation
component_shape_- representation to faceted_b_rep (as representation_- element)	IDENTICAL MAPPING			
component_shape_- representation to faceted_curve (as representation_- element)	IDENTICAL MAPPING			

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
component_shape_- representation to faceted_face_- with_thickness (as representation_- element)	(IDENTICAL MAPPING) (PATH)			faceted_csg_shape_representation <= shape_representation <= representation representation.items[i] ->> representation_item => geometric_representation_item => solid_model => csg_solid csg_solid.tree_root_expression ->> csg_select csg_select = boolean_result boolean_result {boolean_result *} (boolean_result.first_operand ->) (boolean_result.second_operand ->) boolean_operand boolean_operand = solid_model solid_model => faceted_face_with_thickness_shape_representation

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
component_shape_- representation to right_circular_cylinder (as representation_- element)	PATH			(advanced_csg_shape_representation <=) (elementary_csg_shape_representation <=) shape_representation <= representation representation.items[i] ->> representation_item => geometric_representation_item => solid_model => csg_solid csg_solid.tree_root_expression ->> csg_select (csg_select = csg_primitive) (csg_select = boolean_result boolean_result {boolean_result *} (boolean_result.first_operand ->) (boolean_result.second_operand ->) boolean_operand boolean_operand = csg_primitive) csg_primitive csg_primitive = right_circular_cylinder right_circular_cylinder

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
component_shape_- representation to solid_of_linear_extrusion (as representation_- element)	PATH		9	(advanced_csg_shape_representation <=) (elementary_csg_shape_representation <=) shape_representation <= representation representation.items[i] -> representation_item => geometric_representation_item => (solid_model => csg_solid csg_solid.tree_root_expression -> csg_select (csg_select = csg_primitive) (csg_select = boolean_result boolean_result {boolean_result *} (boolean_result.first_operand ->) (boolean_result.second_operand ->) boolean_operand boolean_operand = solid_model solid_model =>) (solid_model =>) swept_area_solid => extruded_area_solid

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
component_shape_- representation to solid_of_revolution (as representation_- element)	PATH		9	(advanced_csg_shape_representation <=) (elementary_csg_shape_representation <=) shape_representation <= representation representation.items[i] ->) representation_item => geometric_representation_item => (solid_model => csg_solid csg_solid.tree_root_expression -> csg_select (csg_select = csg_primitive) (csg_select = boolean_result boolean_result {boolean_result *} (boolean_result.first_operand ->) (boolean_result.second_operand ->) boolean_operand boolean_operand = solid_model solid_model =>) (solid_model =>) swept_area_solid => revolved_area_solid

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
component_shape_- representation to trimmed_sphere (as representation_- element)	PATH			(advanced_csg_shape_representation <=) (elementary_csg_shape_representation <=) shape_representation <= representation representation.items[i] ->> representation_item => geometric_representation_item => solid_model => csg_solid csg_solid.tree_root_expression ->> csg_select csg_select = boolean_result boolean_result {boolean_result *} {boolean_result.first_operand ->> boolean_operand boolean_operand = csg_primitive csg_primitive csg_primitive = sphere}

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
component_shape_- representation to trimmed_torus (as representation_- element)	PATH			(advanced_csg_shape_representation <=) (elementary_csg_shape_representation <=) shape_representation <= representation representation.items[i] ->> representation_item => geometric_representation_item => solid_model => csg_solid csg_solid.tree_root_expression ->> csg_select csg_select = boolean_result boolean_result {boolean_result *} {boolean_result.first_operand ->> boolean_operand boolean_operand = csg_primitive csg_primitive csg_primitive = torus}

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
component_shape_- representation to truncated_cone (as representation_- element)	PATH			(advanced_csg_shape_representation <=) (elementary_csg_shape_representation <=) shape_representation <= representation representation.items[i] ->> representation_item => geometric_representation_item => solid_model => csg_solid csg_solid.tree_root_expression ->> csg_select (csg_select = csg_primitive) (csg_select = boolean_result boolean_result {boolean_result *} (boolean_result.first_operand ->) (boolean_result.second_operand ->) boolean_operand boolean_operand = csg_primitive) csg_primitive csg_primitive = right_circular_cone right_circular_cone

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
component_shape_- representation to truncated_pyramid (as representation_- element)	PATH			(advanced_csg_shape_representation <=) (elementary_csg_shape_representation <=) (faceted_csg_shape_representation <=) shape_representation <= representation representation.items[i] ->> representation_item => geometric_representation_item => solid_model => csg_solid csg_solid.tree_root_expression ->> csg_select (csg_select = boolean_result) (csg_select = boolean_result boolean_result {boolean_result *} (boolean_result.first_operand ->) (boolean_result.second_operand ->) boolean_operand boolean_operand = boolean_result) boolean_result => truncated_pyramid
FIXTURE_EQUIPMENT_- ELEMENT	fixture_equipment_element	225		fixture_equipment_element <= product_definition

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
functional_type	product_category.name	41		fixture_equipment_element <= product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category (product_category.name) (product_category.name = ‘ceiling’) (product_category.name = ‘covering element’) (product_category.name = ‘door’) (product_category.name = ‘floor covering’) (product_category.name = ‘furniture’) (product_category.name = ‘wall covering’) (product_category.name = ‘window’)
SERVICE_ELEMENT	service_element	225		service_element <= product_definition
functional_type	product_category.name	41		service_element <= product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category product_category.name {(product_category.name) (product_category.name = ‘electrical system’) (product_category.name = ‘HVAC system’) (product_category.name = ‘plumbing system’) (product_category.name = ‘transport system’)}

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
SPACE	space_element	225		space_element <= product_definition
space to advanced_shell (as space_shapes)	PATH		3	space_element <= product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition {property_definition => product_definition_shape} property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation ->> representation => shape_representation => advanced_space_boundary_shape_representation
space to elementary_shell (as space_shapes)	PATH		3	space_element <= product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition {property_definition => product_definition_shape} property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation ->> representation => shape_representation => elementary_space_boundary_shape_representation

Table 6 - Mapping table for building_items UoF (continued)

Application element	AIM element	Source	Rules	Reference path
space to faceted_shell (as space_shapes)	PATH		3	space_element <= product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition {property_definition => product_definition_shape} property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation => shape_representation => faceted_space_boundary_shape_representation
space to ground_face (as space_shapes)	PATH		3	space_element <= product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition {property_definition => product_definition_shape} property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation => shape_representation => ground_face_space_boundary_shape_representation
STRUCTURE_-ENCLOSURE_ELEMENT	structure_enclosure_element	225		structure_enclosure_element <= product_definition

Table 6 - Mapping table for building_items UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
functional_type	product_category.name	41		structure_enclosure_element <= product_definition product_definition.formation ->> product_definition_formation product_definition_formation.of_product ->> product <- product_related_product_category.products[i] product_related_product_category <= product_category (product_category.name) (product_category.name = ‘beam’) (product_category.name = ‘brace’) (product_category.name = ‘column’) (product_category.name = ‘floor’) (product_category.name = ‘foundation’) (product_category.name = ‘structural wire’) (product_category.name = ‘wall’)
load_bearing	property_definition	41		structure_enclosure_element <= product_definition characterized_product_definition = product_definition characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition {property_definition [property_definition.name = ‘load bearing’] [(property_definition.description = ‘load carrying’) (property_definition.description = ‘non-load carrying’) (property_definition.description = ‘unknown’)]}

Table 7 - Mapping table for design_administration UoF

Application element	AIM element	Source	Rules	Reference path
APPROVAL	building_design_approval	225		building_design_approval <= approval_assignment
approver	approval_person_organization	41		building_design_approval <= approval_assignment approval_assignment.assigned_approval -> approval <- approval_person_organization.authorized_approval approval_person_organization {approval_person_organization.role -> approval_role approval_role.role = 'approver'}
date	approval_date_time	41		building_design_approval <= approval_assignment approval_assignment.assigned_approval -> approval <- approval_date_time.dated_approval approval_date_time {approval_date_time.date_time -> date_time_select date_time_select = date}
purpose	approval.level	41		building_design_approval <= approval_assignment approval_assignment.assigned_approval -> approval approval.level
status	approval_status	41		building_design_approval <= approval_assignment approval_assignment.assigned_approval -> approval approval.status -> approval_status

Table 7 - Mapping table for design_administration UoF (continued)

Application element	AIM element	Source	Rules	Reference path
BUILDING_- DOCUMENT_- REFERENCE	building_document_reference	225		building_document_reference <= document_reference
document_type	document_type.product_data_type	41		building_document_reference <= document_reference document_reference.assigned_document -> document document.kind -> document_type document_type.product_data_type
identifier	document.id	41		building_document_reference <= document_reference document_reference.assigned_document -> document document.id
item_in_document	document_usage_constraint. subject_element	41		building_document_reference <= document_reference document_reference.assigned_document -> document <- document_usage_constraint.source document_usage_constraint document_usage_constraint.subject_element
BUILDING_ITEM_- IDENTIFICATION	building_item_identification_assignment	225		building_item_identification_assignment <= name_assignment
item_identifier	name_assignment.assigned_name	41		building_item_identification_assignment <= name_assignment name_assignment.assigned_name

Table 7 - Mapping table for design_administration UoF (continued)

Application element	AIM element	Source	Rules	Reference path
administrator	(person) (organization)	41 41		building_item_identification_assignment (building_design_person_item = building_item_identification_assignment building_design_person_item <- building_design_person_assignment.items[i] building_design_person_assignment <= person_assignment {person_assignment.role ->>br>person_role person_role.name = ‘building item identifier administrator’} person_assignment.assigned_person ->>br>person) (building_design_organization_item = building_item_identification_assignment building_design_organization_item <- building_design_organization_assignment.items[i] building_design_organization_assignment <= organization_assignment {organization_assignment.role ->>br>organization_role organization_role.name = ‘building item identifier administrator’} organization_assignment.assigned_organization ->>br>organization)
project	organizational_project	41		building_item_identification_assignment building_design_organization_item = building_item_identification_assignment building_design_organization_item <- building_design_organization_assignment.items[i] building_design_organization_assignment <= organization_assignment {organization_assignment.role ->>br>organization_role organization_role.name = ‘building item identifier responsible project’} organization_assignment.assigned_organization ->>br>organization <- organizational_project.responsible_organizations[i] organizational_project

Table 7 - Mapping table for design_administration UoF (continued)

Application element	AIM element	Source	Rules	Reference path
CHANGE_REQUEST	change	225		change <= action_assignment
description	versioned_action_request.description	41		change <= action_assignment action_assignment.assigned_action ->>br>action action.chosen_method ->>br>action_method <- action_request_solution.method action_request_solution action_request_solution.request ->>br>versioned_action_request versioned_action_request.description
reason	versioned_action_request.purpose	41		change <= action_assignment action_assignment.assigned_action ->>br>action action.chosen_method ->>br>action_method <- action_request_solution.method action_request_solution action_request_solution.request ->>br>versioned_action_request versioned_action_request.purpose

Table 7 - Mapping table for design_administration UoF (continued)

Application element	AIM element	Source	Rules	Reference path
request_date	date	41		change <= action_assignment action_assignment.assigned_action ->> action action.chosen_method ->> action_method <- action_request_solution.method action_request_solution action_request_solution.request ->> versioned_action_request building_design_date_item = versioned_action_request building_design_date_item <- building_design_date_assignment.items[i] building_design_date_assignment <= date_assignment {date_assignment.role ->> date_role date_role.name = 'request date'} date_assignment.assigned_date ->> date

Table 7 - Mapping table for design_administration UoF (continued)

Application element	AIM element	Source	Rules	Reference path
requestor	(person) (organization)	41 41		change <= action_assignment action_assignment.assigned_action ->& action action.chosen_method ->& action_method <-& action_request_solution.method action_request_solution action_request_solution.request ->& versioned_action_request (building_design_person_item = versioned_action_request building_design_person_item <-& building_design_person_assignment.items[i] building_design_person_assignment <=& person_assignment {person_assignment.role ->& person_role person_role.name = 'requestor'} person_assignment.assigned_person ->& person) (building_design_organization_item = versioned_action_request building_design_organization_item <-& building_design_organization_assignment.items[i] building_design_organization_assignment <=& organization_assignment {organization_assignment.role ->& organization_role organization_role.name = 'requestor'} organization_assignment.assigned_organization ->& organization)

Table 7 - Mapping table for design_administration UoF (continued)

Application element	AIM element	Source	Rules	Reference path
responsibility	person_and_organization	41		change <= action_assignment action_assignment.assigned_action ->> action action.chosen_method ->> action_method <- action_request_solution.method action_request_solution action_request_solution.request ->> versioned_action_request building_design_person_and_organization_item = versioned_action_request building_design_person_and_organization_item <- building_design_person_and_organization_assignment.items[i] building_design_person_and_organization_assignment <= person_and_organization_assignment {person_and_organization_assignment.role ->> person_and_organization_role person_and_organization_role.name = 'responsibility'} person_and_organization_assignment.assigned_person_and_organization ->> person_and_organization
solution	action_method.description	41		change <= action_assignment action_assignment.assigned_action ->> action action.chosen_method ->> action_method action_method.description

Table 7 - Mapping table for design_administration UoF (continued)

Application element	AIM element	Source	Rules	Reference path
status	action_request_status.status	41		change <= action_assignment action_assignment.assigned_action -> action action.chosen_method -> action_method <- action_request_solution.method action_request_solution action_request_solution.request -> versioned_action_request <- action_request_status.assigned_request action_request_status action_request_status.status
change_request to approval (as approval_information)	PATH			change building_design_approval_item = change building_design_approval_item <- building_design_approval.items[i] building_design_approval
change_request to building_element_-component (as change_to)	PATH			change change.items[i] -> building_design_change_item building_design_change_item = resulting_item resulting_item (resulting_item = negative_component negative_component) (resulting_item = positive_component positive_component)

Table 7 - Mapping table for design_administration UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
change_request to building_item (as change_to)	PATH			change change.items[i] ->building_design_change_item building_design_change_item = resulting_item resulting_item (resulting_item = building_element building_element) (resulting_item = fixture_equipment_element fixture_equipment_element) (resulting_item = service_element service_element) (resulting_item = space_element space_element) (resulting_item = structure_enclosure_element structure_enclosure_element)
change_request to building_item_-identification (as change_from)	PATH			change change.items[i] ->building_design_change_item building_design_change_item = identified_item {identified_item (identified_item = building_element) (identified_item = fixture_equipment_element) (identified_item = service_element) (identified_item = space_element) (identified_item = structure_enclosure_element) (identified_item = building_level) (identified_item = building_section) (identified_item = positive_component) (identified_item = negative_component)} identified_item <-building_item_identification_assignment.item building_item_identification_assignment

Table 8 - Mapping table for elementary_csg_representation UoF

Application element	AIM element	Source	Rules	Reference path
ELEMENTARY_FACE_- WITH_THICKNESS	elementary_face_with_thickness_shape_- representation	225	3, 6, 9	elementary_face_with_thickness_shape_representation <= [solid_model] [shape_representation {shape_representation <= representation representation.items[i] ->) ({representation_item representation_item.name = 'thickness'} representation_item => measure_representation_item <= measure_with_unit => length_measure_with_unit) (representation_item => topological_representation_item => face => face_surface face_surface.face_geometry -> surface => elementary_surface)}]
RIGHT_CIRCULAR_- CYLINDER	right_circular_cylinder	42		

Table 8 - Mapping table for elementary_csg_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference path
SOLID_OF_LINEAR_-EXTRUSION	extruded_area_solid	42	9	{extruded_area_solid <= swept_area_solid swept_area_solid.swept_area ->curve_bounded_surface [curve_bounded_surface.basis_surface ->surface => elementary_surface => plane] [curve_bounded_surface.boundaries[i] ->boundary_curve <= composite_curve_on_surface <= composite_curve composite_curve.segments[i] ->composite_curve_segment composite_curve_segment.parent_curve ->curve => bounded_curve => trimmed_curve trimmed_curve.basis_curve ->curve => (line) (conic)]}

Table 8 - Mapping table for elementary_csg_representation UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
SOLID_OF_- REVOLUTION	revolved_area_solid	42		{revolved_area_solid <= swept_area_solid swept_area_solid.swept_area ->> curve_bounded_surface curve_bounded_surface.boundaries[i] ->> boundary_curve <= composite_curve_on_surface <= composite_curve composite_curve.segments[i] ->> composite_curve_segment composite_curve_segment.parent_curve ->> curve => bounded_curve => trimmed_curve trimmed_curve.basis_curve ->> curve => (line) (conic)}
TRIMMED_SPHERE	boolean_result	42		{boolean_result boolean_result.first_operand ->> boolean_operand boolean_operand = csg_primitive csg_primitive csg_primitive = sphere}
TRIMMED_TORUS	boolean_result	42		{boolean_result boolean_result.first_operand ->> boolean_operand boolean_operand = csg_primitive csg_primitive csg_primitive = torus}
TRUNCATED_CONE	right_circular_cone	42		

Table 9 - Mapping table for elementary_geometric_representation UoF

Application element	AIM element	Source	Rules	Reference path
ELEMENTARY_B_REP	elementary_geometric_shape_- representation	225	3, 6	elementary_geometric_shape_representation <= shape_representation
ELEMENTARY_CURVE	elementary_wire_shape_representation	225	3, 6	elementary_wire_shape_representation <= shape_representation {shape_representation <= representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition -> property_definition property_definition.definition -> characterized_definition characterized_definition = characterized_product_definition characterized_product_definition characterized_product_definition = product_definition [product_definition => structure_enclosure_element] [product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] product_related_product_category <= product_category product_category.name = ‘structural wire’]}

Table 10 - Mapping table for faceted_csg_representation UoF

Application element	AIM element	Source	Rules	Reference path
BLOCK	block	42		
TRUNCATED_PYRAMID	truncated_pyramid	225		truncated_pyramid <= boolean_result

Table 11 - Mapping table for faceted_geometric_representation UoF

Application element	AIM element	Source	Rules	Reference path
FACET_TRIGON	poly_loop	42		
facet_trigon to point (as border)	PATH			poly_loop poly_loop.polygon[i] ->cartesian_point
FACETED_B_REP	faceted_geometric_shape_representation	225	3, 6	faceted_geometric_shape_representation <=shape_representation
FACETED_CURVE	faceted_wire_shape_representation	225	3, 6	faceted_wire_shape_representation <=shape_representation {shape_representation <=representation <-property_definition_representation.used_representation property_definition_representationproperty_definition_representation.definition ->property_definition property_definition.definition ->characterized_definition characterized_definition = characterized_product_definitioncharacterized_product_definition characterized_product_definition = product_definition[product_definition =>structure_enclosure_element] [product_definitionproduct_definition.formation ->product_definition_formation product_definition_formation.of_product ->product <- product_related_product_category.products[i]product_related_product_category <=product_category product_category.name = ‘structural wire’]}

Table 11 - Mapping table for faceted_geometric_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference path
FACETED_FACE_- WITH_THICKNESS	faceted_face_with_thickness_shape_- representation	225	3, 6, 9	faceted_face_with_thickness_shape_representation <= [solid_model] [shape_representation {shape_representation <= representation representation.items[i] ->> ({representation_item representation_item.name = 'thickness'} representation_item => measure_representation_item <= measure_with_unit => length_measure_with_unit) (representation_item => topological_representation_item => face face.bounds[i] ->> face_bound face_bound.bound ->> loop => poly_loop)}]
FACETED_SURFACE_- REPRESENTATION	site_representation	225	3, 6	site_representation <= shape_representation

Table 11 - Mapping table for faceted_geometric_representation UoF (continued)

Application element	AIM element	Source	Rules	Reference path
faceted_surface_- representation to facet_trigon (as facets)	PATH			site_representation <= shape_representation <= representation representation.items[i] ->> representation_item => topological_representation_item => connected_face_set connected_face_set.cfs_faces[i] ->> {face => face_surface} face face.bounds[i] ->> face_bound face_bound.bound ->> loop => poly_loop
POINT	cartesian_point	42		
POINT_AND_LINE_- REPRESENTATION	site_representation	225	3, 6	site_representation <= shape_representation
point_and_line_- representation to point (as survey_points)	PATH		7	site_representation <= shape_representation <= representation representation.items[i] ->> representation_item => geometric_representation_item => {geometric_set => geometric_curve_set} geometric_set geometric_set.elements[i] ->> geometric_set_select geometric_set_select = point point => cartesian_point
POLYLINE	polyline	42		

Table 11 - Mapping table for faceted_geometric_representation UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
polyline to point (as path)	PATH			polyline polyline.points[i] ->cartesian_point
SITE_SHAPE_-REPRESENTATION	site_representation	225	3	site_representation <= shape_representation
site_shape_representation to gis_position (as global_position)	PATH		5	site_representation <= shape_representation <= representation <- representation_map.mapped_representation representation_map <- mapped_item.mapping_source mapped_item <= representation_item <- representation.items[i] representation {representation representation.name = ‘gis position’}
site_shape_representation to polyline (as breaklines)	PATH		7	site_representation <= shape_representation <= representation representation.items[i] ->representation_item => geometric_representation_item => {geometric_set => geometric_curve_set} geometric_set geometric_set.elements[i] ->geometric_set_select geometric_set_select = curve curve => bounded_curve => polyline

Table 12 - Mapping table for property_and_classification UoF

Application element	AIM element	Source	Rules	Reference path
ITEM_CLASSIFICATION #1: A classification is assigned to an element or level #2: A classification is assigned to a component	#1: ([building_document_reference] [product_category]) #2: ([building_document_reference] [building_component_-classification_assignment])	225 41 225 225		#1: (building_document_reference <= document_reference) #2: ([building_document_reference <= document_reference] [building_component_classification_assignment <= group_assignment])
description #1: A classification is assigned to an element or level #2: A classification is assigned to a component	#1: (product_category.description) #2: (group.description)	41 41	8	#1:(product_category product_category.description) #2: (building_component_classification_assignment <= group_assignment group_assignment.assigned_group -> {group => building_component_classification_group} group group.description)
name	document.id	41		building_document_reference <= document_reference document_reference.assigned_document -> document document.id
notation	document_usage_constraint. subject_element_value	41		building_document_reference <= document_reference document_reference.assigned_document -> document <- document_usage_constraint.source document_usage_constraint document_usage_constraint.subject_element_value

Table 12 - Mapping table for property_and_classification UoF (continued)

Application element	AIM element	Source	Rules	Reference path
table	classification_table	225		building_document_reference <= document_reference document_reference.assigned_document -> document <- document_usage_constraint.source document_usage_constraint => classification_table
PROPERTY	property_definition	41		
code_of_measurement	representation_item.name	43		property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation representation.items[i] -> {representation_item => descriptive_representation_item} representation_item representation_item.name
formula	descriptive_representation_-item.description	45		property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation representation.items[i] -> representation_item => descriptive_representation_item descriptive_representation_item.description

Table 12 - Mapping table for property_and_classification UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
name	representation_item.name	43		property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation ->representation representation.items[i] ->{representation_item =>(measure_representation_item)(descriptive_representation_item)} representation_item representation_item.name
property_type	property_definition.name	41		{property_definition(property_definition.name)(property_definition.name = ‘material’)(property_definition.name = ‘performance’)(property_definition.name = ‘physical’)(property_definition.name = ‘surface’)}
value	([measure_with_unit.value_component][measure_with_unit.unit_component])(descriptive_representation_item.description)	41 41 45		property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation ->representation representation.items[i] ->representation_item =>(measure_representation_item <=measure_with_unit[measure_with_unit.value_component][measure_with_unit.unit_component])(descriptive_representation_item descriptive_representation_item.description)

Table 13 - Mapping table for space_boundary_representation UoF

Application element	AIM element	Source	Rules	Reference path
ADVANCED_SHELL	advanced_space_boundary_shape_-representation	225	3, 6	advanced_space_boundary_shape_representation <= shape_representation
ELEMENTARY_SHELL	elementary_space_boundary_shape_-representation	225	3, 6	elementary_space_boundary_shape_representation <= shape_representation
FACETED_SHELL	faceted_space_boundary_shape_-representation	225	3, 6	faceted_space_boundary_shape_representation <= shape_representation
GROUND_FACE	ground_face_space_boundary_shape_-representation	225	3, 6	ground_face_space_boundary_shape_representation <= shape_representation {shape_representation <= representation representation.items[1] -> topological_representation_item => face}

The following rules are referenced in the preceding table:

1) application_context_requires_ap_definition
2) building_element_maps_into_building_section
3) geometric_representation_context_3d
4) restrict_application_context
5) restrict_origin_and_target
6) shape_representation_subtype_exclusiveness
7) subtype_mandatory_geometric_set
8) subtype_mandatory_group
9) subtype_mandatory_solid_model

5.2 AIM EXPRESS short listing

This clause specifies the EXPRESS schema that uses elements from the integrated resources and contains the types, entity specializations, rules, and functions that are specific to this part of ISO 10303. This clause also specifies modifications to the textual material for the constructs that are imported from the integrated resources. The definitions and EXPRESS provided in the integrated resources for constructs used in the AIM may include select list items and subtypes which are not imported into the AIM. Requirements stated in the integrated resources which refer to such items and subtypes apply exclusively to those items which are imported into the AIM.

```
*)
SCHEMA building_design_schema;

   USE FROM action_schema                                  -- ISO 10303-41
  (action_request_solution,
   action_request_status,
   versioned_action_request);

   USE FROM application_context_schema                     -- ISO 10303-41
  (application_context,
   application_protocol_definition);

   USE FROM approval_schema                                -- ISO 10303-41
  (approval_date_time,
   approval_person_organization);

   USE FROM date_time_schema                               -- ISO 10303-41
  (calendar_date,
   ordinal_date,
   week_of_year_and_day_date);

   USE FROM document_schema                                -- ISO 10303-41
  (document_usage_constraint);

   USE FROM geometric_model_schema                         -- ISO 10303-42
  (block,
   boolean_operand,
   boolean_result,
   brep_with_voids,
   csg_solid,
   extruded_area_solid,
   faceted_brep,
   geometric_curve_set,
   geometric_set,
   manifold_solid_brep,
   revolved_area_solid,
   right_circular_cone,
```

```
    right_circular_cylinder,
    solid_model,
    sphere,
    torus);

   USE FROM geometry_schema                                -- ISO 10303-42
  (axis2_placement_2d,
    axis2_placement_3d,
    bezier_curve,
    bezier_surface,
    b_spline_curve_with_knots,
    b_spline_surface_with_knots,
    cartesian_point,
    cartesian_transformation_operator_3d,
    circle,
    conical_surface,
    curve,
    curve_bounded_surface,
    cylindrical_surface,
    degenerate_toroidal_surface,
    direction,
    ellipse,
    geometric_representation_context,
    hyperbola,
    intersection_curve,
    line,
    outer_boundary_curve,
    parabola,
    pcurve,
    placement,
    plane,
    polyline,
    quasi_uniform_curve,
    quasi_uniform_surface,
    rational_b_spline_curve,
    rational_b_spline_surface,
    rectangular_composite_surface,
    rectangular_trimmed_surface,
    spherical_surface,
    surface,
    surface_curve,
    surface_of_linear_extrusion,
    surface_of_revolution,
    surface_patch,
    swept_surface,
    toroidal_surface,
    trimmed_curve,
    uniform_curve,
    uniform_surface,
    vector);

   USE FROM group_schema                                   -- ISO 10303-41
  (group);

   USE FROM management_resources_schema                    -- ISO 10303-41
  (action_assignment,
    approval_assignment,
    date_assignment,
    document_reference,
    group_assignment,
    name_assignment,
    organization_assignment,
    person_and_organization_assignment,
    person_assignment);
```

```
 USE FROM measure_schema                                    -- ISO 10303-41
(conversion_based_unit,
 derived_unit,
 global_unit_assigned_context,
 length_measure,
 length_measure_with_unit,
 length_unit,
 parameter_value,
 plane_angle_measure_with_unit,
 plane_angle_measure,
 plane_angle_unit,
 positive_length_measure,
 si_unit);

 USE FROM person_organization_schema                        -- ISO 10303-41
(organizational_project);

 USE FROM product_definition_schema                         -- ISO 10303-41
(product_category,
 product_category_relationship,
 product_definition,
 product_definition_relationship,
 product_related_product_category);

 USE FROM product_property_definition_schema                -- ISO 10303-41
(characterized_object,
 property_definition,
 shape_aspect,
 shape_aspect_relationship);

 USE FROM product_property_representation_schema            -- ISO 10303-41
(shape_definition_representation,
 shape_representation);

 USE FROM product_structure_schema                          -- ISO 10303-44
(assembly_component_usage);

 USE FROM qualified_measure_schema                          -- ISO 10303-45
(descriptive_representation_item,
 measure_representation_item);

 USE FROM representation_schema                             -- ISO 10303-43
(definitional_representation,
 mapped_item,
 parametric_representation_context,
 representation_item,
 representation_relationship_with_transformation);

 USE FROM topology_schema                                   -- ISO 10303-42
(closed_shell,
 connected_face_set,
 edge,
 edge_curve,
 edge_loop,
 face,
 face_bound,
 face_outer_bound,
 face_surface,
 loop,
 oriented_closed_shell,
 oriented_edge,
 path,
 poly_loop,
 vertex,
```

```
   vertex_loop,
   vertex_point);
(*
```

NOTE - The schemas referenced above can be found in the following parts of ISO 10303:

action_schema	ISO 10303-41
application_context_schema	ISO 10303-41
approval_schema	ISO 10303-41
date_time_schema	ISO 10303-41
document_schema	ISO 10303-41
geometric_model_schema	ISO 10303-42
geometry_schema	ISO 10303-42
group_schema	ISO 10303-41
management_resources_schema	ISO 10303-41
measure_schema	ISO 10303-41
person_organization_schema	ISO 10303-41
product_definition_schema	ISO 10303-41
product_property_definition_schema	ISO 10303-41
product_property_representation_schema	ISO 10303-41
product_structure_schema	ISO 10303-44
qualified_measure_schema	ISO 10303-45
representation_schema	ISO 10303-43
topology_schema	ISO 10303-42

5.2.1 building_design type definitions

5.2.1.1 building_component_classification_item

A **building_component_classification_item** is a component to which a classification is assigned.

EXPRESS specification:

```
*)
TYPE building_component_classification_item = SELECT
  (negative_component,
   positive_component,
   opening,
   recess);
END_TYPE;
(*
```

5.2.1.2 building_design_approval_item

A **building_design_approval_item** is an item to which approval is given.

EXPRESS specification:

```
*)
TYPE building_design_approval_item = SELECT
  (building_element_assembly,
   building_element,
   fixture_equipment_element,
   service_element,
```

```
   space_element,
   structure_enclosure_element,
   change,
   positive_component,
   negative_component);
END_TYPE;
(*
```

5.2.1.3 building_design_change_item

A **building_design_change_item** is an item to which a design change is assigned.

EXPRESS specification:

```
*)
TYPE building_design_change_item = SELECT
  (identified_item,
   resulting_item);
END_TYPE;
(*
```

5.2.1.4 building_design_date_item

A **building_design_date_item** is an item to which a date is assigned.

EXPRESS specification:

```
*)
TYPE building_design_date_item = SELECT
  (versioned_action_request);
END_TYPE;
(*
```

5.2.1.5 building_design_organization_item

A **building_design_organization_item** is an item to which an organization is assigned.

EXPRESS specification:

```
*)
TYPE building_design_organization_item = SELECT
  (building,
   building_complex,
   building_item_identification_assignment,
   versioned_action_request);
END_TYPE;
(*
```

5.2.1.6 building_design_person_and_organization_item

A **building_design_person_and_organization_item** is an item to which a **person_and_organization** is assigned.

EXPRESS specification:

```
*)
TYPE building_design_person_and_organization_item = SELECT
  (versioned_action_request);
END_TYPE;
(*
```

5.2.1.7 building_design_person_item

A **building_design_person_item** is an item to which a person is assigned.

EXPRESS specification:

```
*)
TYPE building_design_person_item = SELECT
  (building,
   building_complex,
   building_item_identification_assignment,
   versioned_action_request);
END_TYPE;
(*
```

5.2.1.8 building_document_item

A **building_document_item** is an item to which a document reference is assigned.

EXPRESS specification:

```
*)
TYPE building_document_item = SELECT
  (negative_component,
   positive_component,
   building_element,
   fixture_equipment_element,
   service_element,
   space_element,
   structure_enclosure_element,
   building_component_classification_group,
   product_category);
END_TYPE;
(*
```

5.2.1.9 identified_item

An **identified_item** is an item to which a name is assigned.

EXPRESS specification:

```
*)
TYPE identified_item = SELECT
  (building_element,
   fixture_equipment_element,
   service_element,
   space_element,
   structure_enclosure_element,
   building_level,
```

```
   building_section,
   positive_component,
   negative_component);
END_TYPE;
(*
```

5.2.1.10 resulting_item

A **resulting_item** is an item that results from a change.

EXPRESS specification:

```
*)
TYPE resulting_item = SELECT
  (building_element,
   fixture_equipment_element,
   service_element,
   space_element,
   structure_enclosure_element,
   positive_component,
   negative_component);
END_TYPE;
(*
```

5.2.2 building_design entities

5.2.2.1 building_design entity definitions

5.2.2.1.1 advanced_brep_building_shape_representation

The **advanced_brep_building_shape_representation** is a subtype of **shape_representation** in which the **representation_item**s are specialisations of **manifold_solid_brep** entities or reference lines. These differ from the more general B-rep in having only explicit geometric forms for their faces and edges. The face geometry is restricted to elementary surfaces, swept surfaces or B-spline surfaces.

EXPRESS specification:

```
*)
ENTITY advanced_brep_building_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (it <* SELF.items |
       NOT ((SIZEOF (['BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP',
       'BUILDING_DESIGN_SCHEMA.FACETED_BREP',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
       'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D'] *
       TYPEOF(it)) = 1) OR
       (it.name = 'reference curve')))) = 0;
  WR2: (SIZEOF (QUERY (it <* SELF.items |
       SIZEOF (['BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF(it)) = 1))
       >= 1) OR
       ({2 <= SIZEOF (QUERY (it <* SELF.items |
       it.name = 'reference curve')) <= 3});
  WR3: SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP' IN TYPEOF(it))
       | NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
```

```
        NOT (SIZEOF (QUERY (fcs <* csh.cfs_faces |
        NOT ('BUILDING_DESIGN_SCHEMA.ADVANCED_FACE' IN
        TYPEOF(fcs)))) = 0))) = 0))) = 0;
  WR4: SIZEOF (QUERY (msb <* QUERY (it <* items |
        'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP' IN TYPEOF(it))
        | 'BUILDING_DESIGN_SCHEMA.ORIENTED_CLOSED_SHELL' IN
        TYPEOF (msb.outer))) = 0;
  WR5: SIZEOF (QUERY (brv <* QUERY (it <* items |
        'BUILDING_DESIGN_SCHEMA.BREP_WITH_VOIDS' IN TYPEOF(it)) |
        NOT (SIZEOF (QUERY ( csh <* brv.voids |
        csh.orientation))  = 0))) = 0;
  WR6: SIZEOF (QUERY (mi <* QUERY (it <* items |
        'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF(it)) |
        NOT ('BUILDING_DESIGN_SCHEMA.' +
        'ADVANCED_BREP_BUILDING__SHAPE_REPRESENTATION' IN
        TYPEOF (mi.mapping_source.mapped_representation)))) = 0;
  WR7: SIZEOF (QUERY (it <* SELF.items |
        (it.name = 'reference curve') AND
        (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF (it)) OR
        (('BUILDING_DESIGN_SCHEMA_TRIMMED_CURVE' IN TYPEOF (it)) AND
        ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN
        TYPEOF (it\trimmed_curve.basis_curve)))))))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: The items attribute of the **representation** supertype may contain **manifold_solid_brep**s, **mapped_item**s and **axis2_placement_3d**s or have a name of 'reference curve'. A **faceted_brep** is excluded from the set of items since this would be of type **faceted_brep** and of type **manifold_solid_brep**.

WR2: At least one **representation_item** in the set of items shall be a **manifold_solid_brep** entity or a **mapped_item**, or between two and three **representation_item**s in the set of items shall have a name of 'reference curve'.

NOTE - See also WR6.

WR3: For each **manifold_solid_brep** in the set of items each **face** shall be an **advanced_face**. This rule ensures that the following conditions are met:

a) Each **face** is a **face_surface**;

b) Each **face_surface** has its geometry defined by an **elementary_surface**, **swept_surface**, or a **b_spline_surface**;

c) The **edge**s used to define the boundaries of the **face** shall all reference an **edge_curve**;

d) Each **curve** used to define the geometry of the **face**s and face bounds shall be either a **conic**, **line**, **polyline**, or **b_spline_curve**;

e) The **edge**s used to define the face boundaries shall all be trimmed by vertices of type **vertex_point**;

f) No **loop** used to define a **face_bound** shall be of the oriented subtype.

WR4: For each **manifold_solid_brep** in the set of items the outer_shell attribute shall not be of the oriented subtype.

WR5: If a **brep_with_void**s is included in the set of items then each **shell** in the voids set shall be an **oriented_closed_shell** with orientation value FALSE.

WR6: If a **mapped_item** is included in the set of items then the mapped_representation of the mapping_source attribute shall be an **advanced_brep_building_shape_representation**.

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

WR7: Each **representation_item** with the name of 'reference curve' in the set of items of an **advanced_brep_building_shape_representation** shall be either a **polyline** or a **trimmed_curve** with a basis_curve that is a **circle**.

5.2.2.1.2 advanced_csg_shape_representation

An **advanced_csg_shape_representation** is a **shape_representation** that is composed of CSG primitives and boolean operators.

EXPRESS specification:

```
*)
ENTITY advanced_csg_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF.items |
       NOT ((SIZEOF (['BUILDING_DESIGN_SCHEMA.CSG_SOLID',
       'BUILDING_DESIGN_SCHEMA.EXTRUDED_AREA_SOLID',
       'BUILDING_DESIGN_SCHEMA.REVOLVED_AREA_SOLID',
       'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF (item)) = 1) OR
       (item.name = 'reference curve')))) = 0;
  WR2: (SIZEOF (QUERY (item <* SELF.items |
       (SIZEOF (['BUILDING_DESIGN_SCHEMA.CSG_SOLID',
       'BUILDING_DESIGN_SCHEMA.EXTRUDED_AREA_SOLID',
       'BUILDING_DESIGN_SCHEMA.REVOLVED_AREA_SOLID',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF (item)) = 1))) >= 1)
       OR ({2 <= SIZEOF (QUERY (it <* SELF.items |
       it.name = 'reference curve')) <= 3});
  WR3: SIZEOF (QUERY (item <* SELF.items |
       ('BUILDING_DESIGN_SCHEMA.CSG_SOLID' IN TYPEOF (item)) AND
       (NOT (valid_advanced_csg_tree
                (item\csg_solid.tree_root_expression))))) = 0;
  WR4: SIZEOF (QUERY (mi <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF (item)) |
       NOT ('BUILDING_DESIGN_SCHEMA.ADVANCED_CSG_SHAPE_REPRESENTATION' IN
       TYPEOF (mi\mapped_item.mapping_source.mapped_representation)))) = 0;
  WR5: SIZEOF (QUERY (it <* SELF.items |
       (it.name = 'reference curve') AND
       (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF (it)) OR
       (('BUILDING_DESIGN_SCHEMA_TRIMMED_CURVE' IN TYPEOF (it)) AND
       ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN
```

```
       TYPEOF (it\trimmed_curve.basis_curve))))))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: Each item of an **advanced_csg_shape_representation** shall be a **csg_solid**, **extruded_area_solid**, **revolved_area_solid**, **axis2_placement_3d**, or **mapped_item** or has a name of 'reference curve'.

WR2: An **advanced_csg_shape_representation** shall have at least one **representation_item** instance in its set of items that is of type **csg_solid**, **extruded_area_solid**, **revolved_area_solid**, or **mapped_item**, or between two and three **representation_item**s in the set of items shall have a name of 'reference curve'.

WR3: An **advanced_csg_shape_representation** shall be comprised of the proper CSG tree elements.

WR4: For each **mapped_item** in an **advanced_csg_shape_representation**, the source of the **mapped_item** shall be an **advanced_csg_shape_representation**.

WR5: Each **representation_item** with the name of 'reference curve' in the set of items of an **advanced_csg_shape_representation** shall be either a **polyline** or a **trimmed_curve** with a basis_curve that is a **circle**.

5.2.2.1.3 advanced_face

An **advanced_face** is a special subtype of **face_surface** which has additional constraints to ensure that the geometry is directly and completely defined. The **advanced_face** is used to formulate the precise meaning of a topologically bounded surface.

EXPRESS specification:

```
*)
ENTITY advanced_face
  SUBTYPE OF (face_surface);
WHERE
  WR1 : SIZEOF (['BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE',
        'BUILDING_DESIGN_SCHEMA.B_SPLINE_SURFACE',
        'BUILDING_DESIGN_SCHEMA.SWEPT_SURFACE'] *
        TYPEOF (face_geometry)) = 1;
  WR2 : SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* SELF.bounds |
        'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN
        TYPEOF (bnds.bound)) |
        NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
        NOT('BUILDING_DESIGN_SCHEMA.EDGE_CURVE' IN
        TYPEOF(oe.edge_element)))) = 0))) = 0;
  WR3 : SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* SELF.bounds |
        'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN
        TYPEOF (bnds.bound)) |
        NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
        NOT (SIZEOF (['BUILDING_DESIGN_SCHEMA.LINE',
        'BUILDING_DESIGN_SCHEMA.CONIC',
        'BUILDING_DESIGN_SCHEMA.POLYLINE',
        'BUILDING_DESIGN_SCHEMA.SURFACE_CURVE',
        'BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE'] *
```

```
        TYPEOF (oe.edge_element\edge_curve.edge_geometry)) = 1))) = 0)))
        = 0;
  WR4 : SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* SELF.bounds |
        'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN
        TYPEOF (bnds.bound)) |
        NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
        NOT ((('BUILDING_DESIGN_SCHEMA.VERTEX_POINT' IN
        TYPEOF (oe.edge_start)) AND
        ('BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT' IN
        TYPEOF (oe.edge_start\vertex_point.vertex_geometry)))) AND
        (('BUILDING_DESIGN_SCHEMA.VERTEX_POINT' IN
        TYPEOF (oe.edge_end)) AND
        ('BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT' IN
        TYPEOF (oe.edge_end\vertex_point.vertex_geometry)))))) = 0))) = 0;
  WR5 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* SELF.bounds |
        'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN
         TYPEOF (bnds.bound)) |
         'BUILDING_DESIGN_SCHEMA.ORIENTED_PATH' IN
         TYPEOF (elp_fbnds.bound))) = 0;
  WR6 : (NOT ('BUILDING_DESIGN_SCHEMA.SWEPT_SURFACE' IN
        TYPEOF (face_geometry))) OR
        (SIZEOF (['BUILDING_DESIGN_SCHEMA.LINE',
        'BUILDING_DESIGN_SCHEMA.CONIC',
        'BUILDING_DESIGN_SCHEMA.POLYLINE',
        'BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE'] *
        TYPEOF (face_geometry\swept_surface.swept_curve)) = 1);
  WR7 : SIZEOF (QUERY (vlp_fbnds <* QUERY (bnds <* SELF.bounds |
        'BUILDING_DESIGN_SCHEMA.VERTEX_LOOP' IN
        TYPEOF(bnds.bound)) |
        NOT (('BUILDING_DESIGN_SCHEMA.VERTEX_POINT' IN
        TYPEOF (vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex)) AND
        ('BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT' IN
        TYPEOF (vlp_fbnds\face_bound.bound\vertex_loop.
        loop_vertex\vertex_point.vertex_geometry))))) = 0;
  WR8 : SIZEOF (QUERY (bnd <* SELF.bounds |
        NOT (SIZEOF (['BUILDING_DESIGN_SCHEMA.EDGE_LOOP',
        'BUILDING_DESIGN_SCHEMA.VERTEX_LOOP'] *
        TYPEOF(bnd.bound)) = 1))) = 0;
  WR9 : SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* SELF.bounds |
        'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN
        TYPEOF (bnds.bound)) |
        NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
        ('BUILDING_DESIGN_SCHEMA.SURFACE_CURVE' IN
        TYPEOF (oe.edge_element\edge_curve.edge_geometry)) AND
        (NOT (SIZEOF (QUERY (sc_ag <*
        oe.edge_element\edge_curve.edge_geometry\
        surface_curve.associated_geometry |
        NOT ('BUILDING_DESIGN_SCHEMA.PCURVE' IN
        TYPEOF (sc_ag)))) = 0)))) = 0))) = 0;
  WR10 : ((NOT ('BUILDING_DESIGN_SCHEMA.SWEPT_SURFACE' IN
         TYPEOF(face_geometry))) OR
         ((NOT ('BUILDING_DESIGN_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 <* SELF.bounds |
         'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN
         TYPEOF (bnds.bound)) |
         NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
         ('BUILDING_DESIGN_SCHEMA.POLYLINE' IN
         TYPEOF(oe.edge_element\edge_curve.edge_geometry)) AND
         (NOT (SIZEOF (oe.edge_element\edge_curve.edge_geometry\
         polyline.points) < 3)))) = 0))) = 0);
END_ENTITY;
```

(*

Formal propositions:

WR1: The geometry used in the definition of the face shall be restricted. The face geometry shall be an **elementary_surface**, **swept_surface**, or **b_spline_surface**.

WR2: The geometry of all bounding edges of the face shall be fully defined as **edge_curve**s.

WR3: The types of curve used to define the geometry of edges shall be restricted to **line**s, **conic**s, **polyline**s, **surface_curve**s, or **b_spline_curve**s.

WR4: All vertices used in the face definition shall be of type **vertex_point** with geometry defined by a **cartesian_point**.

WR5: The use of oriented paths in the definition of the **edge_loop**s of the **advanced_face** is prohibited.

WR6: If the face geometry is of type **swept_surface**, then the swept_curve used in the definition shall be of type **line**, **conic**, **polyline**, or **b_spline_curve**.

WR7: For any **vertex_loop** used to bound the face, the loop_vertex shall be of type **vertex_point** and the geometry shall be defined by a **cartesian_point**.

WR8: The face bounds shall be defined by either **edge_loop**s or **vertex_loop**s.

WR9: If a **surface_curve** is used as part of a face bound then the associated_geometry attribute shall reference **pcurve**s not **surface**s.

WR10: If a **polyline** is used either to define a **swept_surface** or as part of a face bound, it shall contain at least three points.

5.2.2.1.4 advanced_face_with_thickness_shape_representation

An **advanced_face_with_thickness_shape_representation** is a solid model that is defined by a **face** with underlying geometry that is comprised of analytic or advanced surfaces with a length that defines the thickness.

EXPRESS specification:

```
*)
ENTITY advanced_face_with_thickness_shape_representation
  SUBTYPE OF (shape_representation, solid_model);
WHERE
  WR1: SIZEOF (SELF.items) = 2;
  WR2: SIZEOF (QUERY (item <* SELF.items |
       (SIZEOF (['BUILDING_DESIGN_SCHEMA.MEASURE_REPRESENTATION_ITEM',
       'BUILDING_DESIGN_SCHEMA.LENGTH_MEASURE_WITH_UNIT'] *
       TYPEOF (item)) = 2) AND (item.name = 'thickness'))) = 1;
  WR3: SIZEOF (QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.ADVANCED_FACE' IN TYPEOF (item))) = 1;
END_ENTITY;
```

```
(*
```

Formal propositions:

WR1: An **advanced_face_with_thickness_shape_representation** shall contain exactly two **representation_item**s in its set of items.

WR2: Exactly one **representation_item** in the set of items of an **advanced_face_with_thickness_-shape_representation** shall be a **measure_representation_item** and a **length_measure_with_unit** with a name of 'thickness'.

WR3: Exactly one **representation_item** in the set of items of an **advanced_face_with_thickness_-shape_representation** shall be an **advanced_face**.

5.2.2.1.5 advanced_space_boundary_shape_representation

An **advanced_space_boundary_shape_representation** is a **shape_representation** that represents the shape of a space in a building. This shape or aspect of shape shall be represented using advanced **closed_shell**s.

EXPRESS specification:

```
*)
ENTITY advanced_space_boundary_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF.items |
       NOT ((SIZEOF (['BUILDING_DESIGN_SCHEMA.CLOSED_SHELL',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
       'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D'] *
       TYPEOF (item)) = 1) OR
       (item.name = 'reference curve')))) = 0;
  WR2: (SIZEOF (QUERY (item <* SELF.items |
       SIZEOF (['BUILDING_DESIGN_SCHEMA.CLOSED_SHELL',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF (item)) = 1)) = 1)
       OR ({2 <= SIZEOF (QUERY (it <* SELF.items |
       it.name = 'reference curve')) <= 3});
  WR3: SIZEOF (QUERY (csh <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF (item)) |
       NOT (SIZEOF (QUERY (fcs <* csh\connected_face_set.cfs_faces |
       NOT ('BUILDING_DESIGN_SCHEMA.ADVANCED_FACE' IN TYPEOF (fcs))))
       = 0)))
       = 0;
  WR4: SIZEOF (QUERY (mi <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF (item)) |
       NOT ('BUILDING_DESIGN_SCHEMA.' +
       'ADVANCED_SPACE_BOUNDARY_SHAPE_REPRESENTATION' IN
       TYPEOF (mi\mapped_item.mapping_source.mapped_representation)))) = 0;
  WR5: SIZEOF (QUERY (it <* SELF.items |
       (it.name = 'reference curve') AND
       (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF (it)) OR
       (('BUILDING_DESIGN_SCHEMA_TRIMMED_CURVE' IN TYPEOF (it)) AND
       ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN
       TYPEOF (it\trimmed_curve.basis_curve)))))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: Each item of an **advanced_space_boundary_shape_representation** shall be a **closed_shell**, **mapped_item**, or **axis2_placement_3d** or has a name of 'reference curve'.

WR2: Exactly one of the items of an **advanced_space_boundary_shape_representation** shall be a **closed_shell** or **mapped_item**, or between two and three **representation_item**s in the set of items shall have a name of 'reference curve'.

WR3: For each **closed_shell** in the items of an **advanced_space_boundary_shape_representation**, each face that defines the boundary of the shell shall be an **advanced_face**.

WR4: For each **mapped_item** in an **advanced_space_boundary_shape_representation**, the source of the **mapped_item** shall be an **advanced_space_boundary_shape_representation**.

WR5: Each **representation_item** with the name of 'reference curve' in the set of items of an **advanced_space_boundary_shape_representation** shall be either a **polyline** or a **trimmed_curve** with a basis_curve that is a **circle**.

5.2.2.1.6 advanced_wire_shape_representation

An **advanced_wire_shape_representation** is a shape that is defined by a single quadric curve or a **composite_curve** that is comprised of simple quadric curves.

EXPRESS specification:

```
*)
ENTITY advanced_wire_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF.items |
       NOT ((SIZEOF (['BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
       'BUILDING_DESIGN_SCHEMA.CIRCLE',
       'BUILDING_DESIGN_SCHEMA.ELLIPSE',
       'BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE',
       'BUILDING_DESIGN_SCHEMA.OFFSET_CURVE_3D',
       'BUILDING_DESIGN_SCHEMA.CURVE_REPLICA',
       'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
       'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D']
       * TYPEOF (item)) = 1) OR
       (item.name = 'reference curve')))) = 0;
  WR2: (SIZEOF (QUERY (item <* SELF.items |
       SIZEOF (['BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
       'BUILDING_DESIGN_SCHEMA.CIRCLE',
       'BUILDING_DESIGN_SCHEMA.ELLIPSE',
       'BUILDING_DESIGN_SCHEMA.OFFSET_CURVE_3D',
       'BUILDING_DESIGN_SCHEMA.CURVE_REPLICA',
       'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF (item)) = 1)) = 1)
       OR ({2 <= SIZEOF (QUERY (it <* SELF.items |
       it.name = 'reference curve')) <= 3});
  WR3: SIZEOF (QUERY (item <* SELF.items |
       (SIZEOF (['BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
       'BUILDING_DESIGN_SCHEMA.OFFSET_CURVE_3D',
       'BUILDING_DESIGN_SCHEMA.CURVE_REPLICA',
```

```
        'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE'] * TYPEOF (item)) = 1) AND
        (NOT (valid_advanced_wire_composition (item))))) = 0;
  WR4: SIZEOF (QUERY (mi <* QUERY (item <* SELF.items |
        'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF (item)) |
        NOT ('BUILDING_DESIGN_SCHEMA.' +
        'ADVANCED_WIRE_SHAPE_REPRESENTATION' IN
        TYPEOF (mi\mapped_item.mapping_source.mapped_representation)))) = 0;
  WR5: SIZEOF (QUERY (it <* SELF.items |
        (it.name = 'reference curve') AND
        (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF (it)) OR
        (('BUILDING_DESIGN_SCHEMA_TRIMMED_CURVE' IN TYPEOF (it)) AND
        ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN
        TYPEOF (it\trimmed_curve.basis_curve)))))))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: Each item of an **advanced_wire_shape_representation** shall be a **composite_curve**, **circle**, **ellipse**, **offset_curve_3d**, **curve_replica**, **trimmed_curve**, **mapped_item**, or **axis2_placement_3d** or has a name of 'reference curve'.

WR2: Exactly one of the items of an **advanced_wire_shape_representation** shall be a **composite_-curve**, **circle**, **ellipse**, **offset_curve_3d**, **curve_replica**, **trimmed_curve**, or **mapped_item**, or between two and three **representation_item**s in the set of items shall have a name of 'reference curve'.

WR3: If a **representation_item** in the set of items is a **trimmed_curve**, **offset_curve_3d**, **curve_-replica**, or **composite_curve** it shall reference valid advanced curves.

WR4: For each **mapped_item** in an **advanced_wire_shape_representation**, the source of the **mapped_item** shall be an **advanced_wire_shape_representation**.

WR5: Each **representation_item** with the name of 'reference curve' in the set of items of an **advanced_space_boundary_shape_representation** shall be either a **polyline** or a **trimmed_curve** with a basis_curve that is a **circle**.

5.2.2.1.7 building

A **building** is a **product_definition** that defines a building.

EXPRESS specification:

```
*)
ENTITY building
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.'+
        'PRODUCT_DEFINITION_RELATIONSHIP.RELATING_PRODUCT_DEFINITION') |
        'BUILDING_DESIGN_SCHEMA.BUILDING_SECTION'
        IN TYPEOF (pdr.related_product_definition))) >= 1;
  WR2: SIZEOF (QUERY (pdr <* USEDIN(SELF, 'BUILDING_DESIGN_SCHEMA.' +
        'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION') |
        'BUILDING_DESIGN_SCHEMA.BUILDING_COMPLEX'
        IN TYPEOF (pdr.relating_product_definition))) = 1;
  WR3: SIZEOF (QUERY (bdpa <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
        'BUILDING_DESIGN_PERSON_ASSIGNMENT.ITEMS') |
```

```
       bdpa.role.name = 'owner')) +
       SIZEOF (QUERY (bdoa <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'BUILDING_DESIGN_ORGANIZATION_ASSIGNMENT.ITEMS') |
       bdoa.role.name = 'owner')) = 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **building** shall be used as the relating_product_definition in at least one **product_definition_-relationship** where the related_product_definition is a **building_section**.

WR2: A **building** shall be the related_product_definition in exactly one **product_definition_-relationship** where the relating_product_definition is a **building_complex**.

WR3: A **building** shall have exactly one **person** or **organization** as its owner.

5.2.2.1.8 building_complex

A **building_complex** is a **product_definition** that defines a collection of buildings.

EXPRESS specification:

```
*)
ENTITY building_complex
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'PRODUCT_DEFINITION_RELATIONSHIP.RELATING_PRODUCT_DEFINITION') |
       (pdr.name = 'in building complex')
               AND
               ('BUILDING_DESIGN_SCHEMA.BUILDING'
       IN TYPEOF (pdr.related_product_definition)))) >= 1;
  WR2: SIZEOF (QUERY (pa <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'BUILDING_DESIGN_PERSON_ASSIGNMENT.ITEMS') |
       pa.role.name = 'owner'))+
       SIZEOF (QUERY (oa <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'BUILDING_DESIGN_ORGANIZATION_ASSIGNMENT.ITEMS') |
       oa.role.name = 'owner')) = 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **building_complex** shall be used as the relating_product_definition in at least one **product_-definition_relationship** with a name of 'in building complex', where the related_product_definition is a **building**.

WR2: A **building_complex** shall be an item in exactly one **building_design_person_assignment** with a role of 'owner' or exactly one **building_design_organization_assignment** with a role of 'owner'.

5.2.2.1.9 building_component_classification_assignment

A **building_component_classification_assignment** is a **group_assignment** for a set of **building_component_classification_item** instances.

EXPRESS specification:

```
*)
ENTITY building_component_classification_assignment
  SUBTYPE OF (group_assignment);
  items : SET[1:?] OF building_component_classification_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the **positive_component**s and **negative_component**s to which a group is assigned for classification purposes.

5.2.2.1.10 building_component_classification_group

A **building_component_classification_group** is a **group** for the classification of instances of **negative_component** and **positive_component** entities.

EXPRESS specification:

```
*)
ENTITY building_component_classification_group
  SUBTYPE OF (group);
WHERE
  WR1: SIZEOF (QUERY (ga <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'GROUP_ASSIGNMENT.ASSIGNED_GROUP') |
       'BUILDING_DESIGN_SCHEMA.' +
         'BUILDING_COMPONENT_CLASSIFICATION_ASSIGNMENT' IN TYPEOF (ga)))
         >= 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **building_component_classification_group** shall be used as the **assigned_group** by at least one **building_component_classification_assignment** instance.

5.2.2.1.11 building_design_approval

A **building_design_approval** is an **approval_assignment** for a set of **building_design_approval_item** instances.

EXPRESS specification:

```
*)
ENTITY building_design_approval
  SUBTYPE OF (approval_assignment);
  items : SET[1:?] OF building_design_approval_item;
```

```
WHERE
  WR1: SIZEOF( QUERY( i <* SELF.items |
       SIZEOF (['BUILDING_DESIGN_SCHEMA.BUILDING',
       'BUILDING_DESIGN_SCHEMA.BUILDING_LEVEL',
       'BUILDING_DESIGN_SCHEMA.BUILDING_SECTION',
       'BUILDING_DESIGN_SCHEMA.ELEMENT_ASSEMBLY'] * TYPEOF(i)) = 1)) = 0;
END_ENTITY;
(*
```

Attribute definitions:

items: the **building_design_approval_item** instances to which approval is assigned.

Formal propositions:

WR1: A **building_design_approval** shall not be used to indicate approval for a **building**, **building_-level**, or **building_section**.

5.2.2.1.12 building_design_date_assignment

A **building_design_date_assignment** is a **date_assignment** for a set of **building_design_date_item** instances.

EXPRESS specification:

```
*)
ENTITY building_design_date_assignment
  SUBTYPE OF (date_assignment);
  items : SET[1:?] OF building_design_date_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the **versioned_action_request**s to which a date is assigned.

5.2.2.1.13 building_design_organization_assignment

A **building_design_organization_assignment** is an **organization_assignment** for a set of **building_-design_organization_item** instances.

EXPRESS specification:

```
*)
ENTITY building_design_organization_assignment
  SUBTYPE OF (organization_assignment);
  items : SET[1:?] OF building_design_organization_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the **building**s and **building_item_identification_assignment**s to which an organization is assigned.

5.2.2.1.14 building_design_person_and_organization_assignment

A **building_design_person_and_organization_assignment** is a **person_and_organization_-assignment** for a set of **building_design_person_and_organization_item** instances.

EXPRESS specification:

```
*)
ENTITY building_design_person_and_organization_assignment
  SUBTYPE OF (person_and_organization_assignment);
  items : SET[1:?] OF building_design_person_and_organization_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the **versioned_action_request**s to which a person and organization is assigned.

5.2.2.1.15 building_design_person_assignment

A **building_design_person_assignment** is a **person_assignment** for a set of **building_design_-person_item** instances.

EXPRESS specification:

```
*)
ENTITY building_design_person_assignment
  SUBTYPE OF (person_assignment);
  items : SET[1:?] OF building_design_person_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the **building**s, **building_item_identification_assignment**s, and **versioned_action_request**s to which a person is assigned.

5.2.2.1.16 building_document_reference

A **building_document_reference** is a **document_reference** for a set of **building_document_item** instances.

EXPRESS specification:

```
*)
ENTITY building_document_reference
  SUBTYPE OF (document_reference);
  items : SET[1:?] OF building_document_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the **negative_component**s, **positive_component**s, **building_element**s, **fixture_equipment_-element**s, **service_element**s, **space_element**s, **structure_enclosure_element**s, **building_component_-classification_group**s, and **product_category**s to which a document is assigned.

5.2.2.1.17 building_element

A **building_element** is a **product_definition** that is a physical part of a building or building system and has a characteristic function.

EXPRESS specification:

```
*)
ENTITY building_element
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN(SELF, 'BUILDING_DESIGN_SCHEMA.' +
          'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION') |
         ('BUILDING_DESIGN_SCHEMA.BUILDING_SECTION'
              IN TYPEOF (pdr.relating_product_definition))
       )) = 1;
  WR2: SIZEOF (USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
               'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
  WR3: SIZEOF (QUERY (pd <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.'+
                                  'PROPERTY_DEFINITION.DEFINITION') |
         SIZEOF (QUERY (sa <* USEDIN(pd, 'BUILDING_DESIGN_SCHEMA.'+
                                    'SHAPE_ASPECT.OF_SHAPE') |
           (('BUILDING_DESIGN_SCHEMA.POSITIVE_COMPONENT' IN TYPEOF (sa))
           AND
            (sa.description = 'main component'))
         )) = 1
       )) = 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **building_element** shall be the related_product_definition in exactly one **product_-definition_relationship** where the relating_product_definition is a **building_section**.

WR2: A **building_element** shall be assigned exactly one name in a **building_item_identification_-assignment**.

WR3: A **building_element** shall have a **product_definition_shape** that is referenced by exactly one **positive_component** with a description of 'main component'.

5.2.2.1.18 building_element_assembly

A **building_element_assembly** is a **product_definition** that is a collection of building elements or element assemblies that are assembled together into a larger object.

EXPRESS specification:

```
*)
ENTITY building_element_assembly
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pd <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.'+
       'PROPERTY_DEFINITION.DEFINITION') |
       'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE'
       IN TYPEOF (pd))) = 0;
  WR2: SIZEOF (QUERY (acu <* QUERY (pdr <* USEDIN (SELF,
       'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_RELATIONSHIP.' +
       'RELATING_PRODUCT_DEFINITION') |
       'BUILDING_DESIGN_SCHEMA.ASSEMBLY_COMPONENT_USAGE' IN
       TYPEOF (pdr)) |
       SIZEOF (TYPEOF (acu.related_product_definition) *
       ['BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.SPACE_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.STRUCTURE_ENCLOSURE_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT_ASSEMBLY']) = 1)) >= 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **building_element_assembly** shall not have a **product_definition_shape**.

WR2: A **building_element_assembly** shall be the relating_product_definition in at least one **assembly_component_usage** where the related_product_definition is a **building_element**, **fixture_-equipment_element**, **service_element**, **space_element**, **structure_enclosure_element**, or another **building_element_assembly**.

5.2.2.1.19 building_element_group

A **building_element_group** is a **product_definition** that is a logical collection of building elements or element groups.

EXPRESS specification:

```
*)
ENTITY building_element_group
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pd <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.'+
       'PROPERTY_DEFINITION.DEFINITION') |
       'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE'
       IN TYPEOF (pd)) ) = 0;
  WR2: SIZEOF (QUERY (pdr <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.'+
       'PRODUCT_DEFINITION_RELATIONSHIP.RELATING_PRODUCT_DEFINITION') |
       SIZEOF (TYPEOF (pdr.related_product_definition) *
       ['BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.SPACE_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.STRUCTURE_ENCLOSURE_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT_GROUP']) = 1)) >= 1;
```

```
END_ENTITY;
(*
```

Formal propositions:

WR1: A **building_element_group** shall not have a **product_definition_shape**.

WR2: A **building_element_group** shall be the relating_product_definition in at least one **product_-definition_relationship** where the related_product_definition is a **building_element**, **fixture_-equipment_element**, **service_element**, **space_element**, **structure_enclosure_element**, or another **building_element_group**.

5.2.2.1.20 building_item_identification_assignment

A **building_item_identification_assignment** is a **name_assignment** for a set of **identified_item** instances.

EXPRESS specification:

```
*)
ENTITY building_item_identification_assignment
  SUBTYPE OF (name_assignment);
  item : identified_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the **building_element**s, **fixture_equipment_element**s, **service_element**s, **space_element**s, **structure_enclosure_element**s, **building_level**s, **building_section**s, **positive_component**s, and **negative_component**s to which a name is assigned.

5.2.2.1.21 building_level

A **building_level** is a **product_definition** that defines a part of a building that consists of one or more areas accessible through primarily horizontal access paths.

EXPRESS specification:

```
*)
ENTITY building_level
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION') |
       'BUILDING_DESIGN_SCHEMA.BUILDING_SECTION' IN
              TYPEOF (pdr.relating_product_definition))) = 1;
  WR2: SIZEOF (USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) <= 1;
  WR3: SIZEOF (QUERY (pd <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.'+
       'PROPERTY_DEFINITION.DEFINITION') |
       NOT (SIZEOF (QUERY (pdr <* USEDIN (pd, 'BUILDING_DESIGN_SCHEMA.' +
       'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') |
       NOT (SIZEOF (TYPEOF (pdr.used_representation) *
```

```
       ['BUILDING_DESIGN_SCHEMA.' +
       'ADVANCED_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ELEMENTARY_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'FACETED_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'GROUND_FACE_SPACE_BOUNDARY_SHAPE_REPRESENTATION']) = 1))) = 0)))
       = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **building_level** shall be the related_product_definition in exactly one **product_definition_-relationship** where the relating_product_definition is a **building_section**.

WR2: A **building_level** shall be assigned at most one name in a **building_item_identification_-assignment**.

WR3: A **building_level** shall only be represented by an **advanced_space_boundary_shape_-representation**, **elementary_space_boundary_shape_representation**, **faceted_space_boundary_-shape_representation**, or **ground_face_space_boundary_shape_representation**.

5.2.2.1.22 building_section

A **building_section** is a **product_definition** that is a logical segment of a building. Logical segments of a building are defined based on local coordinate systems for functional or definitional convenience.

EXPRESS specification:

```
*)
ENTITY building_section
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.'+
       'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION') |
       'BUILDING_DESIGN_SCHEMA.BUILDING' IN
       TYPEOF (pdr.relating_product_definition))) = 1;
  WR2: SIZEOF (USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
  WR3: SIZEOF (QUERY (pdr <* USEDIN (SELF,'BUILDING_DESIGN_SCHEMA.' +
       'PRODUCT_DEFINITION_RELATIONSHIP.RELATING_PRODUCT_DEFINITION') |
       SIZEOF (TYPEOF (pdr.related_product_definition) *
       ['BUILDING_DESIGN_SCHEMA.BUILDING_LEVEL',
       'BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.SPACE_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.STRUCTURE_ENCLOSURE_ELEMENT']) = 1)) >= 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **building_section** shall participate in exactly one **product_definition_relationship** as the related_product_definition where the relating_product_definition is a **building**.

WR2: A **building_section** shall be assigned exactly one name in a **building_item_identification_-assignment**.

WR3: A **building_section** shall participate in at least one **product_definition_relationship** as the relating_product_definition where the related_product_definition shall be a **building_level**, **building_-element**, **fixture_equipment_element**, **service_element**, **space_element**, or **structure_enclosure_-element**.

5.2.2.1.23 change

A **change** is an **action_assignment** that assigns an action to an **identified_item** or a **resulting_item**.

EXPRESS specification:

```
*)
ENTITY change
  SUBTYPE OF (action_assignment);
  items : SET[1:?] OF building_design_change_item;
END_ENTITY;
(*
```

Attribute definitions:

items: the **identified_item**s and **resulting_item**s to which an action request is assigned.

5.2.2.1.24 classification_table

A **classification_table** is a **document_usage_constraint** that specifies the contents of a table as a classification of building elements or components.

EXPRESS specification:

```
*)
ENTITY classification_table
  SUBTYPE OF (document_usage_constraint);
WHERE
  WR1: SIZEOF (QUERY (bddr <* QUERY (dr <* USEDIN (SELF.source,
       'BUILDING_DESIGN_SCHEMA.DOCUMENT_REFERENCE.ASSIGNED_DOCUMENT') |
       'BUILDING_DESIGN_SCHEMA.BUILDING_DOCUMENT_REFERENCE'
       IN TYPEOF(dr)) |
       NOT (SIZEOF (QUERY (item <* bddr.items |
       NOT (SIZEOF (TYPEOF (item) *
       ['BUILDING_DESIGN_SCHEMA.PRODUCT_CATEGORY',
       'BUILDING_DESIGN_SCHEMA.BUILDING_CLASSIFICATION_GROUP']) = 1)
       )) = 0))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **classification_table** shall reference a source **document** that is assigned to a **product_-category** or **building_component_classification_group**.

5.2.2.1.25 elementary_csg_shape_representation

An **elementary_csg_shape_representation** is a **shape_representation** that is composed of CSG primitives and boolean operators.

EXPRESS specification:

```
*)
ENTITY elementary_csg_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF.items |
       NOT ((SIZEOF (['BUILDING_DESIGN_SCHEMA.CSG_SOLID',
       'BUILDING_DESIGN_SCHEMA.EXTRUDED_AREA_SOLID',
       'BUILDING_DESIGN_SCHEMA.REVOLVED_AREA_SOLID',
       'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF (item)) = 1) OR
       (item.name = 'reference curve')))) = 0;
  WR2: (SIZEOF (QUERY (item <* SELF.items |
       SIZEOF (['BUILDING_DESIGN_SCHEMA.CSG_SOLID',
       'BUILDING_DESIGN_SCHEMA.EXTRUDED_AREA_SOLID',
       'BUILDING_DESIGN_SCHEMA.REVOLVED_AREA_SOLID',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF (item)) = 1)) >= 1)
       OR ({2 <= SIZEOF (QUERY (it <* SELF.items |
       it.name = 'reference curve')) <= 3});
  WR3: SIZEOF (QUERY (item <* SELF.items |
       ('BUILDING_DESIGN_SCHEMA.CSG_SOLID' IN TYPEOF (item)) AND
       (NOT (valid_elementary_csg_tree
              (item\csg_solid.tree_root_expression))))) = 0;
  WR4: SIZEOF (QUERY (mi <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF (item)) |
       NOT ('BUILDING_DESIGN_SCHEMA.ELEMENTARY_CSG_SHAPE_REPRESENTATION' IN
       TYPEOF(mi\mapped_item.mapping_source.mapped_representation)))) = 0;
  WR5: SIZEOF (QUERY (it <* SELF.items |
       (it.name = 'reference curve') AND
       (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF (it)) OR
       (('BUILDING_DESIGN_SCHEMA_TRIMMED_CURVE' IN TYPEOF (it)) AND
       ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN
       TYPEOF (it\trimmed_curve.basis_curve)))))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: Each item of an **elementary_csg_shape_representation** shall be either a **csg_solid**, **extruded_-area_solid**, **revolved_area_solid**, **axis2_placement_3d**, or **mapped_item** or has a name of 'reference curve'.

WR2: An **elementary_csg_shape_representation** shall have at least one **representation_item** instance in its set of items that is of type **csg_solid**, **extruded_area_solid**, **revolved_area_solid**, or **mapped_item**, or between two and three **representation_item**s in the set of items shall have a name of 'reference curve'.

WR3: An **elementary_csg_shape_representation** shall be comprised of the proper CSG tree elements.

WR4: For each **mapped_item** in an **elementary_csg_shape_representation**, the source of the **mapped_item** shall be an **elementary_csg_shape_representation**.

WR5: Each **representation_item** with the name of 'reference curve' in the set of items of an **elementary_csg_shape_representation** shall be either a **polyline** or a **trimmed_curve** with a basis_curve that is a **circle**.

5.2.2.1.26 elementary_face_with_thickness_shape_representation

An **elementary_face_with_thickness_shape_representation** is a solid model that is defined by a **face** with underlying geometry that is comprised of analytic surfaces bounded by quadric curves with a length that defines the thickness.

EXPRESS specification:

```
*)
ENTITY elementary_face_with_thickness_shape_representation
  SUBTYPE OF (shape_representation, solid_model);
WHERE
  WR1: SIZEOF (SELF.items) = 2;
  WR2: SIZEOF (QUERY (item <* SELF.items |
       (SIZEOF (['BUILDING_DESIGN_SCHEMA.MEASURE_REPRESENTATION_ITEM',
       'BUILDING_DESIGN_SCHEMA.LENGTH_MEASURE_WITH_UNIT'] *
       TYPEOF (item)) = 2) AND (item.name = 'thickness'))) = 1;
  WR3: SIZEOF (QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF (item))) = 1;
  WR4: SIZEOF (QUERY (item <* SELF.items |
       (NOT ('BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF (item))) OR
       ('BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN
       TYPEOF (item\face_surface.face_geometry)))) = 0;
  WR5: SIZEOF (QUERY (item <* SELF.items |
       ('BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF (item)) AND
       (NOT (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* item\face.bounds |
       'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
       NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
       NOT ('BUILDING_DESIGN_SCHEMA.EDGE_CURVE' IN
       TYPEOF(oe.edge_element)))) = 0))) = 0)))) = 0;
  WR6: SIZEOF (QUERY (item <* SELF.items |
       ('BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF (item)) AND
       (NOT (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* item\face.bounds |
       'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
       NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
       NOT (SIZEOF (['BUILDING_DESIGN_SCHEMA.LINE',
       'BUILDING_DESIGN_SCHEMA.CONIC',
       'BUILDING_DESIGN_SCHEMA.POLYLINE',
       'BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE'] *
       TYPEOF (oe.edge_element\edge_curve.edge_geometry)) = 1)))
       = 0))) = 0)))) = 0;
  WR7: SIZEOF (QUERY (item <* SELF.items |
       ('BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF (item)) AND
       (NOT (SIZEOF (QUERY (vlp_fbnds <* QUERY (bnds <* item\face.bounds |
       'BUILDING_DESIGN_SCHEMA.VERTEX_LOOP' IN TYPEOF (bnds.bound)) |
       NOT (('BUILDING_DESIGN_SCHEMA.VERTEX_POINT' IN
       TYPEOF(vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex)) AND
       ('BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT' IN
       TYPEOF(vlp_fbnds\face_bound.bound\vertex_loop.
       loop_vertex\vertex_point.vertex_geometry))))) =  0)))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: An **elementary_face_with_thickness_shape_representation** shall contain exactly two **representation_item**s in its set of items.

WR2: Exactly one **representation_item** in the set of items of an **elementary_face_with_thickness_-shape_representation** shall be a **measure_representation_item** and a **length_measure_with_unit** with a name of 'thickness'.

WR3: Exactly one **representation_item** in the set of items of an **elementary_face_with_thickness_-shape_representation** shall be a **face_surface**.

WR4: If the **representation_item** in the set of items of an **elementary_face_with_thickness_shape_-representation** is a **face_surface**, the face_geometry shall be an **elementary_surface**.

WR5: If the **representation_item** in the set of items of an **elementary_face_with_thickness_shape_-representation** is a **face_surface**, the **edge**s used to define the boundaries shall all be of type **edge_-curve**.

WR6: If the **representation_item** in the set of items of an **elementary_face_with_thickness_shape_-representation** is a **face_surface**, each curve used to define the face bounds shall be either a **conic**, a **line**, a **polyline**, or a **b_spline_curve**.

WR7: If the **representation_item** in the set of items of an **elementary_face_with_thickness_shape_-representation** is a **face_surface**, any **vertex_loop** used to define a face bound shall reference a **vertex_point** with the geometry defined by a **cartesian_point**.

5.2.2.1.27 elementary_geometric_shape_representation

An **elementary_geometric_shape_representation** is a boundary representation solid model **shape_-representation** that consists of simple quadric curves and surface elements.

EXPRESS specification:

```
*)
ENTITY elementary_geometric_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF.items |
       NOT ((SIZEOF (['BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP',
       'BUILDING_DESIGN_SCHEMA.FACETED_BREP',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
       'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D'] *
       TYPEOF (item)) = 1) OR
       (item.name = 'reference curve')))) = 0;
  WR2: (SIZEOF (QUERY (item <* SELF.items |
       SIZEOF (['BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF (item)) =1 )) >= 1)
       OR ({2 <= SIZEOF (QUERY (it <* SELF.items |
       it.name = 'reference curve')) <= 3});
  WR3: SIZEOF (QUERY (msb <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP' IN TYPEOF (item)) |
       NOT (SIZEOF (QUERY (csh <*
       msb_shells (msb) |
```

```
      NOT (SIZEOF (QUERY (fcs <* csh.cfs_faces |
      NOT ('BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF (fcs)))) = 0
      ))) = 0))) = 0;
  WR4: SIZEOF (QUERY (msb <* QUERY (item <* SELF.items |
      'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP' IN TYPEOF (item)) |
      NOT (SIZEOF (QUERY (csh <*
      msb_shells (msb) |
      NOT (SIZEOF (QUERY (fcs <* csh\connected_face_set.cfs_faces |
      NOT (('BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN
      TYPEOF (fcs\face_surface.face_geometry))))) = 0))) = 0))) = 0;
  WR5: SIZEOF (QUERY (msb <* QUERY (item <* SELF.items |
      'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP' IN TYPEOF (item)) |
      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 |
      'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
      NOT ('BUILDING_DESIGN_SCHEMA.EDGE_CURVE' IN
      TYPEOF (oe.edge_element)))) = 0))) = 0))) = 0))) = 0))) = 0;
  WR6: SIZEOF (QUERY (msb <* QUERY (item <* SELF.items |
      'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP' IN TYPEOF (item)) |
      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 |
      'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
      NOT (SIZEOF (['BUILDING_DESIGN_SCHEMA.LINE',
      'BUILDING_DESIGN_SCHEMA.CONIC',
      'BUILDING_DESIGN_SCHEMA.POLYLINE',
      'BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE'] *
      TYPEOF (oe.edge_element\edge_curve.edge_geometry)) = 1)))
      = 0))) = 0))) = 0))) = 0))) = 0;
  WR7: SIZEOF (QUERY (msb <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP' IN TYPEOF (item)) |
       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 |
       'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
       NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
       NOT(('BUILDING_DESIGN_SCHEMA.VERTEX_POINT'
       IN TYPEOF (oe.edge_start))
       AND ('BUILDING_DESIGN_SCHEMA.VERTEX_POINT'
       IN TYPEOF (oe.edge_end)))))
       = 0))) = 0))) = 0))) = 0))) = 0;
  WR8: SIZEOF (QUERY (msb <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP' IN TYPEOF (item)) |
       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 |
       'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
       'BUILDING_DESIGN_SCHEMA.ORIENTED_PATH'
       IN TYPEOF (elp_fbnds.bound)))
       = 0))) = 0))) = 0))) = 0;
  WR9: SIZEOF (QUERY (msb <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP' IN TYPEOF(item)) |
       'BUILDING_DESIGN_SCHEMA.ORIENTED_CLOSED_SHELL' IN TYPEOF
       (msb\manifold_solid_brep.outer))) = 0;
 WR10: SIZEOF (QUERY (brv <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.BREP_WITH_VOIDS' IN TYPEOF (item)) |
       NOT (SIZEOF (QUERY (csh <* brv\brep_with_voids.voids |
```

```
       csh\oriented_closed_shell.orientation)) = 0))) = 0;
  WR11: SIZEOF (QUERY (mi <*  QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF (item)) |
       NOT ('BUILDING_DESIGN_SCHEMA.' +
       'ELEMENTARY_GEOMETRIC_SHAPE_REPRESENTATION'
       IN TYPEOF (mi\mapped_item.mapping_source.
       mapped_representation)))) = 0;
  WR12: SIZEOF (QUERY (msb <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP' IN TYPEOF (item)) |
       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 |
       'BUILDING_DESIGN_SCHEMA.VERTEX_LOOP' IN TYPEOF (bnds.bound)) |
       NOT(('BUILDING_DESIGN_SCHEMA.VERTEX_POINT' IN
       TYPEOF (vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex)) AND
       ('BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT' IN
       TYPEOF (vlp_fbnds\face_bound.bound\vertex_loop.
       loop_vertex\vertex_point.vertex_geometry))
       ))) = 0))) = 0))) = 0))) = 0;
 WR13: SIZEOF (QUERY (it <* SELF.items |
       (it.name = 'reference curve') AND
       (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF (it)) OR
       (('BUILDING_DESIGN_SCHEMA_TRIMMED_CURVE' IN TYPEOF (it)) AND
       ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN
       TYPEOF (it\trimmed_curve.basis_curve)))))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: Each item of an **elementary_geometric_shape_representation** shall be a **manifold_solid_brep**, **mapped_item**, or **axis2_placement_3d** or has a name of 'reference curve'.

NOTE 1 - The use of **faceted_brep** is excluded by this rule since an instance of **faceted_brep** would also be of type **manifold_solid_brep**.

WR2: At least one **representation_item** in the set of items shall be a **manifold_solid_brep** entity or a **mapped_item**, or between two and three **representation_item**s in the set of items shall have a name of 'reference curve'.

WR3: All faces used in constructing a **manifold_solid_brep** shall be of type **face_surface**.

WR4: For each **manifold_solid_brep** in the set of items the associated surface for each **face** shall be an **elementary_surface**.

WR5: For each **manifold_solid_brep** in the set of items the **edge**s used to define the boundaries shall all be of type **edge_curve**.

WR6: For each **manifold_solid_brep** in the set of items each curve used to define the face bounds shall be either a **conic**, a **line**, a **polyline**, or a **b_spline_curve**.

WR7: For each **manifold_solid_brep** in the set of items the **edge**s used to define the boundaries shall all be trimmed by vertices of type **vertex_point**.

WR8: For each **manifold_solid_brep** in the set of items no **loop** used to define a face bound shall be of the oriented subtype.

WR9: For each **manifold_solid_brep** in the set of items the **outer_shell** shall not be of the oriented subtype.

WR10: If a **brep_with_voids** is included in the set of items then each shell in the voids set shall be an **oriented_closed_shell** with orientation value FALSE.

WR11: If a **mapped_item** is included in the set of items then the mapped_representation of the mapping_source shall be an **elementary_geometric_shape_representation**.

NOTE 2 - If a **cartesian_transformation_operator_3d** is included as the mapping_target of the **mapped_item** with an **axis2_placement_3d** that corresponds to the original coordinate system as the mapping_origin of the **representation_map** then the resulting **mapped_item** is a transformed copy of the **elementary_geometric_shape_representation**. The precise definition of the transformation, including translation, rotation, scaling and, if appropriate, mirroring, is given by the transformation operator.

WR12: For each **manifold_solid_brep** in the set of items any **vertex_loop** used to define a face bound shall reference a **vertex_point** with the geometry defined by a **cartesian_point**.

WR13: Each **representation_item** with the name of 'reference curve' in the set of items of an **elementary_geometric_shape_representation** shall be either a **polyline** or a **trimmed_curve** with a basis_curve that is a **circle**.

5.2.2.1.28 elementary_space_boundary_shape_representation

An **elementary_space_boundary_shape_representation** is a **shape_representation** that represents the shape of a space in a building. This shape or aspect of the shape shall be represented using elementary **closed_shell**s.

EXPRESS specification:

```
*)
ENTITY elementary_space_boundary_shape_representation
  SUBTYPE OF(shape_representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF.items |
       NOT ((SIZEOF (['BUILDING_DESIGN_SCHEMA.CLOSED_SHELL',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
       'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D']
       * TYPEOF (item)) = 1) OR
       (item.name = 'reference curve')))) = 0;
  WR2: (SIZEOF (QUERY (item <* SELF.items |
       SIZEOF (['BUILDING_DESIGN_SCHEMA.CLOSED_SHELL',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF (item)) = 1)) = 1)
       OR ({2 <= SIZEOF (QUERY (it <* SELF.items |
       it.name = 'reference curve')) <= 3});
  WR3: SIZEOF (QUERY (mi <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF (item)) |
       NOT ('BUILDING_DESIGN_SCHEMA.' +
       'ELEMENTARY_SPACE_BOUNDARY_SHAPE_REPRESENTATION' IN
       TYPEOF (mi\mapped_item.mapping_source.mapped_representation)))) = 0;
  WR4: SIZEOF (QUERY (csh <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF (item)) |
```

```
      NOT (SIZEOF (QUERY (fcs <* csh\closed_shell.cfs_faces |
      NOT ('BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF (fcs)))) = 0)))
      = 0;
  WR5: SIZEOF (QUERY (csh <* QUERY (item <* SELF.items |
      'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF (item)) |
      NOT (SIZEOF (QUERY (fcs <* csh\closed_shell.cfs_faces |
      NOT ('BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN
      TYPEOF (fcs\face_surface.face_geometry)))) = 0))) = 0;
  WR6: SIZEOF (QUERY (csh <* QUERY (item <* SELF.items |
      'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF (item)) |
      NOT (SIZEOF (QUERY (fcs <* csh\closed_shell.cfs_faces |
      NOT (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
      'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
      NOT ('BUILDING_DESIGN_SCHEMA.EDGE_CURVE' IN
      TYPEOF (oe.edge_element)))) = 0))) = 0))) = 0))) = 0;
  WR7: SIZEOF (QUERY (csh <* QUERY (item <* SELF.items |
      'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF (item)) |
      NOT (SIZEOF (QUERY (fcs <* csh\closed_shell.cfs_faces |
      NOT (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
      'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
      NOT (SIZEOF (['BUILDING_DESIGN_SCHEMA.LINE',
      'BUILDING_DESIGN_SCHEMA.CONIC',
      'BUILDING_DESIGN_SCHEMA.POLYLINE',
      'BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE'] *
      TYPEOF (oe.edge_element\edge_curve.edge_geometry)) = 1)))
      = 0))) = 0))) = 0))) = 0;
  WR8: SIZEOF (QUERY (csh <* QUERY (item <* SELF.items |
      'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF (item)) |
      NOT (SIZEOF (QUERY (fcs <* csh\closed_shell.cfs_faces |
      NOT (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
      'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
      NOT(('BUILDING_DESIGN_SCHEMA.VERTEX_POINT'
        IN TYPEOF (oe.edge_start))
      AND ('BUILDING_DESIGN_SCHEMA.VERTEX_POINT'
        IN TYPEOF (oe.edge_end)))))
      = 0))) = 0))) = 0))) = 0;
  WR9: SIZEOF (QUERY (csh <* QUERY (item <* SELF.items |
      'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF (item)) |
      NOT (SIZEOF (QUERY (fcs <* csh\connected_face_set.cfs_faces |
      NOT (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
      'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
      'BUILDING_DESIGN_SCHEMA.ORIENTED_PATH' IN TYPEOF (elp_fbnds.bound)))
      = 0))) = 0))) = 0;
 WR10: SIZEOF (QUERY (csh <* QUERY (item <* SELF.items |
      'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF (item)) |
      NOT (SIZEOF (QUERY (fcs <* csh\closed_shell.cfs_faces |
      NOT (SIZEOF (QUERY (vlp_fbnds <* QUERY (bnds <* fcs.bounds |
      'BUILDING_DESIGN_SCHEMA.VERTEX_LOOP' IN TYPEOF (bnds.bound)) |
      NOT (('BUILDING_DESIGN_SCHEMA.VERTEX_POINT' IN
      TYPEOF (vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex)) AND
      ('BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT' IN
      TYPEOF (vlp_fbnds\face_bound.bound\vertex_loop.
      loop_vertex\vertex_point.vertex_geometry))
      ))) = 0))) = 0))) = 0;
 WR11: SIZEOF (QUERY (it <* SELF.items |
      (it.name = 'reference curve') AND
      (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF (it)) OR
      (('BUILDING_DESIGN_SCHEMA_TRIMMED_CURVE' IN TYPEOF (it)) AND
      ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN
      TYPEOF (it\trimmed_curve.basis_curve)))))))) = 0;
END_ENTITY;
```

(*

Formal propositions:

WR1: Each item of an **elementary_space_boundary_shape_representation** shall be a **closed_shell**, **mapped_item**, or **axis2_placement_3d** or has a name of ‘reference curve’.

WR2: Exactly one of the items of an **elementary_space_boundary_shape_representation** shall be a **closed_shell** or **mapped_item**, or between two and three **representation_item**s in the set of items shall have a name of ‘reference curve’.

WR3: For each **mapped_item** in an **elementary_space_boundary_shape_representation**, the source of the **mapped_item** shall be an **elementary_space_boundary_shape_representation**.

WR4: All faces used in constructing the **closed_shell**s in an **elementary_space_boundary_shape_-representation** shall be of type **face_surface**.

WR5: For each **closed_shell** in the set of items of an **elementary_space_boundary_shape_-representation** the associated surface for each **face_surface** shall be an **elementary_surface**.

WR6: For each **closed_shell** in the set of items of an **elementary_space_boundary_shape_-representation** the **edge**s used to define the boundaries shall all be of type **edge_curve**.

WR7: For each **closed_shell** in the set of items of an **elementary_space_boundary_shape_-representation** each curve used to define the face bounds shall be either a **conic**, **line**, **polyline**, or **b_spline_curve**.

WR8: For each **closed_shell** in the set of items of an **elementary_space_boundary_shape_-representation** the **edge**s used to define the boundaries shall all be trimmed by vertices of type **vertex_point**.

WR9: For each **closed_shell** in the set of items of an **elementary_space_boundary_shape_-representation** no **loop** used to define a face bound shall be of the oriented subtype.

WR10: For each **closed_shell** in the set of items of an **elementary_space_boundary_shape_-representation** any **vertex_loop** used to define a face bound shall reference a **vertex_point** with the geometry defined by a **cartesian_point**.

WR11: Each **representation_item** with the name of ‘reference curve’ in the set of items of an **elementary_space_boundary_shape_representation** shall be either a **polyline** or a **trimmed_curve** with a basis_curve that is a **circle**.

5.2.2.1.29 elementary_wire_shape_representation

An **elementary_wire_shape_representation** is a shape that is defined by a single quadric curve or a **composite_curve** that is comprised of simple quadric curves.

EXPRESS specification:

```
*)
ENTITY elementary_wire_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF.items |
       NOT ((SIZEOF (['BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
       'BUILDING_DESIGN_SCHEMA.CIRCLE',
       'BUILDING_DESIGN_SCHEMA.ELLIPSE',
       'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
       'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D']
       * TYPEOF (item)) = 1) OR
       (item.name = 'reference curve')))) = 0;
  WR2: (SIZEOF (QUERY (item <* SELF.items |
       SIZEOF (['BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
       'BUILDING_DESIGN_SCHEMA.CIRCLE',
       'BUILDING_DESIGN_SCHEMA.ELLIPSE',
       'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF (item)) = 1)) = 1)
       OR ({2 <= SIZEOF (QUERY (it <* SELF.items |
       it.name = 'reference curve')) <= 3});
  WR3: SIZEOF (QUERY (item <* SELF.items |
       (SIZEOF (['BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
       'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE'] * TYPEOF (item)) = 1) AND
       (NOT (valid_elementary_wire_composition (item))))) = 0;
  WR4: SIZEOF (QUERY (mi <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF (item)) |
       NOT ('BUILDING_DESIGN_SCHEMA.' +
       'ELEMENTARY_WIRE_SHAPE_REPRESENTATION' IN
       TYPEOF (mi\mapped_item.mapping_source.mapped_representation)))) = 0;
  WR5: SIZEOF (QUERY (it <* SELF.items |
       (it.name = 'reference curve') AND
       (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF (it)) OR
       (('BUILDING_DESIGN_SCHEMA_TRIMMED_CURVE' IN TYPEOF (it)) AND
       ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN
       TYPEOF (it\trimmed_curve.basis_curve)))))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: Each item of an **elementary_wire_shape_representation** shall be a **composite_curve**, **circle**, **ellipse**, **trimmed_curve**, **mapped_item**, or **axis2_placement_3d** or has a name of 'reference curve'.

WR2: Exactly one of the items of an **elementary_wire_shape_representation** shall be a **composite_-curve**, **circle**, **ellipse**, **trimmed_curve**, or **mapped_item**, or between two and three **representation_item**s in the set of items shall have a name of 'reference curve'.

WR3: If a **representation_item** in the set of items is a **trimmed_curve** or **composite_curve** it shall reference valid elementary curves.

WR4: For each **mapped_item** in an **elementary_wire_shape_representation**, the source of the **mapped_item** shall be an **elementary_wire_shape_representation**.

WR5: Each **representation_item** with the name of ‘reference curve’ in the set of items of an **elementary_wire_shape_representation** shall be either a **polyline** or a **trimmed_curve** with a basis_curve that is a **circle**.

5.2.2.1.30 faceted_csg_shape_representation

A **faceted_csg_shape_representation** is a **shape_representation** that consists of boolean operators and CSG primitives that are composed of planar and linear elements.

EXPRESS specification:

```
*)
ENTITY faceted_csg_shape_representation
  SUBTYPE OF(shape_representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF.items |
       NOT ((SIZEOF (['BUILDING_DESIGN_SCHEMA.CSG_SOLID',
       'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] *
       TYPEOF (item)) = 1) OR
       (item.name = 'reference curve')))) = 0;
  WR2: (SIZEOF (QUERY (item <* SELF.items |
       NOT (SIZEOF (['BUILDING_DESIGN_SCHEMA.CSG_SOLID',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] *
       TYPEOF (item)) = 1))) >= 1)
       OR ({2 <= SIZEOF (QUERY (it <* SELF.items |
       it.name = 'reference curve')) <= 3});
  WR3: SIZEOF (QUERY (item <* SELF.items |
       ('BUILDING_DESIGN_SCHEMA.CSG_SOLID' IN TYPEOF (item)) AND
       (NOT (valid_faceted_csg_tree
       (item\csg_solid.tree_root_expression))))) = 0;
  WR4: SIZEOF (QUERY (mi <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF (item)) |
       NOT ('BUILDING_DESIGN_SCHEMA.FACETED_CSG_SHAPE_REPRESENTATION' IN
       TYPEOF (mi\mapped_item.mapping_source.mapped_representation)))) = 0;
  WR5: SIZEOF (QUERY (it <* SELF.items |
       (it.name = 'reference curve') AND
       (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF (it)) OR
       (('BUILDING_DESIGN_SCHEMA_TRIMMED_CURVE' IN TYPEOF (it)) AND
       ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN
       TYPEOF (it\trimmed_curve.basis_curve)))))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: Each item of a **faceted_csg_shape_representation** shall be a **csg_solid**, **axis2_placement_3d**, or **mapped_item** or has a name of ‘reference curve’.

WR2: A **faceted_csg_shape_representation** shall have at least one **representation_item** instance in its set of items that is of type **csg_solid** or **mapped_item**, or between two and three **representation_item**s in the set of items shall have a name of ‘reference curve’.

WR3: A **faceted_csg_shape_representation** shall be comprised of the proper CSG tree elements.

WR4: For each **mapped_item** in a **faceted_csg_shape_representation**, the source of the **mapped_item** shall be a **faceted_csg_shape_representation**.

WR5: Each **representation_item** with the name of 'reference curve' in the set of items of a **faceted_csg_shape_representation** shall be either a **polyline** or a **trimmed_curve** with a basis_curve that is a **circle**.

5.2.2.1.31 faceted_face_with_thickness_shape_representation

A **faceted_face_with_thickness_shape_representation** is a solid model that is defined by a **face** that has a **poly_loop** defining it's topology and a length defining the thickness. If the **face** has underlying geometry, that geometry shall be defined by a **plane**.

EXPRESS specification:

```
*)
ENTITY faceted_face_with_thickness_shape_representation
  SUBTYPE OF (shape_representation, solid_model);
WHERE
  WR1: SIZEOF (SELF.items) = 2;
  WR2: SIZEOF (QUERY (item <* SELF.items |
       (SIZEOF (['BUILDING_DESIGN_SCHEMA.MEASURE_REPRESENTATION_ITEM',
       'BUILDING_DESIGN_SCHEMA.LENGTH_MEASURE_WITH_UNIT'] *
       TYPEOF (item)) = 2) AND (item.name = 'thickness'))) = 1;
  WR3: SIZEOF (QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.FACE' IN TYPEOF (item))) = 1;
  WR4: SIZEOF (QUERY (item <* SELF.items |
       ('BUILDING_DESIGN_SCHEMA.FACE' IN TYPEOF (item)) AND
       (NOT (SIZEOF (QUERY (bnds <* item\face.bounds |
       NOT ('BUILDING_DESIGN_SCHEMA.POLY_LOOP' IN TYPEOF (bnds.bound))))
       = 0)))) = 0;
  WR5: SIZEOF (QUERY (item <* SELF.items |
       (NOT ('BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF (item))) OR
       ('BUILDING_DESIGN_SCHEMA.PLANE' IN
       TYPEOF (item\face_surface.face_geometry)))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **faceted_face_with_thickness_shape_representation** shall contain exactly two **representation_item**s in its set of items.

WR2: Exactly one **representation_item** in the set of items of a **faceted_face_with_thickness_shape_-representation** shall be a **measure_representation_item** and a **length_measure_with_unit** with a name of 'thickness'.

WR3: Exactly one **representation_item** in the set of items of a **faceted_face_with_thickness_shape_-representation** shall be a **face**.

WR4: If the **representation_item** in the set of items of a **faceted_face_with_thickness_shape_-representation** is a **face**, the boundary shall be defined by a **poly_loop**.

WR5: If the **representation_item** in the set of items of a **faceted_face_with_thickness_shape_-representation** is a **face_surface**, the underlying geometry shall be defined by a **plane**.

5.2.2.1.32 faceted_geometric_shape_representation

A **faceted_geometric_shape_representation** is a **shape_representation** that consists of linear and planar geometric elements.

EXPRESS specification:

```
*)
ENTITY faceted_geometric_shape_representation
  SUBTYPE OF(shape_representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF.items |
       NOT ((SIZEOF (['BUILDING_DESIGN_SCHEMA.FACETED_BREP',
       'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] *
       TYPEOF (item)) = 1) OR
       (item.name = 'reference curve')))) = 0;
  WR2: (SIZEOF (QUERY (item <* SELF.items |
       SIZEOF (['BUILDING_DESIGN_SCHEMA.FACETED_BREP',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] *
       TYPEOF (item)) = 1)) >= 1)
       OR ({2 <= SIZEOF (QUERY (it <* SELF.items |
       it.name = 'reference curve')) <= 3});
  WR3: SIZEOF (QUERY (msb <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.FACETED_BREP' IN TYPEOF (item)) |
       NOT (SIZEOF (QUERY (csh <*
       msb_shells (msb) |
       NOT (SIZEOF (QUERY (fcs <* csh\connected_face_set.cfs_faces |
       NOT (SIZEOF (QUERY (bnds <* fcs.bounds |
       NOT ('BUILDING_DESIGN_SCHEMA.POLY_LOOP' IN TYPEOF (bnds.bound))))
       = 0))) = 0))) = 0))) = 0;
  WR4: SIZEOF (QUERY (msb <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.FACETED_BREP' IN TYPEOF (item)) |
       NOT (SIZEOF (QUERY (csh <*
       msb_shells (msb) |
       NOT (SIZEOF (QUERY (fcs <* csh.cfs_faces |
       ('BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF (fcs)) AND
       (NOT ('BUILDING_DESIGN_SCHEMA.PLANE' IN
       TYPEOF (fcs\face_surface.face_geometry))))) = 0))) = 0))) = 0;
  WR5: SIZEOF (QUERY (mi <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF (item)) |
       NOT ('BUILDING_DESIGN_SCHEMA.FACETED_GEOMETRIC_SHAPE_REPRESENTATION'
       IN TYPEOF (mi\mapped_item.mapping_source.mapped_representation))))
       = 0;
  WR6: SIZEOF (QUERY (it <* SELF.items |
       (it.name = 'reference curve') AND
       (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF (it)) OR
       (('BUILDING_DESIGN_SCHEMA_TRIMMED_CURVE' IN TYPEOF (it)) AND
       ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN
       TYPEOF (it\trimmed_curve.basis_curve)))))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: Each item of a **faceted_geometric_shape_representation** shall be a **faceted_brep**, **mapped_item**, or **axis2_placement_3d** or has a name of 'reference curve'.

WR2: At least one of the items of a **faceted_geometric_shape_representation** shall a **faceted_brep** or **mapped_item**, or between two and three **representation_item**s in the set of items shall have a name of 'reference curve'.

WR3: For each **faceted_brep** in the set of items of a **faceted_geometric_shape_representation**, a **poly_loop** shall be used to define a face bound.

WR4: For each **faceted_brep** in the set of items of a **faceted_geometric_shape_representation**, if the geometry for a face is specified, it shall be defined by a **plane**.

WR5: For each **mapped_item** in a **faceted_geometric_shape_representation**, the source of the **mapped_item** shall be a **faceted_geometric_shape_representation**.

WR6: Each **representation_item** with the name of 'reference curve' in the set of items of a **faceted_geometric_shape_representation** shall be either a **polyline** or a **trimmed_curve** with a basis_curve that is a **circle**.

5.2.2.1.33 faceted_space_boundary_shape_representation

A **faceted_space_boundary_shape_representation** is a **shape_representation** that represents the shape of a space in a building. This shape or aspect of shape shall be represented using faceted **closed_shell**s.

EXPRESS specification:

```
*)
ENTITY faceted_space_boundary_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF.items |
       NOT ((SIZEOF (['BUILDING_DESIGN_SCHEMA.CLOSED_SHELL',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
       'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D']
       * TYPEOF (item)) = 1) OR
       (item.name = 'reference curve')))) = 0;
  WR2: (SIZEOF (QUERY (item <* SELF.items |
       SIZEOF (['BUILDING_DESIGN_SCHEMA.CLOSED_SHELL',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF (item)) = 1)) = 1)
       OR ({2 <= SIZEOF (QUERY (it <* SELF.items |
       it.name = 'reference curve')) <= 3});
  WR3: SIZEOF (QUERY (csh <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF (item)) |
       NOT (SIZEOF (QUERY (fcs <* csh\connected_face_set.cfs_faces |
       NOT (SIZEOF (QUERY (bnds <* fcs.bounds |
       NOT ('BUILDING_DESIGN_SCHEMA.POLY_LOOP' IN TYPEOF (bnds.bound))))
       = 0))) = 0))) = 0;
  WR4: SIZEOF (QUERY (mi <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF (item)) |
       NOT ('BUILDING_DESIGN_SCHEMA.' +
       'FACETED_SPACE_BOUNDARY_SHAPE_REPRESENTATION' IN
       TYPEOF (mi\mapped_item.mapping_source.mapped_representation)))) = 0;
  WR5: SIZEOF (QUERY (it <* SELF.items |
       (it.name = 'reference curve') AND
       (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF (it)) OR
       (('BUILDING_DESIGN_SCHEMA_TRIMMED_CURVE' IN TYPEOF (it)) AND
       ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN
       TYPEOF (it\trimmed_curve.basis_curve)))))) = 0;
```

```
END_ENTITY;
(*
```

Formal propositions:

WR1: Each item of a **faceted_space_boundary_shape_representation** shall be a **closed_shell**, **mapped_item**, or **axis2_placement_3d** or has a name of 'reference curve'.

WR2: Exactly one of the items of a **faceted_space_boundary_shape_representation** shall be a **closed_shell** or **mapped_item**, or between two and three **representation_item**s in the set of items shall have a name of 'reference curve'.

WR3: For each **closed_shell** in the set of items of a **faceted_space_boundary_shape_representation**, a **poly_loop** shall be used to define a face bound.

WR4: For each **mapped_item** in a **faceted_space_boundary_shape_representation**, the source of the **mapped_item** shall be a **faceted_space_boundary_shape_representation**.

WR5: Each **representation_item** with the name of 'reference curve' in the set of items of a **faceted_geometric_shape_representation** shall be either a **polyline** or a **trimmed_curve** with a basis_curve that is a **circle**.

5.2.2.1.34 faceted_wire_shape_representation

A **faceted_wire_shape_representation** is a shape that is defined by a single quadric curve or a **composite_curve** that is comprised of simple quadric curves.

EXPRESS specification:

```
*)
ENTITY faceted_wire_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF.items |
       NOT ((SIZEOF (['BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
       'BUILDING_DESIGN_SCHEMA.POLYLINE',
       'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
       'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D']
       * TYPEOF (item)) = 1) OR
       (item.name = 'reference curve')))) = 0;
  WR2: (SIZEOF (QUERY (item <* SELF.items |
       SIZEOF (['BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
       'BUILDING_DESIGN_SCHEMA.POLYLINE',
       'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE',
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF (item)) = 1)) = 1)
       OR ({2 <= SIZEOF (QUERY (it <* SELF.items |
       it.name = 'reference curve')) <= 3});
  WR3: SIZEOF (QUERY (item <* SELF.items |
       (SIZEOF (['BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
       'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE'] * TYPEOF (item)) = 1) AND
       (NOT (valid_faceted_wire_composition (item))))) = 0;
  WR4: SIZEOF (QUERY (mi <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF (item)) |
       NOT ('BUILDING_DESIGN_SCHEMA.' +
       'FACETED_WIRE_SHAPE_REPRESENTATION' IN
```

```
      TYPEOF (mi\mapped_item.mapping_source.mapped_representation)))) = 0;
  WR5: SIZEOF (QUERY (it <* SELF.items |
      (it.name = 'reference curve') AND
      (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF (it)) OR
      (('BUILDING_DESIGN_SCHEMA_TRIMMED_CURVE' IN TYPEOF (it)) AND
      ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN
      TYPEOF (it\trimmed_curve.basis_curve)))))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: Each item of a **faceted_wire_shape_representation** shall be a **composite_curve**, **polyline**, **trimmed_curve**, **mapped_item**, or **axis2_placement_3d** or has a name of 'reference curve'.

WR2: Exactly one of the items of an **faceted_wire_shape_representation** shall be a **composite_-curve**, **polyline**, **trimmed_curve**, or **mapped_item**, or between two and three **representation_item**s in the set of items shall have a name of 'reference curve'.

WR3: If a **representation_item** in the set of items is a **trimmed_curve** or **composite_curve** it shall reference valid curves.

WR4: For each **mapped_item** in an **faceted_wire_shape_representation**, the source of the **mapped_-item** shall be an **faceted_wire_shape_representation**.

WR5: Each **representation_item** with the name of 'reference curve' in the set of items of a **faceted_geometric_shape_representation** shall be either a **polyline** or a **trimmed_curve** with a basis_curve that is a **circle**.

5.2.2.1.35 fixture_equipment_element

A **fixture_equipment_element** is a **product_definition** that defines a solid, tangible part of a building that is placed or installed in the building.

EXPRESS specification:

```
*)
ENTITY fixture_equipment_element
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN(SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION') |
       'BUILDING_DESIGN_SCHEMA.BUILDING_SECTION' IN
       TYPEOF (pdr.relating_product_definition))) = 1;
  WR2: SIZEOF (USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
  WR3: SIZEOF (QUERY (pds <* QUERY (pd <* USEDIN (SELF,
       'BUILDING_DESIGN_SCHEMA.PROPERTY_DEFINITION.DEFINITION') |
       'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN TYPEOF (pd)) |
       SIZEOF (QUERY (sa <* USEDIN (pds, 'BUILDING_DESIGN_SCHEMA.' +
       'SHAPE_ASPECT.OF_SHAPE') |
       ('BUILDING_DESIGN_SCHEMA.POSITIVE_COMPONENT' IN TYPEOF (sa))
       AND
       (sa.description = 'main component'))) = 1)) = 1;
END_ENTITY;
```

```
(*
```

Formal propositions:

WR1: A **fixture_equipment_element** shall be the related_product_definition in exactly one **product_-definition_relationship** where the relating_product_definition is a **building_section**.

WR2: A **fixture_equipment_element** shall be assigned exactly one name in a **building_item_-identification_assignment**.

WR3: A **fixture_equipment_element** shall have a **product_definition_shape** that is referenced by exactly one **positive_component** with a description of 'main component'.

5.2.2.1.36 ground_face_space_boundary_shape_representation

A **ground_face_space_boundary_shape_representation** is a solid model that is defined by a **face** with a **poly_loop** defining its topology and with a length that defines the thickness. If the **face** has underlying geometry, the geometry shall be defined by a **plane**.

EXPRESS specification:

```
*)
ENTITY ground_face_space_boundary_shape_representation
  SUBTYPE OF (shape_representation, solid_model);
WHERE
  WR1: SIZEOF (SELF.items) = 1;
  WR2: SIZEOF (QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.FACE' IN TYPEOF (item))) = 1;
  WR3: SIZEOF (QUERY (item <* SELF.items |
       NOT (SIZEOF (QUERY (bnds <* item\face.bounds |
       NOT ('BUILDING_DESIGN_SCHEMA.POLY_LOOP' IN TYPEOF (bnds.bound))))
       = 0))) = 0;
  WR4: SIZEOF (QUERY (item <* SELF.items |
       (NOT ('BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF (item))) OR
       ('BUILDING_DESIGN_SCHEMA.PLANE' IN
       TYPEOF (item\face_surface.face_geometry)))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **ground_face_space_boundary_shape_representation** shall contain exactly one **representation_item** in its set of items.

WR2: The **representation_item** in the set of items of a **ground_face_space_boundary_shape_-representation** shall be a **face**.

WR3: The **face** in the set of items of a **ground_face_space_boundary_shape_representation** shall have a boundary defined by a **poly_loop**.

WR4: If the **face** in the set of items of a **ground_face_space_boundary_shape_representation** is a **face_surface**, the underlying geometry shall be defined by a **plane**.

5.2.2.1.37 negative_component

A **negative_component** is a **shape_aspect** that represents a volume of space that is removed from some shape.

EXAMPLE 36 - A door opening is a **negative_component** of a wall.

EXPRESS specification:

```
*)
ENTITY negative_component
  SUBTYPE OF (shape_aspect);
WHERE
  WR1: (SIZEOF (USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'PROPERTY_DEFINITION.DEFINITION')) >= 1)
       AND
       (SIZEOF (QUERY (pd <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'PROPERTY_DEFINITION.DEFINITION') |
       NOT (SIZEOF (USEDIN (pd, 'BUILDING_DESIGN_SCHEMA.'+
       'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) >= 1))) = 0);
  WR2: SIZEOF (TYPEOF (SELF.of_shape.definition) *
       ['BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.STRUCTURE_ENCLOSURE_ELEMENT']) = 1;
  WR3: SIZEOF (USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
  WR4: SIZEOF (QUERY (pd <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.'+
       'PROPERTY_DEFINITION.DEFINITION') |
       NOT (SIZEOF (QUERY (pdr <* USEDIN(pd, 'BUILDING_DESIGN_SCHEMA.'+
       'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') |
       NOT (SIZEOF (TYPEOF (pdr.used_representation) *
       ['BUILDING_DESIGN_SCHEMA.' +
       'ADVANCED_BREP_BUILDING__SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.ADVANCED_CSG_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ADVANCED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ADVANCED_WIRE_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.ELEMENTARY_CSG_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ELEMENTARY_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ELEMENTARY_GEOMETRIC_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ELEMENTARY_WIRE_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.FACETED_CSG_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'FACETED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.FACETED_GEOMETRIC_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'FACETED_WIRE_SHAPE_REPRESENTATION']) = 1))) = 0))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **negative_component** shall be represented.

WR2: A **negative_component** shall be a component for either a **building_element**, **fixture_-equipment_element**, **service_element**, or **structure_enclosure_element**.

WR3: A **negative_component** shall be assigned exactly one name in a **building_item_-identification_assignment**.

WR4: A **negative_component** shall only be represented by an **advanced_brep_building_shape_-representation**, **advanced_csg_shape_representation**, **advanced_face_with_thickness_shape_-representation**, **advanced_wire_shape_representation**, **elementary_csg_shape_representation**, **elementary_face_with_thickness_shape_representation**, **elementary_geometric_shape_-representation**, **elementary_wire_shape_representation**, **faceted_csg_shape_representation**, **faceted_face_with_thickness_shape_representation**, **faceted_geometric_shape_representation**, or **faceted_wire_shape_representation**.

Informal propositions:

IP1: Each boolean operation that a **negative_component** participates in shall be a difference operation.

5.2.2.1.38 opening

An **opening** is a **negative_component** that represents a volume of space that passes through a structural or enclosing element and serves a functional purpose.

EXPRESS specification:

```
*)
ENTITY opening
  SUBTYPE OF (negative_component);
WHERE
  WR1: SIZEOF (USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.'+
       'BUILDING_COMPONENT_CLASSIFICATION_ASSIGNMENT.ITEMS')) = 1;
  WR2: SIZEOF (QUERY (bcca <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.'+
       'BUILDING_COMPONENT_CLASSIFICATION_ASSIGNMENT.ITEMS') |
       NOT (bcca\group_assignment.assigned_group.description = 'opening')
       )) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: The **opening** shall be assigned to exactly one **group**.

WR2: The **opening** shall be assigned to a **group** that classifies it as an opening.

5.2.2.1.39 positive_component

A **positive_component** is a **shape_aspect** that represents a solid volume that is added to some shape.

EXAMPLE 37 - A bracket is a **positive_component** of a beam.

EXPRESS specification:

```
*)
ENTITY positive_component
  SUBTYPE OF (shape_aspect);
WHERE
  WR1: (SIZEOF (USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'PROPERTY_DEFINITION.DEFINITION')) >= 1)
       AND
       (SIZEOF (QUERY (pd <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'PROPERTY_DEFINITION.DEFINITION') |
       NOT (SIZEOF (USEDIN (pd, 'BUILDING_DESIGN_SCHEMA.'+
       'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) >= 1))) = 0);
  WR2: SIZEOF (TYPEOF (SELF.of_shape.definition) *
       ['BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
       'BUILDING_DESIGN_SCHEMA.STRUCTURE_ENCLOSURE_ELEMENT']) = 1;
  WR3: SIZEOF (USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
  WR4: SIZEOF (QUERY (pd <* USEDIN(SELF, 'BUILDING_DESIGN_SCHEMA.'+
       'PROPERTY_DEFINITION.DEFINITION') |
       NOT (SIZEOF (QUERY (pdr <* USEDIN(pd, 'BUILDING_DESIGN_SCHEMA.'+
       'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') |
       NOT (SIZEOF (TYPEOF (pdr.used_representation) *
       ['BUILDING_DESIGN_SCHEMA.' +
       'ADVANCED_BREP_BUILDING__SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.ADVANCED_CSG_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ADVANCED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ADVANCED_WIRE_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.ELEMENTARY_CSG_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ELEMENTARY_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.ELEMENTARY_GEOMETRIC_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ELEMENTARY_WIRE_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.FACETED_CSG_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'FACETED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.FACETED_GEOMETRIC_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'FACETED_WIRE_SHAPE_REPRESENTATION']) = 1))) = 0))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **positive_component** shall be represented.

WR2: A **positive_component** shall be a component for either a **building_element**, **fixture_-equipment_element**, **service_element**, or **structure_enclosure_element**.

WR3: A **positive_component** shall be assigned exactly one name in a **building_item_identification_-assignment**.

WR4: A **positive_component** shall only be represented by an **advanced_brep_building_shape_-representation**, **advanced_csg_shape_representation**, **advanced_face_with_thickness_shape_-**

representation, **advanced_wire_shape_representation**, **elementary_csg_shape_representation**, **elementary_face_with_thickness_shape_representation**, **elementary_geometric_shape_-representation**, **elementary_wire_shape_representation**, **faceted_csg_shape_representation**, **faceted_face_with_thickness_shape_representation**, **faceted_geometric_shape_representation**, or **faceted_wire_shape_representation**.

Informal propositions:

IP1: Each boolean operation that a **positive_component** participates in shall be a union operation.

5.2.2.1.40 recess

A **recess** is a **negative_component** that represents a volume of space that does not pass through a structural or enclosing element.

EXPRESS specification:

```
*)
ENTITY recess
  SUBTYPE OF (negative_component);
WHERE
  WR1: SIZEOF (USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'BUILDING_COMPONENT_CLASSIFICATION_ASSIGNMENT.ITEMS')) = 1;
  WR2: SIZEOF (QUERY (bcca <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.'+
       'BUILDING_COMPONENT_CLASSIFICATION_ASSIGNMENT.ITEMS') |
       NOT (bcca\group_assignment.assigned_group.description = 'recess')
       )) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: The **recess** shall be assigned to exactly one **group**.

WR2: The **recess** shall be assigned to a **group** that classifies it as a recess.

5.2.2.1.41 service_element

A **service_element** is a **product_definition** that defines a solid, tangible part of a building that is a component in one or more systems installed in the building which provide a service to the building.

EXPRESS specification:

```
*)
ENTITY service_element
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN(SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION') |
       'BUILDING_DESIGN_SCHEMA.BUILDING_SECTION'
       IN TYPEOF (pdr.relating_product_definition))) = 1;
  WR2: SIZEOF (USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
  WR3: SIZEOF (QUERY (pds <* QUERY (pd <* USEDIN (SELF,
```

```
       'BUILDING_DESIGN_SCHEMA.PROPERTY_DEFINITION.DEFINITION') |
       'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN
       TYPEOF (pd)) |
       SIZEOF (QUERY (sa <* USEDIN(pds, 'BUILDING_DESIGN_SCHEMA.'+
       'SHAPE_ASPECT.OF_SHAPE') |
       ('BUILDING_DESIGN_SCHEMA.POSITIVE_COMPONENT' IN TYPEOF (sa))
       AND
       (sa.description = 'main component'))) = 1)) = 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **service_element** shall be the related_product_definition in exactly one **product_definition_-relationship** where the relating_product_definition is a **building_section**.

WR2: A **service_element** shall be assigned exactly one name in a **building_item_identification_-assignment**.

WR3: A **service_element** shall have a **product_definition_shape** that is referenced by at exactly one **positive_component** with a description of 'main component'.

5.2.2.1.42 site

A **site** is a **characterized_object** and a **product_definition** that defines the site of a building.

EXPRESS specification:

```
*)
ENTITY site
  SUBTYPE OF (characterized_object, product_definition);
WHERE
  WR1: (SIZEOF (USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'PROPERTY_DEFINITION.DEFINITION')) >= 1)
       AND
       (SIZEOF (QUERY (pd <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.'+
       'PROPERTY_DEFINITION.DEFINITION') |
       SIZEOF (USEDIN (pd, 'BUILDING_DESIGN_SCHEMA.'+
       'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) = 0)) = 0);
  WR2: SIZEOF (QUERY (pd <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'PROPERTY_DEFINITION.DEFINITION') |
       NOT (SIZEOF (QUERY (pdr <* USEDIN (pd, 'BUILDING_DESIGN_SCHEMA.'+
       'PRODUCT_DEFINITION_REPRESENTATION.DEFINITION') |
       NOT ('BUILDING_DESIGN_SCHEMA.SITE_REPRESENTATION' IN
       TYPEOF (pdr.used_representation)))) = 0))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **site** shall be represented.

WR2: A **site** shall be represented by a **site_representation**.

5.2.2.1.43 site_representation

A **site_representation** is a **shape_representation** that represents the shape of a site.

EXPRESS specification:

```
*)
ENTITY site_representation
  SUBTYPE OF(shape_representation);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN (SELF,'BUILDING_DESIGN_SCHEMA.'+
       'PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTATION') |
       NOT('BUILDING_DESIGN_SCHEMA.SITE' IN
       TYPEOF (pdr.definition.definition)))) = 0;
  WR2: SIZEOF (QUERY (item <* SELF.items |
       NOT (SIZEOF (['BUILDING_DESIGN_SCHEMA.CONNECTED_FACE_SET',
       'BUILDING_DESIGN_SCHEMA.GEOMETRIC_CURVE_SET'] *
       TYPEOF (item)) = 1))) = 1;
  WR3: SIZEOF (QUERY (cfs <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.CONNECTED_FACE_SET' IN TYPEOF (item)) |
       NOT (SIZEOF (QUERY (fcs <* cfs\connected_face_set.cfs_faces |
       NOT (SIZEOF (QUERY (bnds <* fcs.bounds |
       NOT ('BUILDING_DESIGN_SCHEMA.POLY_LOOP' IN TYPEOF (bnds.bound))))
       = 0))) = 0))) = 0;
  WR4: SIZEOF (QUERY (cfs <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.CONNECTED_FACE_SET' IN TYPEOF (item)) |
       NOT (SIZEOF (QUERY (fcs <* cfs\connected_face_set.cfs_faces |
       NOT (SIZEOF (QUERY (bnds <* fcs.bounds |
       NOT (SIZEOF (bnds.bound\poly_loop.polygon) = 3)))
       = 0))) = 0))) = 0;
  WR5: SIZEOF (QUERY (gcs <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.GEOMETRIC_CURVE_SET' IN TYPEOF (item)) |
       NOT (SIZEOF (QUERY (el <* gcs\geometric_set.elements |
       NOT (SIZEOF (['BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT',
       'BUILDING_DESIGN_SCHEMA.POLYLINE'] * TYPEOF (el)) = 1))) = 0))) = 0;
  WR6: SIZEOF (QUERY (gcs <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.GEOMETRIC_CURVE_SET' IN TYPEOF (item)) |
       NOT (SIZEOF (QUERY (el <* gcs\geometric_set.elements |
       'BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT' IN TYPEOF (el))) >= 1)))
       = 0;
  WR7: SIZEOF (QUERY (gcs <* QUERY (item <* SELF.items |
       'BUILDING_DESIGN_SCHEMA.GEOMETRIC_CURVE_SET' IN TYPEOF (item)) |
       NOT (SIZEOF (QUERY (pline <* QUERY (el <* gcs\geometric_set.elements
       | 'BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF (el)) |
       NOT (SIZEOF (QUERY (pline_pt <* pline\polyline.points |
       NOT (pline_pt IN gcs\geometric_set.elements))) = 0))) = 0))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **site_representation** shall be used to represent a **site**.

WR2: A **site_representation** shall have in its set of items exactly one **connected_face_set** or **geometric_curve_set**.

WR3: If the **representation_item** is a **connected_face_set**, it shall contain faces that are bounded by **poly_loop**s.

WR4: If the **representation_item** is a **connected_face_set**, all of its **face** instances shall be bounded by **poly_loop**s with the aggregate polygon defined by exactly three **cartesian_point**s.

WR5: If the **representation_item** is a **geometric_curve_set**, its elements set shall consist of **cartesian_point**s or **polyline**s.

WR6: If the **representation_item** is a **geometric_curve_set**, its elements shall consist of at least one **cartesian_point**.

WR7: If the **representation_item** is a **geometric_curve_set**, its elements that are of type **polyline** shall reference only points that are in the elements set.

5.2.2.1.44 space_element

A **space_element** is a **product_definition** that defines an intangible volumetric region of a building that represents portions of a building usable by building occupants or building service equipment.

EXPRESS specification:

```
*)
ENTITY space_element
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION') |
       'BUILDING_DESIGN_SCHEMA.BUILDING_SECTION' IN
       TYPEOF (pdr.relating_product_definition))) = 1;
  WR2: SIZEOF (USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
  WR3: (SIZEOF (QUERY (pd <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'PROPERTY_DEFINITION.DEFINITION') |
       'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN
       TYPEOF (pd))) >= 1)
       AND
       (SIZEOF (QUERY (pds <* QUERY (pd <* USEDIN (SELF,
       'BUILDING_DESIGN_SCHEMA.PROPERTY_DEFINITION.DEFINITION') |
       'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN TYPEOF (pd)) |
       NOT (SIZEOF (USEDIN (pds, 'BUILDING_DESIGN_SCHEMA.' +
       'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) >= 1))) = 0);
  WR4: SIZEOF (QUERY (pd <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'PROPERTY_DEFINITION.DEFINITION') |
       NOT (SIZEOF (QUERY (pdr <* USEDIN (pd, 'BUILDING_DESIGN_SCHEMA.'+
       'PRODUCT_DEFINITION_REPRESENTATION.DEFINITION') |
       NOT ( SIZEOF (TYPEOF (pdr.used_representation) *
       ['BUILDING_DESIGN_SCHEMA.' +
       'ADVANCED_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ELEMENTARY_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'FACETED_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'GROUND_FACE_SPACE_BOUNDARY_SHAPE_REPRESENTATION']) = 1))) = 0)))
       = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **space_element** shall be the related_product_definition in exactly one **product_definition_relationship** where the relating_product_definition is a **building_section**.

WR2: A **space_element** shall be assigned exactly one name in a **building_item_identification_assignment**.

WR3: A **space_element** shall have a **product_definition_shape**, and that **product_definition_shape** shall be represented.

WR4: A **space_element** shall only be represented by an **advanced_space_boundary_shape_representation**, an **elementary_space_boundary_shape_representation**, a **faceted_space_boundary_shape_representation**, or a **ground_face_space_boundary_shape_representation**.

5.2.2.1.45 structure_enclosure_element

A **structure_enclosure_element** is a **product_definition** that defines an identifiable part of a building that contributes to the basic form or function of a building. The functional roles are differentiated by characteristics such as load bearing behaviour, space separating traits, or protection.

EXPRESS specification:

```
*)
ENTITY structure_enclosure_element
  SUBTYPE OF (product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdr <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION') |
       'BUILDING_DESIGN_SCHEMA.BUILDING_SECTION'
       IN TYPEOF (pdr.relating_product_definition))) = 1;
  WR2: SIZEOF (USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
       'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
  WR3: SIZEOF (QUERY (pds <* QUERY (pd <* USEDIN (SELF,
       'BUILDING_DESIGN_SCHEMA.PROPERTY_DEFINITION.DEFINITION') |
       'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN TYPEOF (pd)) |
       SIZEOF (QUERY (sa <* USEDIN (pds, 'BUILDING_DESIGN_SCHEMA.'+
       'SHAPE_ASPECT.OF_SHAPE') |
       ('BUILDING_DESIGN_SCHEMA.POSITIVE_COMPONENT' IN TYPEOF (sa))
           AND
       (sa.description = 'main component'))) = 1)) = 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **structure_enclosure_element** shall be the related_product_definition in exactly one **product_definition_relationship** where the relating_product_definition is a **building_section**.

WR2: A **structure_enclosure_element** shall be assigned exactly one name in a **building_item_identification_assignment**.

WR3: A **structure_enclosure_element** shall have a **product_definition_shape** that is referenced by exactly one **positive_component** with a description of 'main component'.

5.2.2.1.46 truncated_pyramid

A **truncated_pyramid** is a **boolean_result** that is a solid primitive with a square bottom face, a square top face of smaller size than the bottom face, and four trapezoidal faces that connect the top face and bottom face.

EXPRESS specification:

```
*)
ENTITY truncated_pyramid
  SUBTYPE OF (boolean_result);
END_ENTITY;
(*
```

5.2.2.2 building_design imported entity modifications

5.2.2.2.1 application_context

The base definition of the **application_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 **application_context** entity:

— application_context_requires_ap_definition (see 5.2.3.1);

— restrict_application_context (see 5.2.3.4).

5.2.2.2.2 application_protocol_definition

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

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **application_protocol_-definition** entity:

— application_context_requires_ap_definition (see 5.2.3.1).

5.2.2.2.3 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 rules defined in this part of ISO 10303 apply to the **geometric_representation_-context** entity:

— geometric_representation_context_3d (see 5.2.3.3).

5.2.2.2.4 geometric_set

The base definition of the **geometric_set** 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_set** entity:

— subtype_mandatory_geometric_set (see 5.2.3.7).

5.2.2.2.5 group

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

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **group** entity:

— subtype_mandatory_group (see 5.2.3.8).

5.2.2.2.6 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 rules defined in this part of ISO 10303 apply to the **mapped_item** entity:

— building_element_maps_into_building_section (see 5.2.3.2);

— restrict_origin_and_target (see 5.2.3.5).

5.2.2.2.7 representation_relationship_with_transformation

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

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **representation_relationship_with_transformation** entity:

— building_element_maps_into_building_section (see 5.2.3.2).

5.2.2.2.8 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 rules defined in this part of ISO 10303 apply to the **shape_representation** entity:

— shape_representation_subtype_exclusiveness (see 5.2.3.6).

5.2.2.2.9 shape_aspect_relationship

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

Attribute definitions:

description: text that relates the nature of the **shape_aspect_relationship**. It defines the order of the addition and subtractions of the **shape_aspect**s within a **building_element**, **fixture_equipment_element**, **service_element**, or **structure_enclosure_element**.

5.2.2.2.10 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.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **solid_model** entity:

— subtype_mandatory_solid_model (see 5.2.3.9).

5.2.3 building_design rule definitions

5.2.3.1 application_context_requires_ap_definition

The **application_context_requires_ap_definition** rule specifies that each instance of **application_context** shall be referenced by exactly one **application_protocol_definition** that specifies this part of ISO 10303.

EXPRESS specification:

```
*)
RULE application_context_requires_ap_definition FOR
  (application_context, application_protocol_definition);
WHERE
  WR1: SIZEOF (QUERY (ac <* application_context |
       NOT (SIZEOF (QUERY (apd <* application_protocol_definition |
       (ac :=: apd.application)
       AND
       (apd.application_interpreted_model_schema_name =
       'building_design_schema'))) = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

application_context: the set of all instances of **application_context** entities.

application_protocol_definition: the set of all instances of **application_protocol_definition** entities.

Formal propositions:

WR1: For each instance of **application_context**, there shall be exactly one instance of **application_-protocol_definition** that references the instance of **application_context** as its application with a value of 'building_design_schema' as its application_interpreted_model_schema_name.

5.2.3.2 building_element_maps_into_building_section

When a **representation** which represents a **building_element**, **fixture_equipment_element**, **service_-element**, **space_element**, or **structure_enclosure_element** is used as the mapped_representation of the source of a **mapped_item**, that **mapped_item** shall be used by exactly one **representation** which represents a **building_section**. When a **representation** which represents a **building_element**, **fixture_-equipment_element**, **service_element**, **space_element**, or **structure_enclosure_element** is used as the rep_2 of a **representation_relationship_with_transformation**, exactly one **representation_-relationship_with_transformation** shall have rep_1 be a **representation** which represents a **building_section**.

EXPRESS specification:

```
*)
RULE building_element_maps_into_building_section FOR
  (mapped_item, representation_relationship_with_transformation);
WHERE
  WR1: SIZEOF (QUERY (mi <* mapped_item |
         (SIZEOF (QUERY (pdr <*
                         USEDIN(mi.mapping_source.mapped_representation,
                                'BUILDING_DESIGN_SCHEMA.'+
                                'PROPERTY_DEFINITION_REPRESENTATION.'+
                                'USED_REPRESENTATION') |
           SIZEOF (TYPEOF (pdr.definition.definition) *
                     ['BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
                      'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
                      'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
                      'BUILDING_DESIGN_SCHEMA.SPACE_ELEMENT',
                      'BUILDING_DESIGN_SCHEMA.STRUCTURE_ENCLOSURE_ELEMENT']
```

```
          ) = 1
        )) >= 1)
        AND
        (NOT (SIZEOF (QUERY (r <* USEDIN (mi,'BUILDING_DESIGN_SCHEMA.'+
                                          'REPRESENTATION.ITEMS') |
           SIZEOF (QUERY (pdr <* USEDIN(r,'BUILDING_DESIGN_SCHEMA.'+
                 'PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTATION')
              | ('BUILDING_DESIGN_SCHEMA.BUILDING_SECTION' IN
                  TYPEOF (pdr.definition.definition))
           )) >= 1
        )) = 1))
      )) = 0;
  WR2: (SIZEOF (QUERY (rrwt <*
                       representation_relationship_with_transformation |
        (SIZEOF (QUERY (pdr <*
                       USEDIN(rrwt\representation_relationship.rep_2,
                              'BUILDING_DESIGN_SCHEMA.'+
                              'PROPERTY_DEFINITION_REPRESENTATION.'+
                              'USED_REPRESENTATION') |
           SIZEOF (TYPEOF (pdr.definition.definition) *
                   ['BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
                    'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
                    'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
                    'BUILDING_DESIGN_SCHEMA.SPACE_ELEMENT',
                    'BUILDING_DESIGN_SCHEMA.STRUCTURE_ENCLOSURE_ELEMENT']
           ) = 1
        )) >= 1)
        AND
        (NOT (SIZEOF (QUERY (pdr <*
                      USEDIN(rrwt\representation_relationship.rep_1,
                        'BUILDING_DESIGN_SCHEMA.PROPERTY_DEFINITION_'+
                        'REPRESENTATION.USED_REPRESENTATION') |
           ('BUILDING_DESIGN_SCHEMA.BUILDING_SECTION' IN
                TYPEOF (pdr.definition.definition))
        )) = 1))

      )) = 0);
  WR3: SIZEOF (QUERY (rrwt_1 <*
                      representation_relationship_with_transformation |
        SIZEOF (QUERY (rrwt_2 <*
                       representation_relationship_with_transformation |
           (SIZEOF (QUERY (pdr <*
                    USEDIN(rrwt_1\representation_relationship.rep_2,
                      'BUILDING_DESIGN_SCHEMA.PROPERTY_DEFINITION_' +
                      'REPRESENTATION.USED_REPRESENTATION') |
              SIZEOF (TYPEOF (pdr.definition.definition) *
                      ['BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
                      'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
                       'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
                       'BUILDING_DESIGN_SCHEMA.SPACE_ELEMENT',
                       'BUILDING_DESIGN_SCHEMA.' +
                       'STRUCTURE_ENCLOSURE_ELEMENT']) = 1
           )) >= 1)
           AND
           (rrwt_1\representation_relationship.rep_2 =
            rrwt_2\representation_relationship.rep_2)
        )) >= 1
      )) = 0;
END_RULE;
(*
```

Argument definitions:

mapped_item: the set of all instances of **mapped_item** entities.

representation_relationship_with_transformation: the set of all instances of **representation_-relationship_with_transformation** entities.

Formal propositions:

WR1: Every **mapped_item** which uses a representation that represents a **building_element**, **fixture_-equipment_element**, **service_element**, **space_element**, or **structure_enclosure_element** as its source shall be used by exactly one **representation** which represents a **building_section**.

WR2: Every **representation_relationship_with_transformation** which uses a **representation** that represents a **building_element**, **fixture_equipment_element**, **service_element**, **space_element**, or **structure_enclosure_element** as rep_1 shall have a **representation** as rep_2 which represents a **building_section**.

WR3: No **representation** which represents a **building_element**, **fixture_equipment_element**, **service_element**, **space_element**, or **structure_enclosure_element** shall be used by more than one **representation_relationship_with_transformation** as rep_2.

5.2.3.3 geometric_representation_context_3d

All geometry shall be three dimensional.

EXPRESS specification:

```
*)
RULE geometric_representation_context_3d FOR
  (geometric_representation_context);
WHERE
  WR1: SIZEOF (QUERY (grc <* geometric_representation_context |
       NOT (grc.coordinate_space_dimension = 3)
       )) = 0;
END_RULE;
(*
```

Argument definitions:

geometric_representation_context: the set of all instances of **geometric_representation_context** entities.

Formal propositions:

WR1: All instances of **geometric_representation_context** shall have a value of three for the coordinate_space_dimension.

5.2.3.4 restrict_application_context

An instance of **application_context** shall be a context for the composition of the building shape.

EXPRESS specification:

```
*)
RULE restrict_application_context FOR
  (application_context);
WHERE
  WR1: SIZEOF (QUERY (ac <* application_context |
                NOT (ac.application = 'building shape composition'))) = 0;
END_RULE;
(*
```

Argument definitions:

application_context: the set of all instances of **application_context** entities.

Formal propositions:

WR1: Every instance of **application_context** shall have a value 'building shape composition' for the application attribute.

5.2.3.5 restrict_origin_and_target

The target of a **mapped_item** shall be an **axis2_placement_3d**, and the origin of any corresponding **representation_map** shall also be an **axis2_placement_3d**.

EXPRESS specification:

```
*)
RULE restrict_origin_and_target FOR
  (mapped_item);
WHERE
  WR1: SIZEOF (QUERY (mi <* mapped_item |
       NOT (('BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D'
       IN TYPEOF (mi.mapping_target))
           AND
       ('BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D'
       IN TYPEOF (mi.mapping_source.mapping_origin))))) = 0;
END_RULE;
(*
```

Argument definitions:

mapped_item: the set of all instances of **mapped_item** entities.

Formal propositions:

WR1: Every instance of **mapped_item** shall have a mapping_target of the type **axis2_placement_3d**, and shall have a mapping_origin of the type **axis2_placement_3d** for the mapping_source.

5.2.3.6 shape_representation_subtype_exclusiveness

An instance of a subtype of **shape_representation** shall be either an **advanced_brep_building_-shape_representation**, **advanced_csg_shape_representation**, **advanced_face_with_thickness_-shape_representation**, **advanced_space_boundary_shape_representation**, **advanced_wire_shape_-representation**, **elementary_csg_shape_representation**, **elementary_face_with_thickness_shape_-representation**, **elementary_geometric_shape_representation**, **elementary_space_boundary_shape_-representation**, **elementary_wire_shape_representation**, **faceted_csg_shape_representation**, **faceted_face_with_thickness_shape_representation**, **faceted_geometric_shape_representation**, **faceted_space_boundary_shape_representation**, **faceted_wire_shape_representation**, **ground_-face_space_boundary_shape_representation**, or **site_representation**.

EXPRESS specification:

```
*)
RULE shape_representation_subtype_exclusiveness FOR
  (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (sr <* shape_representation |
       NOT (SIZEOF (TYPEOF (sr) *
       ['BUILDING_DESIGN_SCHEMA.' +
       'ADVANCED_BREP_BUILDING_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.ADVANCED_CSG_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ADVANCED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ADVANCED_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.ADVANCED_WIRE_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.ELEMENTARY_CSG_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ELEMENTARY_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.ELEMENTARY_GEOMETRIC_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ELEMENTARY_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ELEMENTARY_WIRE_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.FACETED_CSG_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'FACETED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.FACETED_GEOMETRIC_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'FACETED_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.FACETED_WIRE_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'GROUND_FACE_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.SITE_REPRESENTATION']) <= 1))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: Each instance of one of the subtypes of **shape_representation** shall be either an **advanced_-brep_building_shape_representation**, **advanced_csg_shape_representation**, **advanced_face_with_-**

thickness_shape_representation, **advanced_space_boundary_shape_representation**, **advanced_wire_shape_representation**, **elementary_csg_shape_representation**, **elementary_face_with_thickness_shape_representation**, **elementary_geometric_shape_representation**, **elementary_space_boundary_shape_representation**, **elementary_wire_shape_representation**, **faceted_csg_shape_representation**, **faceted_face_with_thickness_shape_representation**, **faceted_geometric_shape_representation**, **faceted_space_boundary_shape_representation**, **faceted_wire_shape_representation**, **ground_face_space_boundary_shape_representation**, or **site_representation**.

5.2.3.7 subtype_mandatory_geometric_set

All instances of **geometric_set** shall be instances of **geometric_curve_set**.

EXPRESS specification:

```
*)
RULE subtype_mandatory_geometric_set FOR
  (geometric_set);
WHERE
  WR1: SIZEOF (QUERY (gs <* geometric_set |
       NOT ('BUILDING_DESIGN_SCHEMA.GEOMETRIC_CURVE_SET'
       IN TYPEOF (gs)))) = 0;
END_RULE;
(*
```

Argument definitions:

geometric_set: the set of all instances of **geometric_set** entities.

Formal propositions:

WR1: Every instance of **geometric_set** shall also be an instance of **geometric_curve_set**.

5.2.3.8 subtype_mandatory_group

All instances of **group** shall be instances of **building_component_classification_group**.

EXPRESS specification:

```
*)
RULE subtype_mandatory_group FOR
  (group);
WHERE
  WR1: SIZEOF (QUERY (g <* group |
       NOT ('BUILDING_DESIGN_SCHEMA.BUILDING_COMPONENT_CLASSIFICATION_-
GROUP'
       IN TYPEOF (g)))) = 0;
END_RULE;
(*
```

Argument definitions:

group: the set of all instances of **group** entities.

Formal propositions:

WR1: Every instance of **group** shall also be an instance of **building_component_classification_group**.

5.2.3.9 subtype_mandatory_solid_model

An instance of **solid_model** shall be an instance of **csg_solid**, **manifold_solid_brep**, **revolved_area_solid**, **extruded_area_solid**, **advanced_face_with_thickness_shape_representation**, **elementary_face_with_thickness_shape_representation**, or **faceted_face_with_thickness_shape_representation**.

EXPRESS specification:

```
*)
RULE subtype_mandatory_solid_model FOR
  (solid_model);
WHERE
  WR1: SIZEOF (QUERY (sm <* solid_model |
       NOT (SIZEOF (TYPEOF (sm) *
       ['BUILDING_DESIGN_SCHEMA.CSG_SOLID',
       'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP',
       'BUILDING_DESIGN_SCHEMA.EXTRUDED_AREA_SOLID',
       'BUILDING_DESIGN_SCHEMA.REVOLVED_AREA_SOLID',
       'BUILDING_DESIGN_SCHEMA.' +
       'ADVANCED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'ELEMENTARY_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
       'BUILDING_DESIGN_SCHEMA.' +
       'FACETED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION']) = 1))) = 0;
END_RULE;
(*
```

Argument definitions:

solid_model: the set of all instances of **solid_model** entities.

Formal propositions:

WR1: Every instance of **solid_model** shall also be an instance of **csg_solid**, **manifold_solid_brep**, **revolved_area_solid**, **extruded_area_solid**, **advanced_face_with_thickness_shape_representation**, **elementary_face_with_thickness_shape_representation**, or **faceted_face_with_thickness_shape_representation**.

5.2.4 building_design function definitions

5.2.4.1 msb_shells

The **msb_shells** function builds the set of all **closed_shell**s used in the definition of a **manifold_solid_brep**.

EXPRESS specification:

```
*)
FUNCTION msb_shells (brep: manifold_solid_brep) :
                     SET [1:?] OF  closed_shell;
   IF SIZEOF (QUERY (brtyp <* TYPEOF (brep) |
     brtyp LIKE '*.BREP_WITH_VOIDS')) >= 1 THEN
     RETURN (brep\brep_with_voids.voids + brep.outer);
   ELSE
     RETURN ([brep.outer]);
   END_IF;
END_FUNCTION;
(*
```

Argument definitions:

brep: (input) a **manifold_solid_brep** for which a set of **closed_shell** components is required.

schema_name: (input) a STRING giving the name of the schema to which brep belongs.

5.2.4.2 valid_advanced_csg_tree

The **valid_advanced_csg_tree** function returns true if the elements that comprise the CSG tree passed in as a parameter satisfy the requirements defined for advanced CSG trees.

```
*)
FUNCTION valid_advanced_csg_tree (tree_element : boolean_operand) :
BOOLEAN;

  -- return true if the tree_element is a valid primitive

  IF SIZEOF (TYPEOF (tree_element) *
    ['BUILDING_DESIGN_SCHEMA.BLOCK', 'BUILDING_DESIGN_SCHEMA.TORUS',
    'BUILDING_DESIGN_SCHEMA.RIGHT_CIRCULAR_CYLINDER',
    'BUILDING_DESIGN_SCHEMA.SPHERE',
    'BUILDING_DESIGN_SCHEMA.RIGHT_CIRCULAR_CONE',
    'BUILDING_DESIGN_SCHEMA.' +
    'ADVANCED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
    'BUILDING_DESIGN_SCHEMA.EXTRUDED_AREA_SOLID',
    'BUILDING_DESIGN_SCHEMA.REVOLVED_AREA_SOLID',
    'BUILDING_DESIGN_SCHEMA.HALF_SPACED_SOLID']) = 1 THEN RETURN (TRUE);
    ELSE

    -- if the tree_element is a boolean_result check its operations and
    -- operands

    IF 'BUILDING_DESIGN_SCHEMA.BOOLEAN_RESULT' IN TYPEOF (tree_element)
     THEN

      -- addition and subtraction are the only valid operations

      IF NOT (tree_element\boolean_result.operator
                  IN [UNION, DIFFERENCE])
        THEN RETURN (FALSE);
      END_IF;

      -- if the operand is a half_space_solid, check for advanced surface
      -- otherwise return false and recursively check second operand

      IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN
```

```
        TYPEOF (tree_element\boolean_result.first_operand) THEN
        IF 'BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN
          TYPEOF (tree_element\boolean_result.
                  first_operand\half_space_solid.base_surface) THEN
          IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN
            TYPEOF (tree_element\boolean_result.second_operand) THEN
            IF 'BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN
              TYPEOF (tree_element\boolean_result.
                       second_operand\half_space_solid.base_surface) THEN
              RETURN (TRUE);
              ELSE RETURN (FALSE);
            END_IF;
            ELSE RETURN (valid_advanced_csg_tree
            (tree_element\boolean_result.second_operand));
          END_IF;
          ELSE RETURN (FALSE);
        END_IF;
        ELSE
          IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN
            TYPEOF (tree_element\boolean_result.second_operand) THEN
            IF 'BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF
              (tree_element\boolean_result.second_operand\half_space_solid.
               base_surface) THEN
              RETURN (valid_advanced_csg_tree
                        (tree_element\boolean_result.first_operand));
              ELSE
              RETURN (FALSE);
            END_IF;
            ELSE
            RETURN (valid_advanced_csg_tree
                      (tree_element\boolean_result.first_operand) AND
                    valid_advanced_csg_tree
                      (tree_element\boolean_result.second_operand));
          END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;
(*
```

Argument definitions:

tree_element: (input) the **boolean_operand** to be evaluated.

5.2.4.3 valid_advanced_wire_composition

The **valid_advanced_wire_composition** function checks for a variety of **curve**s whether these curves reference legal basis curves within an **advanced_wire_shape_representation**.

The function returns the logic value TRUE, if the types of all potentially recursively referenced basis curves are permitted ones. Also curves that are valid basis curves for any other curves will cause TRUE to be returned.

EXPRESS specification:

```
*)
FUNCTION valid_advanced_wire_composition (sw_element: curve) : BOOLEAN;
```

```
  -- let those types pass that have valid references to basis curves

  IF SIZEOF (['BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE',
    'BUILDING_DESIGN_SCHEMA.CONIC',
    'BUILDING_DESIGN_SCHEMA.LINE',
    'BUILDING_DESIGN_SCHEMA.POLYLINE'] * TYPEOF (sw_element)) = 1 THEN
    RETURN(TRUE);
    ELSE

    -- check in case curve_replica is the type of the input curve

    IF 'BUILDING_DESIGN_SCHEMA.CURVE_REPLICA' IN TYPEOF (sw_element) THEN
      RETURN (valid_advanced_wire_composition
      (sw_element\curve_replica.parent_curve));
      ELSE

      --check in case offset_curve_3d is the type of the input curve

      IF 'BUILDING_DESIGN_SCHEMA.OFFSET_CURVE_3D' IN TYPEOF (sw_element)
        THEN
        RETURN(valid_advanced_wire_composition
        (sw_element\offset_curve_3d.basis_curve));
        ELSE
        IF ('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE') IN TYPEOF (sw_element)
          THEN  RETURN (valid_faceted_wire_composition
          (sw_element\trimmed_curve.basis_curve));
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;
(*
```

Argument definitions:

sw_element: (input) the **curve** to be evaluated.

5.2.4.4 valid_elementary_csg_tree

The **valid_elementary_csg_tree** function returns true if the elements that comprise the CSG tree passed in as a parameter satisfy the requirements defined for elementary CSG trees.

```
*)
FUNCTION valid_elementary_csg_tree (tree_element : boolean_operand) :
BOOLEAN;

  -- return true if the tree_element is a valid primitive

  IF SIZEOF (TYPEOF (tree_element) *
    ['BUILDING_DESIGN_SCHEMA.BLOCK', 'BUILDING_DESIGN_SCHEMA.TORUS',
    'BUILDING_DESIGN_SCHEMA.RIGHT_CIRCULAR_CYLINDER',
    'BUILDING_DESIGN_SCHEMA.SPHERE',
    'BUILDING_DESIGN_SCHEMA.RIGHT_CIRCULAR_CONE',
    'BUILDING_DESIGN_SCHEMA.' +
    'ELEMENTARY_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
    'BUILDING_DESIGN_SCHEMA.EXTRUDED_AREA_SOLID',
    'BUILDING_DESIGN_SCHEMA.REVOLVED_AREA_SOLID',
    'BUILDING_DESIGN_SCHEMA.HALF_SPACED_SOLID']) = 1 THEN RETURN (TRUE);
    ELSE
```

```
    -- if the tree_element is a boolean_result check its operations and
    -- operands

    IF 'BUILDING_DESIGN_SCHEMA.BOOLEAN_RESULT' IN TYPEOF (tree_element)
      THEN

      -- addition and subtraction are the only valid operations

      IF NOT (tree_element\boolean_result.operator
                IN [UNION, DIFFERENCE])
        THEN RETURN (FALSE);
      END_IF;

      -- if the operand is a half_space_solid, check for elementary surface
      -- otherwise return false and recursively check second operand

      IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN
        TYPEOF (tree_element\boolean_result.first_operand) THEN
        IF 'BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN
          TYPEOF (tree_element\boolean_result.
                  first_operand\half_space_solid.base_surface) THEN
          IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN
            TYPEOF (tree_element\boolean_result.second_operand) THEN
            IF 'BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN
              TYPEOF (tree_element\boolean_result.
                      second_operand\half_space_solid.base_surface) THEN
              RETURN (TRUE);
              ELSE RETURN (FALSE);
            END_IF;
            ELSE RETURN (valid_elementary_csg_tree
            (tree_element\boolean_result.second_operand));
          END_IF;
          ELSE RETURN (FALSE);
        END_IF;
        ELSE
          IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN
            TYPEOF (tree_element\boolean_result.second_operand) THEN
            IF 'BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN
              TYPEOF (tree_element\boolean_result.
                      second_operand\half_space_solid.base_surface) THEN
              RETURN (valid_elementary_csg_tree
                        (tree_element\boolean_result.first_operand));
              ELSE
              RETURN (FALSE);
            END_IF;
            ELSE
            RETURN (valid_elementary_csg_tree
                      (tree_element\boolean_result.first_operand) AND
                    valid_elementary_csg_tree
                      (tree_element\boolean_result.second_operand));
          END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;
(*
```

Argument definitions:

tree_element: (input) the **boolean_operand** to be evaluated.

5.2.4.5 valid_elementary_wire_composition

The **valid_elementary_wire_composition** determines if a curve that is pass in is comprised of valid elementary curves.

EXPRESS specification:

```
*)
FUNCTION valid_elementary_wire_composition
  (sw_element : curve) : BOOLEAN;

  -- check for valid basic curves

  IF SIZEOF (['BUILDING_DESIGN_SCHEMA.LINE',
    'BUILDING_DESIGN_SCHEMA.CONIC'] * TYPEOF (sw_element)) = 1
    THEN RETURN (TRUE);
    ELSE

    -- if the curve is a trimmed_curve

    IF ('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE') IN TYPEOF (sw_element) THEN


        -- recursively check the basis_curve

        RETURN (valid_elementary_wire_composition
        (sw_element\trimmed_curve.basis_curve));
    ELSE

      -- recursively check the composite_curve segments

      IF ('BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE') IN
        TYPEOF (sw_element) THEN
        RETURN (SIZEOF (QUERY (ccs <* sw_element\composite_curve.segments |

        NOT (valid_elementary_wire_composition
        (ccs.parent_curve)))) = 0);
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;
(*
```

Argument definitions:

sw_element: (input) the **curve** to be evaluated.

5.2.4.6 valid_faceted_csg_tree

The **valid_faceted_csg_tree** function returns true if the elements that comprise the CSG tree passed in as a parameter satisfy the requirements defined for faceted CSG trees.

```
*)
FUNCTION valid_faceted_csg_tree (tree_element : boolean_operand) : BOOLEAN;

  -- return true if the tree_element is a valid primitive

  IF SIZEOF (['BUILDING_DESIGN_SCHEMA.BLOCK',
```

```
    'BUILDING_DESIGN_SCHEMA.' +
    'FACETED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION'] *
    TYPEOF (tree_element)) = 1 THEN RETURN (TRUE);
    ELSE

      -- if the tree_element is a boolean_result check its operations and
      -- operands

    IF 'BUILDING_DESIGN_SCHEMA.BOOLEAN_RESULT' IN TYPEOF (tree_element)
      THEN

      -- addition and subtraction are the only valid operations

      IF NOT (tree_element\boolean_result.operator
            IN [UNION, DIFFERENCE])
        THEN RETURN (FALSE);
      END_IF;

      -- if the operand is a half_space_solid, check for elementary surface
      -- otherwise return false and recursively check second operand

      IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN
        TYPEOF (tree_element\boolean_result.first_operand) THEN
        IF 'BUILDING_DESIGN_SCHEMA.PLANE' IN TYPEOF
          (tree_element\boolean_result.first_operand\half_space_solid.
           base_surface) THEN
          IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN
            TYPEOF (tree_element\boolean_result.second_operand) THEN
            IF 'BUILDING_DESIGN_SCHEMA.PLANE' IN TYPEOF
              (tree_element\boolean_result.second_operand\half_space_solid.
               base_surface) THEN
              RETURN (TRUE);
              ELSE RETURN (FALSE);
            END_IF;
            ELSE RETURN (valid_faceted_csg_tree
            (tree_element\boolean_result.second_operand));
          END_IF;
          ELSE RETURN (FALSE);
        END_IF;
        ELSE
          IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN
            TYPEOF (tree_element\boolean_result.second_operand) THEN
            IF 'BUILDING_DESIGN_SCHEMA.PLANE' IN
              TYPEOF (tree_element\boolean_result.
                       second_operand\half_space_solid.base_surface) THEN
              RETURN (valid_faceted_csg_tree
                         (tree_element\boolean_result.first_operand));
              ELSE
              RETURN (FALSE);
            END_IF;
            ELSE
            RETURN (valid_faceted_csg_tree
                      (tree_element\boolean_result.first_operand) AND
                    valid_faceted_csg_tree
                      (tree_element\boolean_result.second_operand));
          END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;
(*
```

Argument definitions:

tree_element: (input) the **boolean_operand** to be evaluated.

5.2.4.7 valid_faceted_wire_composition

The **valid_faceted_wire_composition** determines if a curve that is passed in is comprised of valid faceted curves.

EXPRESS specification:

```
*)
FUNCTION valid_faceted_wire_composition
  (sw_element : curve) : BOOLEAN;

  -- check for valid basic curves

  IF 'BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF (sw_element)
    THEN RETURN (TRUE);
    ELSE

    -- if the curve is a trimmed_curve

    IF ('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE') IN TYPEOF (sw_element) THEN

      -- if a line is being trimmed, then valid

      IF SIZEOF (['BUILDING_DESIGN_SCHEMA.LINE',
            'BUILDING_DESIGN_SCHEMA.CONIC'] * TYPEOF
            (sw_element\trimmed_curve.basis_curve)) = 1
        THEN RETURN (TRUE);

        -- otherwise, recursively check the basis_curve

        ELSE RETURN (valid_faceted_wire_composition
        (sw_element\trimmed_curve.basis_curve));
      END_IF;
      ELSE
        IF ('BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE') IN
          TYPEOF (sw_element) THEN
          RETURN (SIZEOF (QUERY (ccs <* sw_element\composite_curve.segments
         | NOT (valid_faceted_wire_composition
          (ccs.parent_curve)))) = 0);
        END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;
(*
```

Argument definitions:

sw_element: (input) the **curve** to be evaluated.

```
*)
END_SCHEMA;  -- building_design_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 10303-21.

Requirements with respect to implementation method-specific requirements are specified in annex C.

The Protocol Implementation Conformance Statement (PICS) proforma lists the options or the combinations of options that may be included in the implementation. The PICS proforma is provided in annex D.

This part of ISO 10303 provides for a number of options that may be supported by an implementation. These options have been grouped into the following conformance classes. Conformance to a class is distinguished principly by the support of several levels of geometric complexity. The three identified levels of geometric complexity are:

— Faceted - geometric representations composed of lines and planes;

— Elementary - geometric representations composed faceted elements and the following curves and surfaces: circles, ellipses, hyperbolas, parabolas, b-spline curves, conical surface, cylindrical surface, spherical surface, and toroidal surface;

— Advanced - geometric representation composed of elementary elements and b_spline surfaces.

The following information is included in each conformance class:

— building element and component property, classification, and administration information;

— building element and component characterization and element assembly information;

EXAMPLE 38 - An element may be characterized (or described) as a Structure_enclosure_element and as a beam. A component may be characterized as a Negative_component and a window opening.

— building composition and spatial arrangement of building elements.

The conformance classes are characterized as follows:

Class 1: Building element and component property, classification, identification, and administration information; building composition and building element spatial arrangement; single level assemblies; and building element and component shape using faceted geometric shape representations.

Class 2: Building element and component property, classification, identification, and administration information; building composition and building element spatial arrangement; single level assemblies; and building element and component shape using faceted shape representations.

NOTE - The term "geometric shape representation" encompasses both geometric sets and b-reps. Omission of the word "geometric" implies that in addition to geometric sets and b-reps, CSG representations are also included.

Class 3: Building element and component property, classification, identification, and administration information; building composition and building element spatial arrangement; single level assemblies; building site shape; and building element and component shape using faceted and elementary geometric shape representations.

Class 4: Building element and component property, classification, identification, and administration information; building composition and building element spatial arrangement; single level assemblies; building site shape; and building element and component shape using faceted and elementary shape representations.

Class 5: Building element and component property, classification, identification, and administration information; building composition and building element spatial arrangement; single level assemblies; building site shape; and building element and component shape using faceted, elementary, and advanced geometric shape representations.

Class 6: Same as Class 1 except that it includes multi-level assemblies.

Class 7: Same as Class 2 except that it includes multi-level assemblies.

Class 8: Same as Class 3 except that it includes multi-level assemblies.

Class 9: Same as Class 4 except that it includes multi-level assemblies.

Class 10: Same as Class 5 except that it includes multi-level assemblies.

Class 11: Building composition and arrangement of spaces; spaces and space property, classification, identification, and administration information; and space shape using faceted and ground face space representations.

Class 12: Building composition and arrangement of spaces; spaces and space property, classification, identification, and administration information; and space shape using faceted, ground face, and elementary space representations.

Class 13: Building composition and arrangement of spaces; spaces and space property, classification, identification, and administration information; and space shape using faceted, ground face, elementary, and advanced space representations.

Class 14: Building complex and surrounding grounds shape and position.

Table 14 specifies the application objects that are included in each conformance class.

Table 14 - Conformance classes

Application objects	1	2	3	4	5	6	7	8	9	10	11	12	13	14
advanced_b_rep					X					X				
advanced_curve			X	X	X			X	X	X				
advanced_face_with_thickness					X					X				
advanced_shell													X	
approval	X	X	X	X	X	X	X	X	X	X	X	X	X	
block		X		X			X		X					
building	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_complex	X	X	X	X	X	X	X	X	X	X	X	X	X	X
building_document_reference	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_element	X	X	X	X	X	X	X	X	X	X				
building_element_component	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_item	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_item_identification	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_level	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_position_in_complex	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_section	X	X	X	X	X	X	X	X	X	X	X	X	X	
change_request	X	X	X	X	X	X	X	X	X	X	X	X	X	
component_location_in_element	X	X	X	X	X	X	X	X	X	X				
component_shape	X	X	X	X	X	X	X	X	X	X	X	X	X	
component_shape_representation	X	X	X	X	X	X	X	X	X	X	X	X	X	
elementary_b_rep			X	X	X			X	X	X				
elementary_curve			X	X	X			X	X	X				
elementary_face_with_thickness			X	X	X			X	X	X				
elementary_shell												X	X	
facet_trigon														X
faceted_b_rep	X	X	X	X	X	X	X	X	X	X				
faceted_curve	X	X	X	X	X	X	X	X	X	X				
faceted_face_with_thickness	X	X	X	X	X	X	X	X	X	X				
faceted_shell											X	X	X	
faceted_surface_representation														X
fixture_equipment_element	X	X	X	X	X	X	X	X	X	X				

Table 14 - Conformance classes (continued)

Application objects	1	2	3	4	5	6	7	8	9	10	11	12	13	14
gis_position	X	X	X	X	X	X	X	X	X	X	X	X	X	X
ground_face											X	X	X	
item_assembly	1	1	1	1	1	2	2	2	2	2	1	1	1	
item_classification	X	X	X	X	X	X	X	X	X	X	X	X	X	
item_group	1	1	1	1	1	2	2	2	2	2	1	1	1	
item_position_in_section	X	X	X	X	X	X	X	X	X	X	X	X	X	
item_proximity_relationship	X	X	X	X	X	X	X	X	X	X	X	X	X	
level_position_in_section	X	X	X	X	X	X	X	X	X	X	X	X	X	
negative_component	X	X	X	X	X	X	X	X	X	X				
opening	X	X	X	X	X	X	X	X	X	X				
placement	X	X	X	X	X	X	X	X	X	X	X	X	X	
point														X
point_and_line_representation														X
polyline														X
positive_component	X	X	X	X	X	X	X	X	X	X				
property	X	X	X	X	X	X	X	X	X	X	X	X	X	
recess	X	X	X	X	X	X	X	X	X	X				
right_circular_cylinder				X					X					
section_position_in_building	X	X	X	X	X	X	X	X	X	X	X	X	X	
service_element	X	X	X	X	X	X	X	X	X	X				
simple_curve	X	X	X	X	X	X	X	X	X	X	X	X	X	
site_shape_representation														X
solid_of_linear_extrusion		X		X			X		X					
solid_of_revolution				X					X					
space											X	X	X	
structure_enclosure_element	X	X	X	X	X	X	X	X	X	X				
sublevel	X	X	X	X	X	X	X	X	X	X	X	X	X	
trimmed_sphere				X					X					
trimmed_torus				X					X					
truncated_cone				X					X					
truncated_pyramid		X		X			X		X					

Conformance classes 1 through 5 differ from 6 through 10 according to the following legend:

1 — Only single level assemblies and groups are allowed;

2 — Nested assemblies and groups are allowed.

Conformance to a particular class requires that all AIM elements defined as part of that class be supported. Table 14 defines the classes to which each AIM element belongs.

NOTE - ISO 10303-325[1)] defines the abstract test suite to be used in the assessment of conformance. ISO 10303-32[1)] describes the conformance assessment process.

Table 15 specifies the AIM elements are included in each conformance class.

Table 15 - Conformance class elements

AIM entity	Class													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
action	X	X	X	X	X	X	X	X	X	X	X	X	X	
action_assignment	X	X	X	X	X	X	X	X	X	X	X	X	X	
action_method	X	X	X	X	X	X	X	X	X	X	X	X	X	
action_request_solution	X	X	X	X	X	X	X	X	X	X	X	X	X	
action_request_status	X	X	X	X	X	X	X	X	X	X	X	X	X	
advanced_brep_building_shape_representation					X					X				
advanced_csg_shape_representation					X					X				
advanced_face					X					X				
advanced_face_with_thickness_shape_-representation					X					X				
advanced_space_boundary_shape_representation													X	
advanced_wire_shape_representation			X	X	X			X	X	X				
application_context	X	X	X	X	X	X	X	X	X	X	X	X	X	X
application_context_element	X	X	X	X	X	X	X	X	X	X	X	X	X	X
application_protocol_definition	X	X	X	X	X	X	X	X	X	X	X	X	X	X
approval	X	X	X	X	X	X	X	X	X	X	X	X	X	
approval_assignment	X	X	X	X	X	X	X	X	X	X	X	X	X	
approval_date_time	X	X	X	X	X	X	X	X	X	X	X	X	X	
approval_person_organization	X	X	X	X	X	X	X	X	X	X	X	X	X	

[1)]To be published

Table 15 - Conformance class elements (continued)

AIM entity	Class													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
approval_role	X	X	X	X	X	X	X	X	X	X	X	X	X	
approval_status	X	X	X	X	X	X	X	X	X	X	X	X	X	
assembly_component_usage	X	X	X	X	X	X	X	X	X	X	X	X	X	
axis1_placement				X	X				X	X				
axis2_placement_2d					X					X				
axis2_placement_3d	X	X	X	X	X	X	X	X	X	X	X	X	X	
b_spline_curve			X	X	X			X	X	X		X	X	
b_spline_curve_with_knots			X	X	X			X	X	X		X	X	
b_spline_surface					X					X			X	
b_spline_surface_with_knots					X					X			X	
bezier_curve			X	X	X			X	X	X		X	X	
bezier_surface					X					X			X	
block		X		X			X		X					
boolean_result		X		X			X		X					
boundary_curve		X		X			X		X					
bounded_curve		X		X	X		X		X	X				X
bounded_surface					X					X				
brep_with_voids			X	X	X			X	X	X				
building	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_complex	X	X	X	X	X	X	X	X	X	X	X	X	X	X
building_component_classification_assignment	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_component_classification_group	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_design_approval	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_design_date_assignment	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_design_organization_assignment	X	X	X	X	X	X	X	X	X	X	X	X	X	X
building_design_person_and_organization_-assignment	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_design_person_assignment	X	X	X	X	X	X	X	X	X	X	X	X	X	X
building_document_reference	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_element	X	X	X	X	X	X	X	X	X	X				

Table 15 - Conformance class elements (continued)

AIM entity	Class													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
building_element_assembly	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_element_group	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_item_identification_assignment	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_level	X	X	X	X	X	X	X	X	X	X	X	X	X	
building_section	X	X	X	X	X	X	X	X	X	X	X	X	X	
calendar_date	X	X	X	X	X	X	X	X	X	X	X	X	X	
cartesian_point	X	X	X	X	X	X	X	X	X	X	X	X	X	X
cartesian_transformation_operator	X	X	X	X	X	X	X	X	X	X	X	X	X	
cartesian_transformation_operator_3d	X	X	X	X	X	X	X	X	X	X	X	X	X	
change	X	X	X	X	X	X	X	X	X	X	X	X	X	
characterized_object	X	X	X	X	X	X	X	X	X	X	X	X	X	X
circle			X	X	X			X	X	X		X	X	
classification_table	X	X	X	X	X	X	X	X	X	X	X	X	X	
closed_shell	X	X	X	X	X	X	X	X	X	X	X	X	X	
composite_curve	X	X	X	X	X	X	X	X	X	X				
composite_curve_on_surface	X	X	X	X	X	X	X	X	X	X				
composite_curve_segment	X	X	X	X	X	X	X	X	X	X				
conic			X	X	X			X	X	X		X	X	
conical_surface			X	X	X			X	X	X		X	X	
connected_face_set	X	X	X	X	X	X	X	X	X	X	X	X	X	X
conversion_based_unit	X	X	X	X	X	X	X	X	X	X				
csg_solid		X		X	X		X		X	X				
curve	X	X	X	X	X	X	X	X	X	X	X	X	X	X
curve_bounded_surface		X		X			X		X					
curve_replica			X	X	X			X	X	X				
cylindrical_surface			X	X	X			X	X	X		X	X	
date	X	X	X	X	X	X	X	X	X	X	X	X	X	
date_assignment	X	X	X	X	X	X	X	X	X	X	X	X	X	
date_role	X	X	X	X	X	X	X	X	X	X	X	X	X	
definitional_representation					X					X				

Table 15 - Conformance class elements (continued)

AIM entity	Class													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
degenerate_toroidal_surface					X					X				
derived_unit	X	X	X	X	X	X	X	X	X	X				
derived_unit_element	X	X	X	X	X	X	X	X	X	X				
descriptive_representation_item	X	X	X	X	X	X	X	X	X	X	X	X	X	X
dimensional_exponents	X	X	X	X	X	X	X	X	X	X				
direction	X	X	X	X	X	X	X	X	X	X	X	X	X	
document	X	X	X	X	X	X	X	X	X	X	X	X	X	
document_reference	X	X	X	X	X	X	X	X	X	X	X	X	X	
document_type	X	X	X	X	X	X	X	X	X	X	X	X	X	
document_usage_constraint	X	X	X	X	X	X	X	X	X	X	X	X	X	
edge	X	X	X	X	X	X	X	X	X	X	X	X	X	
edge_curve	X	X	X	X	X	X	X	X	X	X	X	X	X	
edge_loop	X	X	X	X	X	X	X	X	X	X	X	X	X	
elementary_csg_shape_representation				X					X					
elementary_face_with_thickness_shape_-representation			X	X	X			X	X	X				
elementary_geometric_shape_representation			X	X	X			X	X	X				
elementary_space_boundary_shape_representation												X	X	
elementary_wire_shape_representation			X	X	X			X	X	X				
elementary_surface			X	X	X			X	X	X				
ellipse			X	X	X			X	X	X		X	X	
extruded_area_solid		X		X			X		X					
face	X	X	X	X	X	X	X	X	X	X	X	X	X	X
face_bound	X	X	X	X	X	X	X	X	X	X	X	X	X	X
face_outer_bound	X	X	X	X	X	X	X	X	X	X	X	X	X	X
face_surface	X	X	X	X	X	X	X	X	X	X				
faceted_brep	X		X		X	X		X		X				
faceted_csg_shape_representation		X		X			X		X					
faceted_face_with_thickness_shape_representation	X	X	X	X	X	X	X	X	X	X				
faceted_geometric_shape_representation	X	X	X	X	X	X	X	X	X	X				

Table 15 - Conformance class elements (continued)

AIM entity	Class													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
faceted_space_boundary_shape_representation											X	X	X	
faceted_wire_shape_representation	X	X	X	X	X	X	X	X	X	X				
fixture_equipment_element	X	X	X	X	X	X	X	X	X	X				
functionally_defined_transformation	X	X	X	X	X	X	X	X	X	X	X	X	X	
geometric_curve_set														X
geometric_representation_context	X	X	X	X	X	X	X	X	X	X	X	X	X	X
geometric_representation_item	X	X	X	X	X	X	X	X	X	X	X	X	X	X
geometric_set														X
global_unit_assigned_context	X	X	X	X	X	X	X	X	X	X	X	X	X	X
ground_face_space_boundary_shape_representation											X	X	X	
group	X	X	X	X	X	X	X	X	X	X	X	X	X	
group_assignment	X	X	X	X	X	X	X	X	X	X	X	X	X	
half_space_solid		X		X			X		X					
hyperbola			X	X	X			X	X	X		X	X	
intersection_curve					X					X				
length_measure_with_unit	X	X	X	X	X	X	X	X	X	X				
length_unit	X	X	X	X	X	X	X	X	X	X				
line	X	X	X	X	X	X	X	X	X	X	X	X	X	X
loop	X	X	X	X	X	X	X	X	X	X				X
manifold_solid_brep					X					X				
mapped_item	X	X	X	X	X	X	X	X	X	X	X	X	X	X
measure_representation_item	X	X	X	X	X	X	X	X	X	X				
measure_with_unit	X	X	X	X	X	X	X	X	X	X				
name_assignment	X	X	X	X	X	X	X	X	X	X	X	X	X	
named_unit	X	X	X	X	X	X	X	X	X	X				
negative_component	X	X	X	X	X	X	X	X	X	X				
offset_curve_3d					X					X				
open_shell														X
opening	X	X	X	X	X	X	X	X	X	X				
ordinal_date	X	X	X	X	X	X	X	X	X	X	X	X	X	

Table 15 - Conformance class elements (continued)

AIM entity	Class													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
organization	X	X	X	X	X	X	X	X	X	X	X	X	X	X
organization_assignment	X	X	X	X	X	X	X	X	X	X	X	X	X	X
organization_role	X	X	X	X	X	X	X	X	X	X	X	X	X	X
organizational_project	X	X	X	X	X	X	X	X	X	X	X	X	X	
oriented_closed_shell	X	X	X	X	X	X	X	X	X	X	X	X	X	
oriented_edge	X	X	X	X	X	X	X	X	X	X	X	X	X	
oriented_face	X	X	X	X	X	X	X	X	X	X	X	X	X	
oriented_open_shell														X
oriented_path			X	X	X			X	X	X		X	X	
outer_boundary_curve		X		X			X		X					
parabola			X	X	X			X	X	X		X	X	
parametric_representation_context					X					X				
path			X	X	X			X	X	X		X	X	
pcurve					X					X				
person	X	X	X	X	X	X	X	X	X	X	X	X	X	X
person_and_organization	X	X	X	X	X	X	X	X	X	X	X	X	X	
person_and_organization_assignment	X	X	X	X	X	X	X	X	X	X	X	X	X	
person_and_organization_role	X	X	X	X	X	X	X	X	X	X	X	X	X	
person_assignment	X	X	X	X	X	X	X	X	X	X	X	X	X	X
person_role	X	X	X	X	X	X	X	X	X	X	X	X	X	X
placement	X	X	X	X	X	X	X	X	X	X	X	X	X	
plane	X	X	X	X	X	X	X	X	X	X	X	X	X	
plane_angle_measure_with_unit			X	X	X			X	X	X		X	X	
plane_angle_unit			X	X	X			X	X	X		X	X	
point	X	X	X	X	X	X	X	X	X	X	X	X	X	X
poly_loop	X	X	X	X	X	X	X	X	X	X	X	X	X	X
polyline	X	X	X	X	X	X	X	X	X	X		X	X	X
positive_component	X	X	X	X	X	X	X	X	X	X				
product	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_category	X	X	X	X	X	X	X	X	X	X	X	X	X	

Table 15 - Conformance class elements (continued)

AIM entity	Class													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
product_category_relationship	X	X	X	X	X	X	X	X	X	X	X	X	X	
product_context	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition_context	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition_formation	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition_relationship	X	X	X	X	X	X	X	X	X	X	X	X	X	
product_definition_shape	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition_usage	X	X	X	X	X	X	X	X	X	X	X	X	X	
product_related_product_category	X	X	X	X	X	X	X	X	X	X	X	X	X	
property_definition	X	X	X	X	X	X	X	X	X	X	X	X	X	X
property_definition_representation	X	X	X	X	X	X	X	X	X	X	X	X	X	X
quasi_uniform_curve					X					X				
quasi_uniform_surface					X					X				
rational_b_spline_curve					X					X				
rational_b_spline_surface					X					X				
recess	X	X	X	X	X	X	X	X	X	X				
rectangular_composite_surface					X					X				
rectangular_trimmed_surface					X					X				
representation	X	X	X	X	X	X	X	X	X	X	X	X	X	X
representation_context	X	X	X	X	X	X	X	X	X	X	X	X	X	X
representation_item	X	X	X	X	X	X	X	X	X	X	X	X	X	X
representation_map	X	X	X	X	X	X	X	X	X	X	X	X	X	X
representation_relationship	X	X	X	X	X	X	X	X	X	X	X	X	X	X
representation_relationship_with_transformation	X	X	X	X	X	X	X	X	X	X	X	X	X	X
revolved_area_solid				X					X					
right_circular_cone				X					X					
right_circular_cylinder				X					X					
service_element	X	X	X	X	X	X	X	X	X	X				
shape_aspect	X	X	X	X	X	X	X	X	X	X	X	X	X	
shape_aspect_relationship	X	X	X	X	X	X	X	X	X	X	X	X	X	

Table 15 - Conformance class elements (continued)

AIM entity	Class													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
shape_definition_representation	X	X	X	X	X	X	X	X	X	X	X	X	X	
shape_representation	X	X	X	X	X	X	X	X	X	X	X	X	X	X
si_unit	X	X	X	X	X	X	X	X	X	X				
site	X	X	X	X	X	X	X	X	X	X	X	X	X	X
site_representation														X
solid_model	X	X	X	X	X	X	X	X	X	X				
space_element											X	X	X	
sphere				X					X					
spherical_surface			X	X	X			X	X	X		X	X	
structure_enclosure_element	X	X	X	X	X	X	X	X	X	X				
surface	X	X	X	X	X	X	X	X	X	X	X	X	X	
surface_curve					X					X				
surface_of_linear_extrusion		X		X	X		X		X	X				
surface_of_revolution				X	X				X	X				
surface_patch					X					X				
swept_area_solid		X		X			X		X					
swept_surface		X		X	X		X		X	X				
topological_representation_item	X	X	X	X	X	X	X	X	X	X	X	X	X	X
toroidal_surface			X	X	X			X	X	X		X	X	
torus				X					X					
trimmed_curve	X	X	X	X	X	X	X	X	X	X				
truncated_pyramid		X		X			X		X					
uniform_curve					X				X					
uniform_surface					X				X					
vector	X	X	X	X	X	X	X	X	X	X	X	X	X	X
versioned_action_request	X	X	X	X	X	X	X	X	X	X	X	X	X	
vertex	X	X	X	X	X	X	X	X	X	X	X	X	X	
vertex_loop	X	X	X	X	X	X	X	X	X	X	X	X	X	
vertex_point	X	X	X	X	X	X	X	X	X	X	X	X	X	
week_of_year_and_day_date	X	X	X	X	X	X	X	X	X	X	X	X	X	

Annex A
(normative)

AIM EXPRESS expanded listing

The following EXPRESS is the expanded form of the short form schema given in 5.2. In the event of any discrepancy between the short form and this expanded listing, the expanded listing shall be used.

```
*)
SCHEMA building_design_schema;

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

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

  TYPE b_spline_surface_form = ENUMERATION OF
    (plane_surf,
     cylindrical_surf,
     conical_surf,
     spherical_surf,
     toroidal_surf,
     surf_of_revolution,
     ruled_surf,
     generalised_cone,
     quadric_surf,
     surf_of_linear_extrusion,
     unspecified);
  END_TYPE; -- b_spline_surface_form

  TYPE boolean_operand = SELECT
    (solid_model,
     half_space_solid,
     csg_primitive,
     boolean_result);
  END_TYPE; -- boolean_operand

  TYPE boolean_operator = ENUMERATION OF
    (union,
     intersection,
     difference);
  END_TYPE; -- boolean_operator

  TYPE building_component_classification_item = SELECT
    (negative_component,
     positive_component,
     opening,
     recess);
  END_TYPE; -- building_component_classification_item

  TYPE building_design_approval_item = SELECT
```

```
    (building_element_assembly,
     building_element,
     fixture_equipment_element,
     service_element,
     space_element,
     structure_enclosure_element,
     change,
     positive_component,
     negative_component);
  END_TYPE; -- building_design_approval_item

  TYPE building_design_change_item = SELECT
    (identified_item,
     resulting_item);
  END_TYPE; -- building_design_change_item

  TYPE building_design_date_item = SELECT
    (versioned_action_request);
  END_TYPE; -- building_design_date_item

  TYPE building_design_organization_item = SELECT
    (building,
     building_complex,
     building_item_identification_assignment,
     versioned_action_request);
  END_TYPE; -- building_design_organization_item

  TYPE building_design_person_and_organization_item = SELECT
    (versioned_action_request);
  END_TYPE; -- building_design_person_and_organization_item

  TYPE building_design_person_item = SELECT
    (building,
     building_item_identification_assignment,
     versioned_action_request);
  END_TYPE; -- building_design_person_item

  TYPE building_document_item = SELECT
    (negative_component,
     positive_component,
     building_element,
     fixture_equipment_element,
     service_element,
     space_element,
     structure_enclosure_element,
     building_component_classification_group,
     product_category);
  END_TYPE; -- building_document_item

  TYPE characterized_definition = SELECT
    (characterized_object,
     characterized_product_definition,
     shape_definition);
  END_TYPE; -- characterized_definition

  TYPE characterized_product_definition = SELECT
    (product_definition,
     product_definition_relationship);
  END_TYPE; -- characterized_product_definition

  TYPE csg_primitive = SELECT
    (sphere,
     block,
     torus,
```

```
    right_circular_cone,
    right_circular_cylinder);
END_TYPE; -- csg_primitive

TYPE csg_select = SELECT
  (boolean_result,
   csg_primitive);
END_TYPE; -- csg_select

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

TYPE date_time_select = SELECT
  (date);
END_TYPE; -- date_time_select

TYPE day_in_month_number = INTEGER;
END_TYPE; -- day_in_month_number

TYPE day_in_week_number = INTEGER;
WHERE
  wr1: (1 <= SELF) AND (SELF <= 7);
END_TYPE; -- day_in_week_number

TYPE day_in_year_number = INTEGER;
END_TYPE; -- day_in_year_number

TYPE dimension_count = INTEGER;
WHERE
  wr1: SELF > 0;
END_TYPE; -- dimension_count

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

TYPE identified_item = SELECT
  (building_element,
   fixture_equipment_element,
   service_element,
   space_element,
   structure_enclosure_element,
   building_level,
   building_section,
   positive_component,
   negative_component);
END_TYPE; -- identified_item

TYPE identifier = STRING;
END_TYPE; -- identifier

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

TYPE label = STRING;
```

```
END_TYPE; -- label

TYPE length_measure = REAL;
END_TYPE; -- length_measure

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

TYPE measure_value = SELECT
  (length_measure,
   plane_angle_measure,
   parameter_value,
   positive_length_measure);
END_TYPE; -- measure_value

TYPE month_in_year_number = INTEGER;
WHERE
  wr1: (1 <= SELF) AND (SELF <= 12);
END_TYPE; -- month_in_year_number

TYPE parameter_value = REAL;
END_TYPE; -- parameter_value

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

TYPE person_organization_select = SELECT
  (person,
   organization,
   person_and_organization);
END_TYPE; -- person_organization_select

TYPE plane_angle_measure = REAL;
END_TYPE; -- plane_angle_measure

TYPE positive_length_measure = length_measure;
WHERE
  wr1: SELF > 0;
END_TYPE; -- positive_length_measure

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

TYPE resulting_item = SELECT
  (building_element,
   fixture_equipment_element,
   service_element,
   space_element,
   structure_enclosure_element,
   positive_component,
   negative_component);
END_TYPE; -- resulting_item

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

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

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

TYPE shape_definition = SELECT
  (product_definition_shape,
   shape_aspect,
   shape_aspect_relationship);
END_TYPE; -- shape_definition

TYPE shell = SELECT
  (open_shell,
   closed_shell);
END_TYPE; -- shell

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

TYPE si_unit_name = ENUMERATION OF
  (metre,
   gram,
   second,
   ampere,
   kelvin,
   mole,
   candela,
   radian,
   steradian,
   hertz,
   newton,
   pascal,
   joule,
   watt,
   coulomb,
   volt,
   farad,
   ohm,
   siemens,
   weber,
```

```
    tesla,
    henry,
    degree_celsius,
    lumen,
    lux,
    becquerel,
    gray,
    sievert);
  END_TYPE; -- si_unit_name

  TYPE supported_item = SELECT
    (action,
     action_method);
  END_TYPE; -- supported_item

  TYPE text = STRING;
  END_TYPE; -- text

  TYPE transformation = SELECT
    (functionally_defined_transformation);
  END_TYPE; -- transformation

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

  TYPE trimming_preference = ENUMERATION OF
    (cartesian,
     parameter,
     unspecified);
  END_TYPE; -- trimming_preference

  TYPE trimming_select = SELECT
    (cartesian_point,
     parameter_value);
  END_TYPE; -- trimming_select

  TYPE unit = SELECT
    (named_unit,
     derived_unit);
  END_TYPE; -- unit

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

  TYPE week_in_year_number = INTEGER;
  WHERE
    wr1: (1 <= SELF) AND (SELF <= 53);
  END_TYPE; -- week_in_year_number

  TYPE year_number = INTEGER;
  END_TYPE; -- year_number

  ENTITY action;
      name          : label;
      description   : text;
      chosen_method : action_method;
  END_ENTITY; -- action
```

```
ENTITY action_assignment
  ABSTRACT SUPERTYPE;
    assigned_action : action;
END_ENTITY; -- action_assignment

ENTITY action_method;
    name        : label;
    description : text;
    consequence : text;
    purpose     : text;
END_ENTITY; -- action_method

ENTITY action_request_solution;
    method  : action_method;
    request : versioned_action_request;
END_ENTITY; -- action_request_solution

ENTITY action_request_status;
    status           : label;
    assigned_request : versioned_action_request;
END_ENTITY; -- action_request_status

ENTITY advanced_brep_building_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    wr1: SIZEOF(QUERY ( it <* SELF.items | (NOT ((SIZEOF([
              'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP',
              'BUILDING_DESIGN_SCHEMA.FACETED_BREP',
              'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
              'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D'] * TYPEOF(it)) =
               1) OR (it.name = 'reference curve'))) )) = 0;
    wr2: (SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
              'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP',
              'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF(it)) = 1) ))
              >= 1) OR ((2 <= SIZEOF(QUERY ( it <* SELF.items | (it.name =
               'reference curve') ))) AND (SIZEOF(QUERY ( it <* SELF.items
               | (it.name = 'reference curve') )) <= 3));
    wr3: SIZEOF(QUERY ( msb <* QUERY ( it <* SELF.items | (
            'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) )
             | (NOT (SIZEOF(QUERY ( csh <* msb_shells(msb) | (NOT (
            SIZEOF(QUERY ( fcs <* csh.cfs_faces | (NOT (
            'BUILDING_DESIGN_SCHEMA.ADVANCED_FACE' IN TYPEOF(fcs))) )) =
             0)) )) = 0)) )) = 0;
    wr4: SIZEOF(QUERY ( msb <* QUERY ( it <* items | (
            'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) )
             | ('BUILDING_DESIGN_SCHEMA.ORIENTED_CLOSED_SHELL' IN
            TYPEOF(msb.outer)) )) = 0;
    wr5: SIZEOF(QUERY ( brv <* QUERY ( it <* items | (
              'BUILDING_DESIGN_SCHEMA.BREP_WITH_VOIDS' IN TYPEOF(it)) ) |
              (NOT (SIZEOF(QUERY ( csh <* brv.voids | csh.orientation )) =
               0)) )) = 0;
    wr6: SIZEOF(QUERY ( mi <* QUERY ( it <* items | (
              'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF(it)) ) | (
              NOT (('BUILDING_DESIGN_SCHEMA.' +
              'ADVANCED_BREP_BUILDING_SHAPE_REPRESENTATION') IN TYPEOF(mi
              .mapping_source.mapped_representation))) )) = 0;
    wr7: SIZEOF(QUERY ( it <* SELF.items | ((it.name = 'reference curve')
              AND (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF(it))
              OR (('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(it))
              AND ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN TYPEOF(it\
              trimmed_curve.basis_curve)))))) )) = 0;
END_ENTITY; -- advanced_brep_building_shape_representation

ENTITY advanced_csg_shape_representation
```

```
    SUBTYPE OF (shape_representation);
    WHERE
      wr1: SIZEOF(QUERY ( item <* SELF.items | (NOT ((SIZEOF([
                'BUILDING_DESIGN_SCHEMA.CSG_SOLID',
                'BUILDING_DESIGN_SCHEMA.EXTRUDED_AREA_SOLID',
                'BUILDING_DESIGN_SCHEMA.REVOLVED_AREA_SOLID',
                'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D',
                'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF(item)) = 1)
                OR (item.name = 'reference curve'))) )) = 0;
      wr2: SIZEOF(QUERY ( item <* SELF.items | (SIZEOF([
                'BUILDING_DESIGN_SCHEMA.CSG_SOLID',
                'BUILDING_DESIGN_SCHEMA.EXTRUDED_AREA_SOLID',
                'BUILDING_DESIGN_SCHEMA.REVOLVED_AREA_SOLID',
                'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF(item)) = 1) )
               >= 1) OR ((2 <= SIZEOF(QUERY ( it <* SELF.items | (it.name =
                 'reference curve') ))) AND (SIZEOF(QUERY ( it <* SELF.items
                 | (it.name = 'reference curve') )) <= 3));
      wr3: SIZEOF(QUERY ( item <* SELF.items | ((
                'BUILDING_DESIGN_SCHEMA.CSG_SOLID' IN TYPEOF(item)) AND (
                NOT valid_advanced_csg_tree(item\csg_solid.
                tree_root_expression))) )) = 0;
      wr4: SIZEOF(QUERY ( mi <* QUERY ( item <* SELF.items | (
                'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF(item)) ) | (
                NOT (
                'BUILDING_DESIGN_SCHEMA.ADVANCED_CSG_SHAPE_REPRESENTATION'
                IN TYPEOF(mi\mapped_item.mapping_source.
                mapped_representation))) )) = 0;
      wr5: SIZEOF(QUERY ( it <* SELF.items | ((it.name = 'reference curve')

                AND (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF(it))
                OR (('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(it))
                AND ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN TYPEOF(it\
                trimmed_curve.basis_curve)))))) )) = 0;
  END_ENTITY; -- advanced_csg_shape_representation

  ENTITY advanced_face
    SUBTYPE OF (face_surface);
    WHERE
      wr1 : SIZEOF(['BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE',
                'BUILDING_DESIGN_SCHEMA.B_SPLINE_SURFACE',
                'BUILDING_DESIGN_SCHEMA.SWEPT_SURFACE'] * TYPEOF(
                face_geometry)) = 1;
      wr2 : SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* SELF.bounds | (
                'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list | (NOT ('BUILDING_DESIGN_SCHEMA.EDGE_CURVE' IN
                TYPEOF(oe.edge_element))) )) = 0)) )) = 0;
      wr3 : SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* SELF.bounds | (
                'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list | (NOT (SIZEOF(['BUILDING_DESIGN_SCHEMA.LINE',
                'BUILDING_DESIGN_SCHEMA.CONIC',
                'BUILDING_DESIGN_SCHEMA.POLYLINE',
                'BUILDING_DESIGN_SCHEMA.SURFACE_CURVE',
                'BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE'] * TYPEOF(oe.
                edge_element\edge_curve.edge_geometry)) = 1)) )) = 0)) )) =
                 0;
      wr4 : SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* SELF.bounds | (
                'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list | (NOT (('BUILDING_DESIGN_SCHEMA.VERTEX_POINT' IN
                 TYPEOF(oe.edge_start)) AND (
                'BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT' IN TYPEOF(oe.
                edge_start\vertex_point.vertex_geometry)) AND (
```

```
                'BUILDING_DESIGN_SCHEMA.VERTEX_POINT'
                IN TYPEOF(oe.edge_end))
                AND ('BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT' IN TYPEOF(oe.
                edge_end\vertex_point.vertex_geometry)))) )) = 0)) )) = 0;
      wr5 : SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* SELF.bounds | (
                'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) )
                 | ('BUILDING_DESIGN_SCHEMA.ORIENTED_PATH' IN TYPEOF(
                elp_fbnds.bound)) )) = 0;
      wr6 : (NOT ('BUILDING_DESIGN_SCHEMA.SWEPT_SURFACE' IN TYPEOF(
                face_geometry))) OR (SIZEOF(['BUILDING_DESIGN_SCHEMA.LINE',
                'BUILDING_DESIGN_SCHEMA.CONIC',
                'BUILDING_DESIGN_SCHEMA.POLYLINE',
                'BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE'] * TYPEOF(
                face_geometry\swept_surface.swept_curve)) = 1);
      wr7 : SIZEOF(QUERY ( vlp_fbnds <* QUERY ( bnds <* SELF.bounds | (
              'BUILDING_DESIGN_SCHEMA.VERTEX_LOOP' IN TYPEOF(bnds.bound)) )
               | (NOT (('BUILDING_DESIGN_SCHEMA.VERTEX_POINT' IN TYPEOF(
               vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex)) AND (
              'BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT' IN TYPEOF(
               vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex\
               vertex_point.vertex_geometry)))) )) = 0;
      wr8 : SIZEOF(QUERY ( bnd <* SELF.bounds | (NOT (SIZEOF([
                'BUILDING_DESIGN_SCHEMA.EDGE_LOOP',
                'BUILDING_DESIGN_SCHEMA.VERTEX_LOOP'] * TYPEOF(bnd.bound))
                = 1)) )) = 0;
      wr9 : SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* SELF.bounds | (
                'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list | (('BUILDING_DESIGN_SCHEMA.SURFACE_CURVE' IN
                TYPEOF(oe.edge_element\edge_curve.edge_geometry)) AND (NOT
                (SIZEOF(QUERY ( sc_ag <* oe.edge_element\edge_curve.
                edge_geometry\surface_curve.associated_geometry | (NOT (
                'BUILDING_DESIGN_SCHEMA.PCURVE' IN TYPEOF(sc_ag))) ))
                = 0))) ))  = 0)) )) = 0;
      wr10: ((NOT ('BUILDING_DESIGN_SCHEMA.SWEPT_SURFACE' IN TYPEOF(
                face_geometry))) OR (NOT ('BUILDING_DESIGN_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 <* SELF.bounds | (
                'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list | (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF(
                oe.edge_element\edge_curve.edge_geometry)) AND (NOT (
                SIZEOF(oe.edge_element\edge_curve.edge_geometry\polyline.
                points) < 3))) )) = 0)) )) = 0);
  END_ENTITY; -- advanced_face

  ENTITY advanced_face_with_thickness_shape_representation
    SUBTYPE OF (shape_representation, solid_model);
    WHERE
      wr1: SIZEOF(SELF.items) = 2;
      wr2: SIZEOF(QUERY ( item <* SELF.items | ((SIZEOF([
                'BUILDING_DESIGN_SCHEMA.MEASURE_REPRESENTATION_ITEM',
                'BUILDING_DESIGN_SCHEMA.LENGTH_MEASURE_WITH_UNIT'] * TYPEOF(
                item)) = 2) AND (item.name = 'thickness')) )) = 1;
      wr3: SIZEOF(QUERY ( item <* SELF.items | (
                'BUILDING_DESIGN_SCHEMA.ADVANCED_FACE' IN TYPEOF(item)) )) =
                 1;
  END_ENTITY; -- advanced_face_with_thickness_shape_representation

  ENTITY advanced_space_boundary_shape_representation
    SUBTYPE OF (shape_representation);
    WHERE
```

```
      wr1: SIZEOF(QUERY ( item <* SELF.items | (NOT ((SIZEOF([
                'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL',
                'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
                'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D'] * TYPEOF(item))

                = 1) OR (item.name = 'reference curve'))) )) = 0;
      wr2: (SIZEOF(QUERY ( item <* SELF.items | (SIZEOF([
                'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL',
                'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM']*TYPEOF(item)) = 1) ))
                = 1) OR ((2 <= SIZEOF(QUERY ( it <* SELF.items | (it.name =
                'reference curve') ))) AND (SIZEOF(QUERY ( it <* SELF.items
                 | (it.name = 'reference curve') )) <= 3));
      wr3: SIZEOF(QUERY ( csh <* QUERY ( item <* SELF.items | (
                'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF(item)) ) | (
                NOT (SIZEOF(QUERY ( fcs <* csh\connected_face_set.cfs_faces
                 | (NOT ('BUILDING_DESIGN_SCHEMA.ADVANCED_FACE' IN TYPEOF(
                fcs))) )) = 0)) )) = 0;
      wr4: SIZEOF(QUERY ( mi <* QUERY ( item <* SELF.items | (
                'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF(item)) ) | (
                NOT (('BUILDING_DESIGN_SCHEMA.' +
                'ADVANCED_SPACE_BOUNDARY_SHAPE_REPRESENTATION') IN TYPEOF(mi
                \mapped_item.mapping_source.mapped_representation))) )) = 0;
      wr5: SIZEOF(QUERY ( it <* SELF.items | ((it.name = 'reference curve')

                AND (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF(it))
                OR ('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(it))
                AND ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN TYPEOF(it\
                trimmed_curve.basis_curve)))))) )) = 0;
  END_ENTITY; -- advanced_space_boundary_shape_representation

  ENTITY advanced_wire_shape_representation
    SUBTYPE OF (shape_representation);
    WHERE
      wr1: SIZEOF(QUERY ( item <* SELF.items | (NOT ((SIZEOF([
                'BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
                'BUILDING_DESIGN_SCHEMA.CIRCLE',
                'BUILDING_DESIGN_SCHEMA.ELLIPSE',
                'BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE',
                'BUILDING_DESIGN_SCHEMA.OFFSET_CURVE_3D',
                'BUILDING_DESIGN_SCHEMA.CURVE_REPLICA',
                'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE',
                'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
                'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D'] * TYPEOF(item))
                = 1) OR (item.name = 'reference curve'))) )) = 0;
      wr2: (SIZEOF(QUERY ( item <* SELF.items | (SIZEOF([
                'BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
                'BUILDING_DESIGN_SCHEMA.CIRCLE',
                'BUILDING_DESIGN_SCHEMA.ELLIPSE',
                'BUILDING_DESIGN_SCHEMA.OFFSET_CURVE_3D',
                'BUILDING_DESIGN_SCHEMA.CURVE_REPLICA',
                'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE',
                'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM']*TYPEOF(item)) = 1) ))
                = 1) OR ((2 <= SIZEOF(QUERY ( it <* SELF.items | (it.name =
                'reference curve') ))) AND (SIZEOF(QUERY ( it <* SELF.items
                 | (it.name = 'reference curve') )) <= 3));
      wr3: SIZEOF(QUERY ( item <* SELF.items | ((SIZEOF([
                'BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
                'BUILDING_DESIGN_SCHEMA.OFFSET_CURVE_3D',
                'BUILDING_DESIGN_SCHEMA.CURVE_REPLICA',
                'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE'] * TYPEOF(item)) = 1)

                AND (NOT valid_advanced_wire_composition(item))) )) = 0;
      wr4: SIZEOF(QUERY ( mi <* QUERY ( item <* SELF.items | (
                'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF(item)) ) | (
```

```
              NOT (('BUILDING_DESIGN_SCHEMA.' +
              'ADVANCED_WIRE_SHAPE_REPRESENTATION') IN TYPEOF(mi\
              mapped_item.mapping_source.mapped_representation))) )) = 0;
    wr5: SIZEOF(QUERY ( it <* SELF.items | ((it.name = 'reference curve')
              AND (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF(it))
              OR (('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(it))
              AND ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN TYPEOF(it\
              trimmed_curve.basis_curve)))))) )) = 0;
END_ENTITY; -- advanced_wire_shape_representation

ENTITY application_context;
    application : text;
  INVERSE
    context_elements : SET [1:?] OF application_context_element FOR
                          frame_of_reference;
END_ENTITY; -- application_context

ENTITY application_context_element
  SUPERTYPE OF (ONEOF (product_context,product_definition_context));
    name               : label;
    frame_of_reference : application_context;
END_ENTITY; -- application_context_element

ENTITY application_protocol_definition;
    status                                 : label;
    application_interpreted_model_schema_name : label;
    application_protocol_year              : year_number;
    application                            : application_context;
END_ENTITY; -- application_protocol_definition

ENTITY approval;
    status : approval_status;
    level  : label;
END_ENTITY; -- approval

ENTITY approval_assignment
  ABSTRACT SUPERTYPE;
    assigned_approval : approval;
END_ENTITY; -- approval_assignment

ENTITY approval_date_time;
    date_time      : date_time_select;
    dated_approval : approval;
END_ENTITY; -- approval_date_time

ENTITY approval_person_organization;
    person_organization : person_organization_select;
    authorized_approval : approval;
    role                : approval_role;
END_ENTITY; -- approval_person_organization

ENTITY approval_role;
    role : label;
END_ENTITY; -- approval_role

ENTITY approval_status;
    name : label;
END_ENTITY; -- approval_status

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

```
ENTITY axis1_placement
  SUBTYPE OF (placement);
    axis : OPTIONAL direction;
  DERIVE
    z : direction := NVL(normalise(axis),direction([0,0,1]));
  WHERE
    wr1: SELF\geometric_representation_item.dim = 3;
END_ENTITY; -- axis1_placement

ENTITY axis2_placement_2d
  SUBTYPE OF (placement);
    ref_direction : OPTIONAL direction;
  DERIVE
    p : LIST [2:2] OF direction := build_2axes(ref_direction);
  WHERE
    wr1: SELF\geometric_representation_item.dim = 2;
END_ENTITY; -- axis2_placement_2d

ENTITY axis2_placement_3d
  SUBTYPE OF (placement);
    axis          : OPTIONAL direction;
    ref_direction : OPTIONAL direction;
  DERIVE
    p : LIST [3:3] OF direction := build_axes(axis,ref_direction);
  WHERE
    wr1: SELF\placement.location.dim = 3;
    wr2: (NOT EXISTS(axis)) OR (axis.dim = 3);
    wr3: (NOT EXISTS(ref_direction)) OR (ref_direction.dim = 3);
    wr4: (NOT EXISTS(axis)) OR (NOT EXISTS(ref_direction)) OR (
              cross_product(axis,ref_direction).magnitude > 0);
END_ENTITY; -- axis2_placement_3d

ENTITY b_spline_curve
  SUPERTYPE OF (ONEOF (uniform_curve,b_spline_curve_with_knots,
     quasi_uniform_curve,bezier_curve) ANDOR rational_b_spline_curve)
  SUBTYPE OF (bounded_curve);
    degree              : INTEGER;
    control_points_list : LIST [2:?] OF cartesian_point;
    curve_form          : b_spline_curve_form;
    closed_curve        : LOGICAL;
    self_intersect      : LOGICAL;
  DERIVE
    upper_index_on_control_points : INTEGER := SIZEOF(
                                       control_points_list) - 1;
    control_points                : ARRAY [0:
                                       upper_index_on_control_points] OF
                                        cartesian_point := list_to_array(
                                       control_points_list,0,
                                       upper_index_on_control_points);
  WHERE
    wr1: ('BUILDING_DESIGN_SCHEMA.UNIFORM_CURVE' IN TYPEOF(SELF)) OR (
              'BUILDING_DESIGN_SCHEMA.QUASI_UNIFORM_CURVE'
              IN TYPEOF(SELF))
              OR ('BUILDING_DESIGN_SCHEMA.BEZIER_CURVE' IN TYPEOF(SELF))
              OR ('BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE_WITH_KNOTS' IN
              TYPEOF(SELF));
END_ENTITY; -- b_spline_curve

ENTITY b_spline_curve_with_knots
  SUBTYPE OF (b_spline_curve);
    knot_multiplicities : LIST [2:?] OF INTEGER;
    knots               : LIST [2:?] OF parameter_value;
    knot_spec           : knot_type;
  DERIVE
```

```
    upper_index_on_knots : INTEGER := SIZEOF(knots);
  WHERE
    wr1: constraints_param_b_spline(degree,upper_index_on_knots,
            upper_index_on_control_points,knot_multiplicities,knots);
    wr2: SIZEOF(knot_multiplicities) = upper_index_on_knots;
END_ENTITY; -- b_spline_curve_with_knots

ENTITY b_spline_surface
  SUPERTYPE OF (ONEOF (b_spline_surface_with_knots,uniform_surface,
      quasi_uniform_surface,bezier_surface) ANDOR
      rational_b_spline_surface)
  SUBTYPE OF (bounded_surface);
    u_degree           : INTEGER;
    v_degree           : INTEGER;
    control_points_list : LIST [2:?] OF LIST [2:?] OF cartesian_point;
    surface_form       : b_spline_surface_form;
    u_closed           : LOGICAL;
    v_closed           : LOGICAL;
    self_intersect     : LOGICAL;
  DERIVE
    u_upper        : INTEGER := SIZEOF(control_points_list) - 1;
    v_upper        : INTEGER := SIZEOF(control_points_list[1]) - 1;
    control_points : ARRAY [0:u_upper] OF ARRAY [0:v_upper] OF
                         cartesian_point := make_array_of_array(
                        control_points_list,0,u_upper,0,v_upper);
  WHERE
    wr1: ('BUILDING_DESIGN_SCHEMA.UNIFORM_SURFACE' IN TYPEOF(SELF)) OR (
            'BUILDING_DESIGN_SCHEMA.QUASI_UNIFORM_SURFACE' IN TYPEOF(
            SELF)) OR ('BUILDING_DESIGN_SCHEMA.BEZIER_SURFACE' IN
            TYPEOF(SELF)) OR (
            'BUILDING_DESIGN_SCHEMA.B_SPLINE_SURFACE_WITH_KNOTS' IN
            TYPEOF(SELF));
END_ENTITY; -- b_spline_surface

ENTITY b_spline_surface_with_knots
  SUBTYPE OF (b_spline_surface);
    u_multiplicities : LIST [2:?] OF INTEGER;
    v_multiplicities : LIST [2:?] OF INTEGER;
    u_knots          : LIST [2:?] OF parameter_value;
    v_knots          : LIST [2:?] OF parameter_value;
    knot_spec        : knot_type;
  DERIVE
    knot_u_upper : INTEGER := SIZEOF(u_knots);
    knot_v_upper : INTEGER := SIZEOF(v_knots);
  WHERE
    wr1: constraints_param_b_spline(SELF\b_spline_surface.u_degree,
            knot_u_upper,SELF\b_spline_surface.u_upper,u_multiplicities,
            u_knots);
    wr2: constraints_param_b_spline(SELF\b_spline_surface.v_degree,
            knot_v_upper,SELF\b_spline_surface.v_upper,v_multiplicities,
            v_knots);
    wr3: SIZEOF(u_multiplicities) = knot_u_upper;
    wr4: SIZEOF(v_multiplicities) = knot_v_upper;
END_ENTITY; -- b_spline_surface_with_knots

ENTITY bezier_curve
  SUBTYPE OF (b_spline_curve);
END_ENTITY; -- bezier_curve

ENTITY bezier_surface
  SUBTYPE OF (b_spline_surface);
END_ENTITY; -- bezier_surface

ENTITY block
```

```
    SUBTYPE OF (geometric_representation_item);
      position : axis2_placement_3d;
      x        : positive_length_measure;
      y        : positive_length_measure;
      z        : positive_length_measure;
  END_ENTITY; -- block

  ENTITY boolean_result
    SUBTYPE OF (geometric_representation_item);
      operator       : boolean_operator;
      first_operand  : boolean_operand;
      second_operand : boolean_operand;
  END_ENTITY; -- boolean_result

  ENTITY boundary_curve
    SUBTYPE OF (composite_curve_on_surface);
    WHERE
      wr1: SELF\composite_curve.closed_curve;
  END_ENTITY; -- boundary_curve

  ENTITY bounded_curve
    SUPERTYPE OF (ONEOF (polyline,b_spline_curve,trimmed_curve,
        composite_curve))
    SUBTYPE OF (curve);
  END_ENTITY; -- bounded_curve

  ENTITY bounded_surface
    SUPERTYPE OF (ONEOF (b_spline_surface,rectangular_trimmed_surface,
        curve_bounded_surface,rectangular_composite_surface))
    SUBTYPE OF (surface);
  END_ENTITY; -- bounded_surface

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

  ENTITY building
    SUBTYPE OF (product_definition);
    WHERE
      wr1: SIZEOF(QUERY ( pdr <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
           'PRODUCT_DEFINITION_RELATIONSHIP.RELATING_PRODUCT_DEFINITION')
                | ('BUILDING_DESIGN_SCHEMA.BUILDING_SECTION' IN TYPEOF(pdr.
              related_product_definition)) )) >= 1;
      wr2: SIZEOF(QUERY ( pdr <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
           'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION')
                | ('BUILDING_DESIGN_SCHEMA.BUILDING_COMPLEX' IN TYPEOF(pdr.
              relating_product_definition)) )) = 1;
      wr3: SIZEOF (QUERY (bdpa <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
           'BUILDING_DESIGN_PERSON_ASSIGNMENT.ITEMS') |
           bdpa.role.name = 'owner')) +
           SIZEOF (QUERY (bdoa <* USEDIN (SELF, 'BUILDING_DESIGN_SCHEMA.' +
           'BUILDING_DESIGN_ORGANIZATION_ASSIGNMENT.ITEMS') |
           bdoa.role.name = 'owner')) = 1;
  END_ENTITY; -- building

  ENTITY building_complex
    SUBTYPE OF (product_definition);
    WHERE
      wr1: SIZEOF(QUERY ( pdr <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
           'PRODUCT_DEFINITION_RELATIONSHIP.RELATING_PRODUCT_DEFINITION')
                | ((pdr.name = 'in building complex') AND (
              'BUILDING_DESIGN_SCHEMA.BUILDING' IN TYPEOF(pdr.
              related_product_definition))) )) >= 1;
```

```
    wr2: (SIZEOF(QUERY ( pa <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
              'BUILDING_DESIGN_PERSON_ASSIGNMENT.ITEMS') | (pa.role.name =
               'owner') )) + SIZEOF(QUERY ( oa <* USEDIN(SELF,
              'BUILDING_DESIGN_SCHEMA.' +
              'BUILDING_DESIGN_ORGANIZATION_ASSIGNMENT.ITEMS') | (oa.role.
              name = 'owner') ))) = 1;
END_ENTITY; -- building_complex

ENTITY building_component_classification_assignment
  SUBTYPE OF (group_assignment);
    items : SET [1:?] OF building_component_classification_item;
END_ENTITY; -- building_component_classification_assignment

ENTITY building_component_classification_group
  SUBTYPE OF (group);
  WHERE
    wr1: SIZEOF(QUERY ( ga <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
              'GROUP_ASSIGNMENT.ASSIGNED_GROUP') | ((
              'BUILDING_DESIGN_SCHEMA.' +
              'BUILDING_COMPONENT_CLASSIFICATION_ASSIGNMENT')
              IN TYPEOF(ga)) ))  >= 1;
END_ENTITY; -- building_component_classification_group

ENTITY building_design_approval
  SUBTYPE OF (approval_assignment);
    items : SET [1:?] OF building_design_approval_item;
  WHERE
    wr1: SIZEOF(QUERY ( i <* SELF.items | (SIZEOF([
              'BUILDING_DESIGN_SCHEMA.BUILDING',
              'BUILDING_DESIGN_SCHEMA.BUILDING_LEVEL',
              'BUILDING_DESIGN_SCHEMA.BUILDING_SECTION',
              'BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT_ASSEMBLY']
              * TYPEOF(i))=1) ))
              = 0;
END_ENTITY; -- building_design_approval

ENTITY building_design_date_assignment
  SUBTYPE OF (date_assignment);
    items : SET [1:?] OF building_design_date_item;
END_ENTITY; -- building_design_date_assignment

ENTITY building_design_organization_assignment
  SUBTYPE OF (organization_assignment);
    items : SET [1:?] OF building_design_organization_item;
END_ENTITY; -- building_design_organization_assignment

ENTITY building_design_person_and_organization_assignment
  SUBTYPE OF (person_and_organization_assignment);
    items : SET [1:?] OF building_design_person_and_organization_item;
END_ENTITY; -- building_design_person_and_organization_assignment

ENTITY building_design_person_assignment
  SUBTYPE OF (person_assignment);
    items : SET [1:?] OF building_design_person_item;
END_ENTITY; -- building_design_person_assignment

ENTITY building_document_reference
  SUBTYPE OF (document_reference);
    items : SET [1:?] OF building_document_item;
END_ENTITY; -- building_document_reference

ENTITY building_element
  SUBTYPE OF (product_definition);
  WHERE
```

```
      wr1: SIZEOF(QUERY ( pdr <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
           'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION')
                | ('BUILDING_DESIGN_SCHEMA.BUILDING_SECTION' IN TYPEOF(pdr.
               relating_product_definition)) )) = 1;
      wr2: SIZEOF(USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
               'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
      wr3: SIZEOF(QUERY ( pd <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
               'PROPERTY_DEFINITION.DEFINITION') | (SIZEOF(QUERY ( sa <*
               USEDIN(pd,'BUILDING_DESIGN_SCHEMA.' +
               'SHAPE_ASPECT.OF_SHAPE') | ((
               'BUILDING_DESIGN_SCHEMA.POSITIVE_COMPONENT' IN TYPEOF(sa))
               AND (sa.description = 'main component')) )) = 1) )) = 1;
  END_ENTITY; -- building_element

  ENTITY building_element_assembly
    SUBTYPE OF (product_definition);
    WHERE
      wr1: SIZEOF(QUERY ( pd <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
               'PROPERTY_DEFINITION.DEFINITION') | (
               'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN TYPEOF(
               pd)) )) = 0;
      wr2: SIZEOF(QUERY ( acu <* QUERY ( pdr <* USEDIN(SELF,
               'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_RELATIONSHIP.' +
               'RELATING_PRODUCT_DEFINITION') | (
               'BUILDING_DESIGN_SCHEMA.ASSEMBLY_COMPONENT_USAGE' IN TYPEOF(
               pdr)) ) | (SIZEOF(TYPEOF(acu.related_product_definition) * [
               'BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
               'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
               'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
               'BUILDING_DESIGN_SCHEMA.SPACE_ELEMENT',
               'BUILDING_DESIGN_SCHEMA.STRUCTURE_ENCLOSURE_ELEMENT',
               'BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT_ASSEMBLY']) = 1) ))
               >= 1;
  END_ENTITY; -- building_element_assembly

  ENTITY building_element_group
    SUBTYPE OF (product_definition);
    WHERE
      wr1: SIZEOF(QUERY ( pd <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
               'PROPERTY_DEFINITION.DEFINITION') | (
               'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN TYPEOF(
               pd)) )) = 0;
      wr2: SIZEOF(QUERY ( pdr <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
           'PRODUCT_DEFINITION_RELATIONSHIP.RELATING_PRODUCT_DEFINITION')
                | (SIZEOF(TYPEOF(pdr.related_product_definition) * [
               'BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
               'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
               'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
               'BUILDING_DESIGN_SCHEMA.SPACE_ELEMENT',
               'BUILDING_DESIGN_SCHEMA.STRUCTURE_ENCLOSURE_ELEMENT',
               'BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT_GROUP']) = 1) )) >=
                1;
  END_ENTITY; -- building_element_group

  ENTITY building_item_identification_assignment
    SUBTYPE OF (name_assignment);
      item : identified_item;
  END_ENTITY; -- building_item_identification_assignment

  ENTITY building_level
    SUBTYPE OF (product_definition);
    WHERE
      wr1: SIZEOF(QUERY ( pdr <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
           'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION')
```

```
              | ('BUILDING_DESIGN_SCHEMA.BUILDING_SECTION' IN TYPEOF(pdr.
            relating_product_definition)) )) = 1;
    wr2: SIZEOF(USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
            'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) <= 1;
    wr3: SIZEOF(QUERY ( pd <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
            'PROPERTY_DEFINITION.DEFINITION') | (NOT (SIZEOF(
            QUERY ( pdr <* USEDIN(pd,'BUILDING_DESIGN_SCHEMA.' +
            'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') | (NOT (
            SIZEOF(TYPEOF(pdr.used_representation) * [
            'BUILDING_DESIGN_SCHEMA.' +
            'ADVANCED_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
            'BUILDING_DESIGN_SCHEMA.' +
            'ELEMENTARY_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
            'BUILDING_DESIGN_SCHEMA.' +
            'FACETED_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
            'BUILDING_DESIGN_SCHEMA.' +
            'GROUND_FACE_SPACE_BOUNDARY_SHAPE_REPRESENTATION']) = 1)) ))
            = 0)) )) = 0;
END_ENTITY; -- building_level

ENTITY building_section
  SUBTYPE OF (product_definition);
  WHERE
    wr1: SIZEOF(QUERY ( pdr <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
         'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION')
              | ('BUILDING_DESIGN_SCHEMA.BUILDING' IN TYPEOF(pdr.
            relating_product_definition)) )) = 1;
    wr2: SIZEOF(USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
            'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
    wr3: SIZEOF(QUERY ( pdr <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
         'PRODUCT_DEFINITION_RELATIONSHIP.RELATING_PRODUCT_DEFINITION')
              | (SIZEOF(TYPEOF(pdr.related_product_definition) * [
            'BUILDING_DESIGN_SCHEMA.BUILDING_LEVEL',
            'BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
            'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
            'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
            'BUILDING_DESIGN_SCHEMA.SPACE_ELEMENT',
            'BUILDING_DESIGN_SCHEMA.STRUCTURE_ENCLOSURE_ELEMENT'])=1) ))
            >= 1;
END_ENTITY; -- building_section

ENTITY calendar_date
  SUBTYPE OF (date);
    day_component   : day_in_month_number;
    month_component : month_in_year_number;
  WHERE
    wr1: valid_calendar_date(SELF);
END_ENTITY; -- calendar_date

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

ENTITY cartesian_transformation_operator
  SUPERTYPE OF (cartesian_transformation_operator_3d)
  SUBTYPE OF (geometric_representation_item,
      functionally_defined_transformation);
    axis1        : OPTIONAL direction;
    axis2        : OPTIONAL direction;
    local_origin : cartesian_point;
    scale        : OPTIONAL REAL;
  DERIVE
    scl : REAL := NVL(scale,1);
```

```
    WHERE
      wr1: scl > 0;
  END_ENTITY; -- cartesian_transformation_operator

  ENTITY cartesian_transformation_operator_3d
    SUBTYPE OF (cartesian_transformation_operator);
      axis3 : OPTIONAL direction;
    DERIVE
      u : LIST [3:3] OF direction := base_axis(3,SELF\
             cartesian_transformation_operator.axis1,SELF\
             cartesian_transformation_operator.axis2,axis3);
    WHERE
      wr1: SELF\cartesian_transformation_operator.dim = 3;
  END_ENTITY; -- cartesian_transformation_operator_3d

  ENTITY change
    SUBTYPE OF (action_assignment);
      items : SET [1:?] OF building_design_change_item;
  END_ENTITY; -- change

  ENTITY characterized_object;
      name        : label;
      description : text;
  END_ENTITY; -- characterized_object

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

  ENTITY classification_table
    SUBTYPE OF (document_usage_constraint);
    WHERE
      wr1: SIZEOF(QUERY ( bddr <* QUERY ( dr <* USEDIN(SELF.source,
           'BUILDING_DESIGN_SCHEMA.DOCUMENT_REFERENCE.ASSIGNED_DOCUMENT')
                | ('BUILDING_DESIGN_SCHEMA.BUILDING_DOCUMENT_REFERENCE' IN
              TYPEOF(dr)) ) | (NOT (SIZEOF(QUERY ( item <* bddr.items | (
              NOT (SIZEOF(TYPEOF(item) * [
              'BUILDING_DESIGN_SCHEMA.PRODUCT_CATEGORY',
              'BUILDING_DESIGN_SCHEMA.' +
              'BUILDING_COMPONENT_CLASSIFICATION_GROUP'])
              = 1)) ))  = 0)) )) = 0;
  END_ENTITY; -- classification_table

  ENTITY closed_shell
    SUBTYPE OF (connected_face_set);
  END_ENTITY; -- closed_shell

  ENTITY composite_curve
    SUBTYPE OF (bounded_curve);
      segments       : LIST [1:?] OF composite_curve_segment;
      self_intersect : LOGICAL;
    DERIVE
      n_segments   : INTEGER := SIZEOF(segments);
      closed_curve : LOGICAL := segments[n_segments].transition <>
                        discontinuous;
    WHERE
      wr1: ((NOT closed_curve) AND (SIZEOF(QUERY ( temp <* segments | (
                temp.transition = discontinuous) )) = 1)) OR (closed_curve
                AND (SIZEOF(QUERY ( temp <* segments | (temp.transition =
                discontinuous) )) = 0));
  END_ENTITY; -- composite_curve

  ENTITY composite_curve_on_surface
```

```
    SUPERTYPE OF (boundary_curve)
    SUBTYPE OF (composite_curve);
    DERIVE
      basis_surface : SET [0:2] OF surface := get_basis_surface(SELF);
    WHERE
      wr1: SIZEOF(basis_surface) > 0;
      wr2: constraints_composite_curve_on_surface(SELF);
  END_ENTITY; -- composite_curve_on_surface

  ENTITY composite_curve_segment;
      transition   : transition_code;
      same_sense   : BOOLEAN;
      parent_curve : curve;
    INVERSE
      using_curves : BAG [1:?] OF composite_curve FOR segments;
    WHERE
      wr1: 'BUILDING_DESIGN_SCHEMA.BOUNDED_CURVE' IN TYPEOF(parent_curve);
  END_ENTITY; -- composite_curve_segment

  ENTITY conic
    SUPERTYPE OF (ONEOF (circle,ellipse,hyperbola,parabola))
    SUBTYPE OF (curve);
      position : axis2_placement;
  END_ENTITY; -- conic

  ENTITY conical_surface
    SUBTYPE OF (elementary_surface);
      radius     : length_measure;
      semi_angle : plane_angle_measure;
    WHERE
      wr1: radius >= 0;
  END_ENTITY; -- conical_surface

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

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

  ENTITY csg_solid
    SUBTYPE OF (solid_model);
      tree_root_expression : csg_select;
  END_ENTITY; -- csg_solid

  ENTITY curve
    SUPERTYPE OF (ONEOF (line,conic,pcurve,surface_curve,offset_curve_3d,
        curve_replica))
    SUBTYPE OF (geometric_representation_item);
  END_ENTITY; -- curve

  ENTITY curve_bounded_surface
    SUBTYPE OF (bounded_surface);
      basis_surface  : surface;
      boundaries     : SET [1:?] OF boundary_curve;
      implicit_outer : BOOLEAN;
    WHERE
      wr1: NOT (implicit_outer AND (
                'BUILDING_DESIGN_SCHEMA.OUTER_BOUNDARY_CURVE' IN TYPEOF(
```

```
            boundaries)));
    wr2: (NOT implicit_outer) OR (
            'BUILDING_DESIGN_SCHEMA.BOUNDED_SURFACE' IN TYPEOF(
            basis_surface));
    wr3: SIZEOF(QUERY ( temp <* boundaries | (
            'BUILDING_DESIGN_SCHEMA.OUTER_BOUNDARY_CURVE'
            IN TYPEOF(temp)) )) <= 1;
    wr4: SIZEOF(QUERY ( temp <* boundaries | (temp\
            composite_curve_on_surface.basis_surface[1] :<>: SELF.
            basis_surface) )) = 0;
END_ENTITY; -- curve_bounded_surface

ENTITY curve_replica
  SUBTYPE OF (curve);
    parent_curve   : curve;
    transformation : cartesian_transformation_operator;
  WHERE
    wr1: transformation.dim = parent_curve.dim;
    wr2: acyclic_curve_replica(SELF,parent_curve);
END_ENTITY; -- curve_replica

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

ENTITY date
  SUPERTYPE OF (ONEOF (calendar_date,ordinal_date,
      week_of_year_and_day_date));
    year_component : year_number;
END_ENTITY; -- date

ENTITY date_assignment
  ABSTRACT SUPERTYPE;
    assigned_date : date;
    role          : date_role;
END_ENTITY; -- date_assignment

ENTITY date_role;
    name : label;
END_ENTITY; -- date_role

ENTITY definitional_representation
  SUBTYPE OF (representation);
  WHERE
    wr1: 'BUILDING_DESIGN_SCHEMA.PARAMETRIC_REPRESENTATION_CONTEXT' IN
            TYPEOF(SELF\representation.context_of_items);
END_ENTITY; -- definitional_representation

ENTITY degenerate_toroidal_surface
  SUBTYPE OF (toroidal_surface);
    select_outer : BOOLEAN;
  WHERE
    wr1: major_radius < minor_radius;
END_ENTITY; -- degenerate_toroidal_surface

ENTITY derived_unit;
    elements : SET [1:?] OF derived_unit_element;
  WHERE
    wr1: (SIZEOF(elements) > 1) OR ((SIZEOF(elements) = 1) AND (elements
            [1].exponent <> 1));
END_ENTITY; -- derived_unit

ENTITY derived_unit_element;
```

```
    unit     : named_unit;
    exponent : REAL;
END_ENTITY; -- derived_unit_element

ENTITY descriptive_representation_item
  SUBTYPE OF (representation_item);
    description : text;
END_ENTITY; -- descriptive_representation_item

ENTITY dimensional_exponents;
    length_exponent                   : REAL;
    mass_exponent                     : REAL;
    time_exponent                     : REAL;
    electric_current_exponent         : REAL;
    thermodynamic_temperature_exponent : REAL;
    amount_of_substance_exponent      : REAL;
    luminous_intensity_exponent       : REAL;
END_ENTITY; -- dimensional_exponents

ENTITY direction
  SUBTYPE OF (geometric_representation_item);
    direction_ratios : LIST [2:3] OF REAL;
  WHERE
    wr1: SIZEOF(QUERY ( tmp <* direction_ratios | (tmp <> 0) )) > 0;
END_ENTITY; -- direction

ENTITY document;
    id          : identifier;
    name        : label;
    description : text;
    kind        : document_type;
  UNIQUE
    ur1 : id;
END_ENTITY; -- document

ENTITY document_reference
  ABSTRACT SUPERTYPE;
    assigned_document : document;
    source            : label;
END_ENTITY; -- document_reference

ENTITY document_type;
    product_data_type : label;
END_ENTITY; -- document_type

ENTITY document_usage_constraint;
    source                : document;
    subject_element       : label;
    subject_element_value : text;
END_ENTITY; -- document_usage_constraint

ENTITY edge
  SUPERTYPE OF (ONEOF (edge_curve,oriented_edge))
  SUBTYPE OF (topological_representation_item);
    edge_start : vertex;
    edge_end   : vertex;
END_ENTITY; -- edge

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

```
ENTITY edge_loop
  SUBTYPE OF (loop, path);
  DERIVE
    ne : INTEGER := SIZEOF(SELF\path.edge_list);
  WHERE
    wr1: SELF\path.edge_list[1].edge_start :=: SELF\path.edge_list[ne].
         edge_end;
END_ENTITY; -- edge_loop

ENTITY elementary_csg_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    wr1: SIZEOF(QUERY ( item <* SELF.items | (NOT ((SIZEOF([
         'BUILDING_DESIGN_SCHEMA.CSG_SOLID',
         'BUILDING_DESIGN_SCHEMA.EXTRUDED_AREA_SOLID',
         'BUILDING_DESIGN_SCHEMA.REVOLVED_AREA_SOLID',
         'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D',
         'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF(item)) = 1)
         OR (item.name = 'reference curve'))) )) = 0;
    wr2: (SIZEOF(QUERY ( item <* SELF.items | (SIZEOF([
         'BUILDING_DESIGN_SCHEMA.CSG_SOLID',
         'BUILDING_DESIGN_SCHEMA.EXTRUDED_AREA_SOLID',
         'BUILDING_DESIGN_SCHEMA.REVOLVED_AREA_SOLID',
         'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM']*TYPEOF(item)) = 1) ))
         >= 1) OR ((2 <= SIZEOF(QUERY ( it <* SELF.items | (it.name =
          'reference curve') ))) AND (SIZEOF(QUERY ( it <* SELF.items
          | (it.name = 'reference curve') )) <= 3));
    wr3: SIZEOF(QUERY ( item <* SELF.items | ((
         'BUILDING_DESIGN_SCHEMA.CSG_SOLID' IN TYPEOF(item)) AND (
         NOT valid_elementary_csg_tree(item\csg_solid.
         tree_root_expression))) )) = 0;
    wr4: SIZEOF(QUERY ( mi <* QUERY ( item <* SELF.items | (
         'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF(item)) ) | (
         NOT
        ('BUILDING_DESIGN_SCHEMA.ELEMENTARY_CSG_SHAPE_REPRESENTATION'
         IN TYPEOF(mi\mapped_item.mapping_source.
         mapped_representation))) )) = 0;
    wr5: SIZEOF(QUERY ( it <* SELF.items | ((it.name = 'reference curve')
         AND (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF(it))
         OR (('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(it))
         AND ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN TYPEOF(it\
         trimmed_curve.basis_curve)))))) )) = 0;
END_ENTITY; -- elementary_csg_shape_representation

ENTITY elementary_face_with_thickness_shape_representation
  SUBTYPE OF (shape_representation, solid_model);
  WHERE
    wr1: SIZEOF(SELF.items) = 2;
    wr2: SIZEOF(QUERY ( item <* SELF.items | ((SIZEOF([
         'BUILDING_DESIGN_SCHEMA.MEASURE_REPRESENTATION_ITEM',
         'BUILDING_DESIGN_SCHEMA.LENGTH_MEASURE_WITH_UNIT'] * TYPEOF(
         item)) = 2) AND (item.name = 'thickness')) )) = 1;
    wr3: SIZEOF(QUERY ( item <* SELF.items | (
         'BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF(item)) )) =
         1;
    wr4: SIZEOF(QUERY ( item <* SELF.items | ((NOT (
         'BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF(item))) OR (
         'BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF(item\
         face_surface.face_geometry))) )) = 0;
    wr5: SIZEOF(QUERY ( item <* SELF.items | ((
         'BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF(item)) AND (
         NOT (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* item\face.
         bounds | ('BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.
         bound)) ) | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
```

```
            edge_list | (NOT ('BUILDING_DESIGN_SCHEMA.EDGE_CURVE' IN
            TYPEOF(oe.edge_element))) )) = 0)) )) = 0))) )) = 0;
    wr6: SIZEOF(QUERY ( item <* SELF.items | ((
            'BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF(item)) AND (
            NOT (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* item\face.
            bounds | ('BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.
            bound)) ) | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
            edge_list | (NOT (SIZEOF(['BUILDING_DESIGN_SCHEMA.LINE',
            'BUILDING_DESIGN_SCHEMA.CONIC',
            'BUILDING_DESIGN_SCHEMA.POLYLINE',
            'BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE'] * TYPEOF(oe.
            edge_element\edge_curve.edge_geometry)) = 1)) )) = 0)) )) =
            0))) )) = 0;
    wr7: SIZEOF(QUERY ( item <* SELF.items | ((
            'BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF(item)) AND (
            NOT (SIZEOF(QUERY ( vlp_fbnds <* QUERY ( bnds <* item\face.
            bounds | ('BUILDING_DESIGN_SCHEMA.VERTEX_LOOP' IN TYPEOF(
            bnds.bound)) ) | (NOT ((
            'BUILDING_DESIGN_SCHEMA.VERTEX_POINT' IN TYPEOF(vlp_fbnds\
            face_bound.bound\vertex_loop.loop_vertex)) AND (
            'BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT' IN TYPEOF(vlp_fbnds
            \face_bound.bound\vertex_loop.loop_vertex\vertex_point.
            vertex_geometry)))) )) = 0))) )) = 0;
END_ENTITY; -- elementary_face_with_thickness_shape_representation

ENTITY elementary_geometric_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    wr1 : SIZEOF(QUERY ( item <* SELF.items | (NOT ((SIZEOF([
            'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP',
            'BUILDING_DESIGN_SCHEMA.FACETED_BREP',
            'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
            'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D'] *
            TYPEOF(item))
            = 1) OR (item.name = 'reference curve'))) )) = 0;
    wr2 : (SIZEOF(QUERY ( item <* SELF.items | (SIZEOF([
            'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP',
            'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM']*TYPEOF(item)) = 1)))
            >= 1) OR ((2 <= SIZEOF(QUERY ( it <* SELF.items | (it.name
            = 'reference curve') ))) AND (SIZEOF(QUERY ( it <* SELF.
            items | (it.name = 'reference curve') )) <= 3));
    wr3 : SIZEOF(QUERY ( msb <* QUERY ( item <* SELF.items | (
            'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP'
             IN TYPEOF(item)) )
             | (NOT (SIZEOF(QUERY ( csh <* msb_shells(msb) | (NOT (
            SIZEOF(QUERY ( fcs <* csh.cfs_faces | (NOT (
            'BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF(fcs))) )) =
             0)) )) = 0)) )) = 0;
    wr4 : SIZEOF(QUERY ( msb <* QUERY ( item <* SELF.items | (
            'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP'
             IN TYPEOF(item)) )
             | (NOT (SIZEOF(QUERY ( csh <* msb_shells(msb) | (NOT (
            SIZEOF(QUERY ( fcs <* csh\connected_face_set.cfs_faces | (
            NOT ('BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF(
            fcs\face_surface.face_geometry))) )) = 0)) )) = 0)) )) = 0;
    wr5 : SIZEOF(QUERY ( msb <* QUERY ( item <* SELF.items | (
            'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP'
            IN TYPEOF(item)) )
             | (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
             | ('BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN
             TYPEOF(bnds.bound)) )
             | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
```

```
          edge_list | (NOT ('BUILDING_DESIGN_SCHEMA.EDGE_CURVE' IN
          TYPEOF(oe.edge_element))) )) = 0)) )) = 0)) )) = 0)) ))
          = 0)) )) = 0;
     wr6 : SIZEOF(QUERY ( msb <* QUERY ( item <* SELF.items | (
          'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP'
          IN TYPEOF(item)) )
          | (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
           | ('BUILDING_DESIGN_SCHEMA.EDGE_LOOP'
          IN TYPEOF(bnds.bound)) )
           | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
          edge_list | (NOT (SIZEOF(['BUILDING_DESIGN_SCHEMA.LINE',
          'BUILDING_DESIGN_SCHEMA.CONIC',
          'BUILDING_DESIGN_SCHEMA.POLYLINE',
          'BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE'] * TYPEOF(oe.
          edge_element\edge_curve.edge_geometry)) = 1)) )) = 0)) )) =
           0)) )) = 0)) )) = 0)) )) = 0;
     wr7 : SIZEOF(QUERY ( msb <* QUERY ( item <* SELF.items | (
          'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP'
          IN TYPEOF(item)) )
           | (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
           | ('BUILDING_DESIGN_SCHEMA.EDGE_LOOP'
          IN TYPEOF(bnds.bound)) )
           | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
          edge_list | (NOT (('BUILDING_DESIGN_SCHEMA.VERTEX_POINT' IN
           TYPEOF(oe.edge_start)) AND (
          'BUILDING_DESIGN_SCHEMA.VERTEX_POINT'
          IN TYPEOF(oe.edge_end)))) ))
          = 0)) )) = 0)) )) = 0)) )) = 0;
     wr8 : SIZEOF(QUERY ( msb <* QUERY ( item <* SELF.items | (
          'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP'
          IN TYPEOF(item)) )
           | (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
           | ('BUILDING_DESIGN_SCHEMA.EDGE_LOOP'
          IN TYPEOF(bnds.bound)) )
           | ('BUILDING_DESIGN_SCHEMA.ORIENTED_PATH' IN TYPEOF(
          elp_fbnds.bound)) )) = 0)) )) = 0)) )) = 0)) )) = 0;
     wr9 : SIZEOF(QUERY ( msb <* QUERY ( item <* SELF.items | (
          'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP'
          IN TYPEOF(item)) )
           | ('BUILDING_DESIGN_SCHEMA.ORIENTED_CLOSED_SHELL' IN
          TYPEOF(msb\manifold_solid_brep.outer)) )) = 0;
     wr10: SIZEOF(QUERY ( brv <* QUERY ( item <* SELF.items | (
          'BUILDING_DESIGN_SCHEMA.BREP_WITH_VOIDS' IN TYPEOF(item)) )
           | (NOT (SIZEOF(QUERY ( csh <* brv\brep_with_voids.voids |
          csh\oriented_closed_shell.orientation )) = 0)) )) = 0;
     wr11: SIZEOF(QUERY ( mi <* QUERY ( item <* SELF.items | (
          'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF(item)) ) | (
          NOT (('BUILDING_DESIGN_SCHEMA.' +
          'ELEMENTARY_GEOMETRIC_SHAPE_REPRESENTATION') IN TYPEOF(mi\
          mapped_item.mapping_source.mapped_representation))) )) = 0;
     wr12: SIZEOF(QUERY ( msb <* QUERY ( item <* SELF.items | (
          'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP'
          IN TYPEOF(item)) )
           | (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
           | ('BUILDING_DESIGN_SCHEMA.VERTEX_LOOP' IN TYPEOF(bnds.
          bound)) ) | (NOT (('BUILDING_DESIGN_SCHEMA.VERTEX_POINT'
```

```
            IN TYPEOF(vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex))
                AND ('BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT' IN TYPEOF(
                vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex\
                vertex_point.vertex_geometry)))) ))=0))))=0)) )) = 0)) ))
                = 0;
    wr13: SIZEOF(QUERY ( it <* SELF.items |
                ((it.name = 'reference curve')
                AND (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF(it))

                OR (('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(it))
                AND ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN TYPEOF(it\
                trimmed_curve.basis_curve)))))) )) = 0;
END_ENTITY; -- elementary_geometric_shape_representation

ENTITY elementary_space_boundary_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    wr1 : SIZEOF(QUERY ( item <* SELF.items | (NOT ((SIZEOF([
                'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL',
                'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
                'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D'] *
                TYPEOF(item))
                = 1) OR (item.name = 'reference curve'))) )) = 0;
    wr2 : (SIZEOF(QUERY ( item <* SELF.items | (SIZEOF([
                'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL',
                'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM']*TYPEOF(item))=1) ))
                = 1) OR ((2 <= SIZEOF(QUERY ( it <* SELF.items | (it.name =
                 'reference curve') ))) AND
                (SIZEOF(QUERY ( it <* SELF.items
                 | (it.name = 'reference curve') )) <= 3));
    wr3 : SIZEOF(QUERY ( mi <* QUERY ( item <* SELF.items | (
                'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF(item)) ) | (
                NOT (('BUILDING_DESIGN_SCHEMA.' +
                'ELEMENTARY_SPACE_BOUNDARY_SHAPE_REPRESENTATION') IN
             TYPEOF(mi\mapped_item.mapping_source.mapped_representation)))))
                = 0;
    wr4 : SIZEOF(QUERY ( csh <* QUERY ( item <* SELF.items | (
                'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF(item)) ) |
                (NOT (SIZEOF(QUERY ( fcs <* csh\closed_shell.cfs_faces | (
              NOT ('BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF(fcs))) ))
                = 0)) )) = 0;
    wr5 : SIZEOF(QUERY ( csh <* QUERY ( item <* SELF.items | (
                'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF(item)) ) |
                (NOT (SIZEOF(QUERY ( fcs <* csh\closed_shell.cfs_faces | (
                NOT ('BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF(
                fcs\face_surface.face_geometry))) )) = 0)) )) = 0;
    wr6 : SIZEOF(QUERY ( csh <* QUERY ( item <* SELF.items | (
                'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF(item)) ) |
                (NOT (SIZEOF(QUERY ( fcs <* csh\closed_shell.cfs_faces | (
                NOT (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* fcs.bounds
                 | ('BUILDING_DESIGN_SCHEMA.EDGE_LOOP'
                IN TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list | (NOT ('BUILDING_DESIGN_SCHEMA.EDGE_CURVE' IN
                TYPEOF(oe.edge_element))) )) = 0)) )) = 0)) )) = 0)) ))
                = 0;
    wr7 : SIZEOF(QUERY ( csh <* QUERY ( item <* SELF.items | (
                'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF(item)) ) |
                (NOT (SIZEOF(QUERY ( fcs <* csh\closed_shell.cfs_faces | (
                NOT (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* fcs.bounds
                 | ('BUILDING_DESIGN_SCHEMA.EDGE_LOOP'
                IN TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list | (NOT (SIZEOF(['BUILDING_DESIGN_SCHEMA.LINE',
```

```
                'BUILDING_DESIGN_SCHEMA.CONIC',
                'BUILDING_DESIGN_SCHEMA.POLYLINE',
                'BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE'] * TYPEOF(oe.
                edge_element\edge_curve.edge_geometry)) = 1)) )) = 0)) )) =
                 0)) )) = 0)) )) = 0;
      wr8 : SIZEOF(QUERY ( csh <* QUERY ( item <* SELF.items | (
                'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF(item)) ) |
                (NOT (SIZEOF(QUERY ( fcs <* csh\closed_shell.cfs_faces | (
                NOT (SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* fcs.bounds
                 | ('BUILDING_DESIGN_SCHEMA.EDGE_LOOP'
                IN TYPEOF(bnds.bound)) )
                 | (NOT (SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.
                edge_list | (NOT (('BUILDING_DESIGN_SCHEMA.VERTEX_POINT' IN
                 TYPEOF(oe.edge_start)) AND (
                'BUILDING_DESIGN_SCHEMA.VERTEX_POINT'
                IN TYPEOF(oe.edge_end)))) ))
                = 0)) )) = 0)) )) = 0)) )) = 0;
      wr9 : SIZEOF(QUERY ( csh <* QUERY ( item <* SELF.items | (
                'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF(item)) ) |
                (NOT (SIZEOF(QUERY ( fcs <* csh\connected_face_set.
                cfs_faces | (NOT (SIZEOF(QUERY ( elp_fbnds <*
                QUERY ( bnds <* fcs.bounds | (
                'BUILDING_DESIGN_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) )
                 | ('BUILDING_DESIGN_SCHEMA.ORIENTED_PATH' IN TYPEOF(
                elp_fbnds.bound)) )) = 0)) )) = 0)) )) = 0;
      wr10: SIZEOF(QUERY ( csh <* QUERY ( item <* SELF.items | (
                'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF(item)) ) |
                (NOT (SIZEOF(QUERY ( fcs <* csh\closed_shell.cfs_faces | (
                NOT (SIZEOF(QUERY ( vlp_fbnds <* QUERY ( bnds <* fcs.bounds
                 | ('BUILDING_DESIGN_SCHEMA.VERTEX_LOOP' IN TYPEOF(bnds.
                bound)) ) | (NOT (('BUILDING_DESIGN_SCHEMA.VERTEX_POINT' IN
                TYPEOF(vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex))

                AND ('BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT' IN TYPEOF(
                vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex\
                vertex_point.vertex_geometry)))) )) = 0)) )) = 0)) )) = 0;
      wr11: SIZEOF(QUERY ( it <* SELF.items |
                ((it.name = 'reference curve')
                AND (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE'IN TYPEOF(it))
                OR (('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(it))
                AND ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN TYPEOF(it\
                trimmed_curve.basis_curve)))))) )) = 0;
  END_ENTITY; -- elementary_space_boundary_shape_representation

  ENTITY elementary_surface
    SUPERTYPE OF (ONEOF (plane,cylindrical_surface,conical_surface,
        spherical_surface,toroidal_surface))
    SUBTYPE OF (surface);
      position : axis2_placement_3d;
  END_ENTITY; -- elementary_surface

  ENTITY elementary_wire_shape_representation
    SUBTYPE OF (shape_representation);
    WHERE
      wr1: SIZEOF(QUERY ( item <* SELF.items | (NOT ((SIZEOF([
                'BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
                'BUILDING_DESIGN_SCHEMA.CIRCLE',
                'BUILDING_DESIGN_SCHEMA.ELLIPSE',
                'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE',
                'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
                'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D']*TYPEOF(item))
               = 1) OR (item.name = 'reference curve'))) )) = 0;
      wr2: (SIZEOF(QUERY ( item <* SELF.items | (SIZEOF([
                'BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
```

```
              'BUILDING_DESIGN_SCHEMA.CIRCLE',
              'BUILDING_DESIGN_SCHEMA.ELLIPSE',
              'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE',
              'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM']*TYPEOF(item)) = 1) ))
              = 1) OR ((2 <= SIZEOF(QUERY ( it <* SELF.items | (it.name =
              'reference curve') ))) AND (SIZEOF(QUERY ( it <* SELF.items
               | (it.name = 'reference curve') )) <= 3));
    wr3: SIZEOF(QUERY ( item <* SELF.items | ((SIZEOF([
              'BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
              'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE']*TYPEOF(item)) = 1)
              AND (NOT valid_elementary_wire_composition(item))) )) = 0;
    wr4: SIZEOF(QUERY ( mi <* QUERY ( item <* SELF.items | (
              'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF(item)) ) | (
              NOT (('BUILDING_DESIGN_SCHEMA.' +
              'ELEMENTARY_WIRE_SHAPE_REPRESENTATION') IN TYPEOF(mi\
              mapped_item.mapping_source.mapped_representation))) )) = 0;
    wr5: SIZEOF(QUERY ( it <* SELF.items |
              ((it.name = 'reference curve')
              AND (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF(it))
              OR (('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(it))
              AND ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN TYPEOF(it\
              trimmed_curve.basis_curve)))))) )) = 0;
END_ENTITY; -- elementary_wire_shape_representation

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

ENTITY extruded_area_solid
  SUBTYPE OF (swept_area_solid);
    extruded_direction : direction;
    depth              : positive_length_measure;
  WHERE
    wr1: dot_product(SELF\swept_area_solid.swept_area.basis_surface\
              elementary_surface.position.p[3],extruded_direction) <> 0;
END_ENTITY; -- extruded_area_solid

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 | (
              'BUILDING_DESIGN_SCHEMA.FACE_OUTER_BOUND'
              IN TYPEOF(temp)) ))
              <= 1;
END_ENTITY; -- face

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

ENTITY face_outer_bound
  SUBTYPE OF (face_bound);
END_ENTITY; -- face_outer_bound

ENTITY face_surface
  SUBTYPE OF (face, geometric_representation_item);
    face_geometry : surface;
```

```
    same_sense    : BOOLEAN;
END_ENTITY; -- face_surface

ENTITY faceted_brep
  SUBTYPE OF (manifold_solid_brep);
END_ENTITY; -- faceted_brep

ENTITY faceted_csg_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    wr1: SIZEOF(QUERY ( item <* SELF.items | (NOT ((SIZEOF([
              'BUILDING_DESIGN_SCHEMA.CSG_SOLID',
              'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D',
              'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF(item)) = 1)
              OR (item.name = 'reference curve'))) )) = 0;
    wr2: (SIZEOF(QUERY ( item <* SELF.items | (NOT (SIZEOF([
              'BUILDING_DESIGN_SCHEMA.CSG_SOLID',
              'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM']*TYPEOF(item))=1)) ))
              >= 1) OR ((2 <= SIZEOF(QUERY ( it <* SELF.items | (it.name =
               'reference curve') ))) AND (SIZEOF(QUERY ( it <* SELF.items
               | (it.name = 'reference curve') )) <= 3));
    wr3: SIZEOF(QUERY ( item <* SELF.items | ((
              'BUILDING_DESIGN_SCHEMA.CSG_SOLID' IN TYPEOF(item)) AND (
              NOT valid_faceted_csg_tree(item\csg_solid.
              tree_root_expression))) )) = 0;
    wr4: SIZEOF(QUERY ( mi <* QUERY ( item <* SELF.items | (
              'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF(item)) ) | (
              NOT (
              'BUILDING_DESIGN_SCHEMA.FACETED_CSG_SHAPE_REPRESENTATION' IN
         TYPEOF(mi\mapped_item.mapping_source.mapped_representation)))))
              = 0;
    wr5: SIZEOF(QUERY( it <* SELF.items | ((it.name = 'reference curve')
              AND (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF(it))
              OR (('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(it))
              AND ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN TYPEOF(it\
              trimmed_curve.basis_curve)))))) )) = 0;
END_ENTITY; -- faceted_csg_shape_representation

ENTITY faceted_face_with_thickness_shape_representation
  SUBTYPE OF (shape_representation, solid_model);
  WHERE
    wr1: SIZEOF(SELF.items) = 2;
    wr2: SIZEOF(QUERY ( item <* SELF.items | ((SIZEOF([
              'BUILDING_DESIGN_SCHEMA.MEASURE_REPRESENTATION_ITEM',
              'BUILDING_DESIGN_SCHEMA.LENGTH_MEASURE_WITH_UNIT'] * TYPEOF(
              item)) = 2) AND (item.name = 'thickness')) )) = 1;
    wr3: SIZEOF(QUERY ( item <* SELF.items | (
              'BUILDING_DESIGN_SCHEMA.FACE' IN TYPEOF(item)) )) = 1;
    wr4: SIZEOF(QUERY ( item <* SELF.items | ((
              'BUILDING_DESIGN_SCHEMA.FACE' IN TYPEOF(item)) AND (NOT (
              SIZEOF(QUERY ( bnds <* item\face.bounds | (NOT (
              'BUILDING_DESIGN_SCHEMA.POLY_LOOP'
              IN TYPEOF(bnds.bound))) ))
              = 0))) )) = 0;
    wr5: SIZEOF(QUERY ( item <* SELF.items | ((NOT (
              'BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF(item))) OR (
              'BUILDING_DESIGN_SCHEMA.PLANE' IN TYPEOF(item\face_surface.
              face_geometry))) )) = 0;
END_ENTITY; -- faceted_face_with_thickness_shape_representation

ENTITY faceted_geometric_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    wr1: SIZEOF(QUERY ( item <* SELF.items | (NOT ((SIZEOF([
```

```
            'BUILDING_DESIGN_SCHEMA.FACETED_BREP',
            'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D',
            'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'] * TYPEOF(item)) = 1)
            OR (item.name = 'reference curve'))) )) = 0;
    wr2: (SIZEOF(QUERY ( item <* SELF.items | (SIZEOF([
            'BUILDING_DESIGN_SCHEMA.FACETED_BREP',
            'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM']*TYPEOF(item)) = 1) ))
            >= 1) OR ((2 <= SIZEOF(QUERY ( it <* SELF.items | (it.name =
             'reference curve') ))) AND (SIZEOF(QUERY ( it <* SELF.items
              | (it.name = 'reference curve') )) <= 3));
    wr3: SIZEOF(QUERY ( msb <* QUERY ( item <* SELF.items | (
            'BUILDING_DESIGN_SCHEMA.FACETED_BREP' IN TYPEOF(item)) ) | (
            NOT (SIZEOF(QUERY ( csh <* msb_shells(msb) | (NOT (SIZEOF(
            QUERY ( fcs <* csh\connected_face_set.cfs_faces | (NOT (
            SIZEOF(QUERY ( bnds <* fcs.bounds | (NOT (
            'BUILDING_DESIGN_SCHEMA.POLY_LOOP'
            IN TYPEOF(bnds.bound))) ))
            = 0)) )) = 0)) )) = 0)) )) = 0;
    wr4: SIZEOF(QUERY ( msb <* QUERY ( item <* SELF.items | (
            'BUILDING_DESIGN_SCHEMA.FACETED_BREP' IN TYPEOF(item)) ) | (
            NOT (SIZEOF(QUERY ( csh <* msb_shells(msb) | (NOT (SIZEOF(
            QUERY ( fcs <* csh.cfs_faces | ((
            'BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF(fcs)) AND (
            NOT ('BUILDING_DESIGN_SCHEMA.PLANE' IN TYPEOF(fcs\
            face_surface.face_geometry)))) )) = 0)) )) = 0)) )) = 0;
    wr5: SIZEOF(QUERY ( mi <* QUERY ( item <* SELF.items | (
            'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF(item)) ) | (
            NOT
        ('BUILDING_DESIGN_SCHEMA.FACETED_GEOMETRIC_SHAPE_REPRESENTATION'
            IN TYPEOF(mi\mapped_item.mapping_source.
            mapped_representation))) )) = 0;
    wr6: SIZEOF(QUERY( it <* SELF.items | ((it.name = 'reference curve')
            AND (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF(it))
            OR (('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(it))
            AND ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN TYPEOF(it\
            trimmed_curve.basis_curve)))))) )) = 0;
END_ENTITY; -- faceted_geometric_shape_representation

ENTITY faceted_space_boundary_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    wr1: SIZEOF(QUERY ( item <* SELF.items | (NOT ((SIZEOF([
            'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL',
            'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
            'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D']*TYPEOF(item))
            = 1) OR (item.name = 'reference curve'))) )) = 0;
    wr2: (SIZEOF(QUERY ( item <* SELF.items | (SIZEOF([
            'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL',
            'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM']*TYPEOF(item)) = 1) ))
            = 1) OR ((2 <= SIZEOF(QUERY ( it <* SELF.items | (it.name =
            'reference curve') ))) AND (SIZEOF(QUERY ( it <* SELF.items
              | (it.name = 'reference curve') )) <= 3));
    wr3: SIZEOF(QUERY ( csh <* QUERY ( item <* SELF.items | (
            'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF(item)) ) | (
            NOT (SIZEOF(QUERY ( fcs <* csh\connected_face_set.cfs_faces
              | (NOT (SIZEOF(QUERY ( bnds <* fcs.bounds | (NOT (
            'BUILDING_DESIGN_SCHEMA.POLY_LOOP'
            IN TYPEOF(bnds.bound))) ))
            = 0)) )) = 0)) )) = 0;
    wr4: SIZEOF(QUERY ( mi <* QUERY ( item <* SELF.items | (
            'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF(item)) ) | (
            NOT (('BUILDING_DESIGN_SCHEMA.' +
            'FACETED_SPACE_BOUNDARY_SHAPE_REPRESENTATION') IN TYPEOF(mi\
            mapped_item.mapping_source.mapped_representation))) )) = 0;
```

```
    wr5: SIZEOF(QUERY( it <* SELF.items | ((it.name = 'reference curve')
             AND (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF(it))
             OR (('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(it))
             AND ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN TYPEOF(it\
             trimmed_curve.basis_curve)))))) )) = 0;
END_ENTITY; -- faceted_space_boundary_shape_representation

ENTITY faceted_wire_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    wr1: SIZEOF(QUERY ( item <* SELF.items | (NOT ((SIZEOF([
             'BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
             'BUILDING_DESIGN_SCHEMA.POLYLINE',
             'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE',
             'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM',
             'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D']*TYPEOF(item))
             = 1) OR (item.name = 'reference curve'))) )) = 0;
    wr2: (SIZEOF(QUERY ( item <* SELF.items | (SIZEOF([
             'BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
             'BUILDING_DESIGN_SCHEMA.POLYLINE',
             'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE',
             'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM']*TYPEOF(item)) = 1) ))
             = 1) OR ((2 <= SIZEOF(QUERY ( it <* SELF.items | (it.name =
             'reference curve') ))) AND (SIZEOF(QUERY ( it <* SELF.items
              | (it.name = 'reference curve') )) <= 3));
    wr3: SIZEOF(QUERY ( item <* SELF.items | ((SIZEOF([
             'BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE',
             'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE']*TYPEOF(item)) = 1)
             AND (NOT valid_faceted_wire_composition(item))) )) = 0;
    wr4: SIZEOF(QUERY ( mi <* QUERY ( item <* SELF.items | (
             'BUILDING_DESIGN_SCHEMA.MAPPED_ITEM' IN TYPEOF(item)) ) | (
             NOT (('BUILDING_DESIGN_SCHEMA.' +
             'FACETED_WIRE_SHAPE_REPRESENTATION') IN TYPEOF(mi\
             mapped_item.mapping_source.mapped_representation))) )) = 0;
    wr5: SIZEOF(QUERY( it <* SELF.items | ((it.name = 'reference curve')
             AND (NOT (('BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF(it))
             OR (('BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(it))
             AND ('BUILDING_DESIGN_SCHEMA.CIRCLE' IN TYPEOF(it\
             trimmed_curve.basis_curve)))))) )) = 0;
END_ENTITY; -- faceted_wire_shape_representation

ENTITY fixture_equipment_element
  SUBTYPE OF (product_definition);
  WHERE
    wr1: SIZEOF(QUERY ( pdr <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
         'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION')
              | ('BUILDING_DESIGN_SCHEMA.BUILDING_SECTION' IN TYPEOF(pdr.
             relating_product_definition)) )) = 1;
    wr2: SIZEOF(USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
             'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
    wr3: SIZEOF(QUERY ( pds <* QUERY ( pd <* USEDIN(SELF,
             'BUILDING_DESIGN_SCHEMA.PROPERTY_DEFINITION.DEFINITION') | (
             'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN TYPEOF(
             pd)) ) | (SIZEOF(QUERY ( sa <* USEDIN(pds,
             'BUILDING_DESIGN_SCHEMA.' + 'SHAPE_ASPECT.OF_SHAPE') | ((
             'BUILDING_DESIGN_SCHEMA.POSITIVE_COMPONENT' IN TYPEOF(sa))
             AND (sa.description = 'main component')) )) = 1) )) = 1;
END_ENTITY; -- fixture_equipment_element

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

```
ENTITY geometric_curve_set
  SUBTYPE OF (geometric_set);
  WHERE
    wr1: SIZEOF(QUERY ( temp <* SELF\geometric_set.elements | (
              'BUILDING_DESIGN_SCHEMA.SURFACE' IN TYPEOF(temp)) )) = 0;
END_ENTITY; -- geometric_curve_set

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

ENTITY geometric_representation_item
  SUPERTYPE OF (ONEOF (point,direction,vector,placement,
      cartesian_transformation_operator,curve,surface,edge_curve,
      face_surface,poly_loop,vertex_point,solid_model,boolean_result,
      sphere,right_circular_cone,right_circular_cylinder,torus,block,
      half_space_solid,geometric_set))
  SUBTYPE OF (representation_item);
  DERIVE
    dim : dimension_count := dimension_of(SELF);
  WHERE
    wr1: SIZEOF(QUERY ( using_rep <* using_representations(SELF) | (NOT
              ('BUILDING_DESIGN_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT'
              IN TYPEOF(using_rep.context_of_items))) )) = 0;
END_ENTITY; -- geometric_representation_item

ENTITY geometric_set
  SUPERTYPE OF (geometric_curve_set)
  SUBTYPE OF (geometric_representation_item);
    elements : SET [1:?] OF geometric_set_select;
END_ENTITY; -- geometric_set

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

ENTITY ground_face_space_boundary_shape_representation
  SUBTYPE OF (shape_representation, solid_model);
  WHERE
    wr1: SIZEOF(SELF.items) = 1;
    wr2: SIZEOF(QUERY ( item <* SELF.items | (
              'BUILDING_DESIGN_SCHEMA.FACE' IN TYPEOF(item)) )) = 1;
    wr3: SIZEOF(QUERY ( item <* SELF.items | (NOT (SIZEOF(
              QUERY ( bnds <* item\face.bounds | (NOT (
              'BUILDING_DESIGN_SCHEMA.POLY_LOOP'
              IN TYPEOF(bnds.bound))) ))
              = 0)) )) = 0;
    wr4: SIZEOF(QUERY ( item <* SELF.items | ((NOT (
              'BUILDING_DESIGN_SCHEMA.FACE_SURFACE' IN TYPEOF(item))) OR (
              'BUILDING_DESIGN_SCHEMA.PLANE' IN TYPEOF(item\face_surface.
              face_geometry))) )) = 0;
END_ENTITY; -- ground_face_space_boundary_shape_representation

ENTITY group;
    name        : label;
    description : text;
END_ENTITY; -- group

ENTITY group_assignment
  ABSTRACT SUPERTYPE;
    assigned_group : group;
END_ENTITY; -- group_assignment
```

```
ENTITY half_space_solid
  SUBTYPE OF (geometric_representation_item);
    base_surface   : surface;
    agreement_flag : BOOLEAN;
END_ENTITY; -- half_space_solid

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

ENTITY intersection_curve
  SUBTYPE OF (surface_curve);
  WHERE
    wr1: SIZEOF(SELF\surface_curve.associated_geometry) = 2;
    wr2: associated_surface(SELF\surface_curve.associated_geometry[1])
              <> associated_surface(SELF\surface_curve.associated_geometry
              [2]);
END_ENTITY; -- intersection_curve

ENTITY length_measure_with_unit
  SUBTYPE OF (measure_with_unit);
  WHERE
    wr1: 'BUILDING_DESIGN_SCHEMA.LENGTH_UNIT' IN TYPEOF(SELF\
              measure_with_unit.unit_component);
END_ENTITY; -- length_measure_with_unit

ENTITY length_unit
  SUBTYPE OF (named_unit);
  WHERE
    wr1: (SELF\named_unit.dimensions.length_exponent = 1) AND (SELF\
              named_unit.dimensions.mass_exponent = 0) AND (SELF\
              named_unit.dimensions.time_exponent = 0) AND (SELF\
              named_unit.dimensions.electric_current_exponent = 0) AND (
              SELF\named_unit.dimensions.
              thermodynamic_temperature_exponent = 0) AND (SELF\named_unit
              .dimensions.amount_of_substance_exponent = 0) AND (SELF\
              named_unit.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- length_unit

ENTITY line
  SUBTYPE OF (curve);
    pnt : cartesian_point;
    dir : vector;
  WHERE
    wr1: dir.dim = pnt.dim;
END_ENTITY; -- line

ENTITY loop
  SUPERTYPE OF (ONEOF (vertex_loop,edge_loop,poly_loop))
  SUBTYPE OF (topological_representation_item);
END_ENTITY; -- loop

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

ENTITY mapped_item
  SUBTYPE OF (representation_item);
    mapping_source : representation_map;
    mapping_target : representation_item;
  WHERE
```

```
    wr1: acyclic_mapped_representation(using_representations(SELF),
         [SELF]);
END_ENTITY; -- mapped_item

ENTITY measure_representation_item
  SUBTYPE OF (representation_item, measure_with_unit);
END_ENTITY; -- measure_representation_item

ENTITY measure_with_unit
  SUPERTYPE OF (ONEOF (length_measure_with_unit,
      plane_angle_measure_with_unit));
    value_component : measure_value;
    unit_component  : unit;
  WHERE
    wr1: valid_units(SELF);
END_ENTITY; -- measure_with_unit

ENTITY name_assignment
  ABSTRACT SUPERTYPE;
    assigned_name : label;
END_ENTITY; -- name_assignment

ENTITY named_unit
  SUPERTYPE OF (ONEOF (si_unit,conversion_based_unit) ANDOR ONEOF (
      length_unit,plane_angle_unit));
    dimensions : dimensional_exponents;
END_ENTITY; -- named_unit

ENTITY negative_component
  SUBTYPE OF (shape_aspect);
  WHERE
    wr1: (SIZEOF(USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
              'PROPERTY_DEFINITION.DEFINITION')) >= 1) AND (SIZEOF(
              QUERY ( pd <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
              'PROPERTY_DEFINITION.DEFINITION') | (NOT (SIZEOF(USEDIN(pd,
              'BUILDING_DESIGN_SCHEMA.' +
              'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) >= 1)) ))
              = 0);
    wr2: SIZEOF(TYPEOF(SELF.of_shape.definition) * [
              'BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
              'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
              'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
              'BUILDING_DESIGN_SCHEMA.STRUCTURE_ENCLOSURE_ELEMENT']) = 1;
    wr3: SIZEOF(USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
              'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
    wr4: SIZEOF(QUERY ( pd <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
              'PROPERTY_DEFINITION.DEFINITION') | (NOT (SIZEOF(
              QUERY ( pdr <* USEDIN(pd,'BUILDING_DESIGN_SCHEMA.' +
              'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') | (NOT (
              SIZEOF(TYPEOF(pdr.used_representation) * [
              'BUILDING_DESIGN_SCHEMA.' +
              'ADVANCED_BREP_BUILDING_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.ADVANCED_CSG_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'ADVANCED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'ADVANCED_WIRE_SHAPE_REPRESENTATION',
            'BUILDING_DESIGN_SCHEMA.ELEMENTARY_CSG_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'ELEMENTARY_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'ELEMENTARY_GEOMETRIC_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'ELEMENTARY_WIRE_SHAPE_REPRESENTATION',
```

```
              'BUILDING_DESIGN_SCHEMA.FACETED_CSG_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'FACETED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
          'BUILDING_DESIGN_SCHEMA.FACETED_GEOMETRIC_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'FACETED_WIRE_SHAPE_REPRESENTATION']) = 1)) )) = 0)) )) = 0;
  END_ENTITY; -- negative_component

  ENTITY offset_curve_3d
    SUBTYPE OF (curve);
      basis_curve    : curve;
      distance       : length_measure;
      self_intersect : LOGICAL;
      ref_direction  : direction;
    WHERE
      wr1: (basis_curve.dim = 3) AND (ref_direction.dim = 3);
  END_ENTITY; -- offset_curve_3d

  ENTITY open_shell
    SUBTYPE OF (connected_face_set);
  END_ENTITY; -- open_shell

  ENTITY opening
    SUBTYPE OF (negative_component);
    WHERE
      wr1: SIZEOF(USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
                'BUILDING_COMPONENT_CLASSIFICATION_ASSIGNMENT.ITEMS')) = 1;
      wr2: SIZEOF(QUERY ( bcca <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
                'BUILDING_COMPONENT_CLASSIFICATION_ASSIGNMENT.ITEMS') | (
                NOT (bcca\group_assignment.assigned_group.description =
                'opening')) )) = 0;
  END_ENTITY; -- opening

  ENTITY ordinal_date
    SUBTYPE OF (date);
      day_component : day_in_year_number;
    WHERE
      wr1: ((NOT leap_year(SELF.year_component)) AND (1 <= day_component)
                AND (day_component <= 365)) OR (leap_year(SELF.
                year_component) AND (1 <= day_component) AND (day_component
                <= 366));
  END_ENTITY; -- ordinal_date

  ENTITY organization;
      id          : OPTIONAL identifier;
      name        : label;
      description : text;
  END_ENTITY; -- organization

  ENTITY organization_assignment
    ABSTRACT SUPERTYPE;
      assigned_organization : organization;
      role                  : organization_role;
  END_ENTITY; -- organization_assignment

  ENTITY organization_role;
      name : label;
  END_ENTITY; -- organization_role

  ENTITY organizational_project;
      name                      : label;
      description               : text;
      responsible_organizations : SET [1:?] OF organization;
  END_ENTITY; -- organizational_project
```

```
ENTITY oriented_closed_shell
  SUBTYPE OF (closed_shell);
    closed_shell_element : closed_shell;
    orientation          : BOOLEAN;
  DERIVE
    SELF\connected_face_set.cfs_faces : SET [1:?] OF face :=
                                          conditional_reverse(SELF.
                                          orientation,SELF.
                            closed_shell_element.cfs_faces);
  WHERE
    wr1: NOT ('BUILDING_DESIGN_SCHEMA.ORIENTED_CLOSED_SHELL' IN TYPEOF(
              SELF.closed_shell_element));
END_ENTITY; -- oriented_closed_shell

ENTITY oriented_edge
  SUBTYPE OF (edge);
    edge_element : edge;
    orientation  : BOOLEAN;
  DERIVE
    SELF\edge.edge_start : vertex := boolean_choose(SELF.orientation,
                              SELF.edge_element.edge_start,SELF.
                              edge_element.edge_end);
    SELF\edge.edge_end   : vertex := boolean_choose(SELF.orientation,
                              SELF.edge_element.edge_end,SELF.
                              edge_element.edge_start);
  WHERE
    wr1: NOT ('BUILDING_DESIGN_SCHEMA.ORIENTED_EDGE' IN TYPEOF(SELF.
              edge_element));
END_ENTITY; -- oriented_edge

ENTITY oriented_face
  SUBTYPE OF (face);
    face_element : face;
    orientation  : BOOLEAN;
  DERIVE
    SELF\face.bounds : SET [1:?] OF face_bound := conditional_reverse(
                           SELF.orientation,SELF.face_element.bounds);
  WHERE
    wr1: NOT ('BUILDING_DESIGN_SCHEMA.ORIENTED_FACE' IN TYPEOF(SELF.
              face_element));
END_ENTITY; -- oriented_face

ENTITY oriented_open_shell
  SUBTYPE OF (open_shell);
    open_shell_element : open_shell;
    orientation        : BOOLEAN;
  DERIVE
    SELF\connected_face_set.cfs_faces : SET [1:?] OF face :=
                                          conditional_reverse(SELF.
                                          orientation,SELF.
                                          open_shell_element.cfs_faces);
  WHERE
    wr1: NOT ('BUILDING_DESIGN_SCHEMA.ORIENTED_OPEN_SHELL' IN TYPEOF(
              SELF.open_shell_element));
END_ENTITY; -- oriented_open_shell

ENTITY oriented_path
  SUBTYPE OF (path);
    path_element : path;
    orientation  : BOOLEAN;
  DERIVE
    SELF\path.edge_list : LIST [1:?] OF UNIQUE oriented_edge :=
                            conditional_reverse(SELF.orientation,SELF.
                            path_element.edge_list);
```

```
    WHERE
      wr1: NOT ('BUILDING_DESIGN_SCHEMA.ORIENTED_PATH' IN TYPEOF(SELF.
              path_element));
  END_ENTITY; -- oriented_path

  ENTITY outer_boundary_curve
    SUBTYPE OF (boundary_curve);
  END_ENTITY; -- outer_boundary_curve

  ENTITY parabola
    SUBTYPE OF (conic);
      focal_dist : length_measure;
    WHERE
      wr1: focal_dist <> 0;
  END_ENTITY; -- parabola

  ENTITY parametric_representation_context
    SUBTYPE OF (representation_context);
  END_ENTITY; -- parametric_representation_context

  ENTITY path
    SUPERTYPE OF (ONEOF (edge_loop,oriented_path))
    SUBTYPE OF (topological_representation_item);
      edge_list : LIST [1:?] OF UNIQUE oriented_edge;
    WHERE
      wr1: path_head_to_tail(SELF);
  END_ENTITY; -- path

  ENTITY pcurve
    SUBTYPE OF (curve);
      basis_surface      : surface;
      reference_to_curve : definitional_representation;
    WHERE
      wr1: SIZEOF(reference_to_curve\representation.items) = 1;
      wr2: 'BUILDING_DESIGN_SCHEMA.CURVE' IN TYPEOF(reference_to_curve\
              representation.items[1]);
      wr3: reference_to_curve\representation.items[1]\
              geometric_representation_item.dim = 2;
  END_ENTITY; -- pcurve

  ENTITY person;
      id            : identifier;
      last_name     : OPTIONAL label;
      first_name    : OPTIONAL label;
      middle_names  : OPTIONAL LIST [1:?] OF label;
      prefix_titles : OPTIONAL LIST [1:?] OF label;
      suffix_titles : OPTIONAL LIST [1:?] OF label;
    UNIQUE
      ur1 : id;
    WHERE
      wr1: EXISTS(last_name) OR EXISTS(first_name);
  END_ENTITY; -- person

  ENTITY person_and_organization;
      the_person       : person;
      the_organization : organization;
  END_ENTITY; -- person_and_organization

  ENTITY person_and_organization_assignment
    ABSTRACT SUPERTYPE;
      assigned_person_and_organization : person_and_organization;
      role                             : person_and_organization_role;
  END_ENTITY; -- person_and_organization_assignment
```

```
ENTITY person_and_organization_role;
    name : label;
END_ENTITY; -- person_and_organization_role

ENTITY person_assignment
  ABSTRACT SUPERTYPE;
    assigned_person : person;
    role            : person_role;
END_ENTITY; -- person_assignment

ENTITY person_role;
    name : label;
END_ENTITY; -- person_role

ENTITY placement
  SUPERTYPE OF (ONEOF (axis1_placement,axis2_placement_2d,
      axis2_placement_3d))
  SUBTYPE OF (geometric_representation_item);
    location : cartesian_point;
END_ENTITY; -- placement

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

ENTITY plane_angle_measure_with_unit
  SUBTYPE OF (measure_with_unit);
  WHERE
    wr1: 'BUILDING_DESIGN_SCHEMA.PLANE_ANGLE_UNIT' IN TYPEOF(SELF\
           measure_with_unit.unit_component);
END_ENTITY; -- plane_angle_measure_with_unit

ENTITY plane_angle_unit
  SUBTYPE OF (named_unit);
  WHERE
    wr1: (SELF\named_unit.dimensions.length_exponent = 0) AND (SELF\
           named_unit.dimensions.mass_exponent = 0) AND (SELF\
           named_unit.dimensions.time_exponent = 0) AND (SELF\
           named_unit.dimensions.electric_current_exponent = 0) AND (
           SELF\named_unit.dimensions.
           thermodynamic_temperature_exponent = 0) AND (SELF\named_unit
           .dimensions.amount_of_substance_exponent = 0) AND (SELF\
           named_unit.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- plane_angle_unit

ENTITY point
  SUPERTYPE OF (cartesian_point)
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- point

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

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

ENTITY positive_component
  SUBTYPE OF (shape_aspect);
  WHERE
    wr1: (SIZEOF(USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
```

```
            'PROPERTY_DEFINITION.DEFINITION')) >= 1) AND (SIZEOF(
            QUERY ( pd <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
            'PROPERTY_DEFINITION.DEFINITION') | (NOT (SIZEOF(USEDIN(pd,
            'BUILDING_DESIGN_SCHEMA.' +
            'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) >= 1)) ))
            = 0);
    wr2: SIZEOF(TYPEOF(SELF.of_shape.definition) * [
            'BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
            'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
            'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
            'BUILDING_DESIGN_SCHEMA.STRUCTURE_ENCLOSURE_ELEMENT']) = 1;
    wr3: SIZEOF(USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
            'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
    wr4: SIZEOF(QUERY ( pd <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
            'PROPERTY_DEFINITION.DEFINITION') | (NOT (SIZEOF(
            QUERY ( pdr <* USEDIN(pd,'BUILDING_DESIGN_SCHEMA.' +
            'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') | (NOT (
            SIZEOF(TYPEOF(pdr.used_representation) * [
            'BUILDING_DESIGN_SCHEMA.' +
            'ADVANCED_BREP_BUILDING_SHAPE_REPRESENTATION',
            'BUILDING_DESIGN_SCHEMA.ADVANCED_CSG_SHAPE_REPRESENTATION',
            'BUILDING_DESIGN_SCHEMA.' +
            'ADVANCED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
            'BUILDING_DESIGN_SCHEMA.' +
            'ADVANCED_WIRE_SHAPE_REPRESENTATION',
           'BUILDING_DESIGN_SCHEMA.ELEMENTARY_CSG_SHAPE_REPRESENTATION',
            'BUILDING_DESIGN_SCHEMA.' +
            'ELEMENTARY_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
      'BUILDING_DESIGN_SCHEMA.ELEMENTARY_GEOMETRIC_SHAPE_REPRESENTATION',
            'BUILDING_DESIGN_SCHEMA.' +
            'ELEMENTARY_WIRE_SHAPE_REPRESENTATION',
            'BUILDING_DESIGN_SCHEMA.FACETED_CSG_SHAPE_REPRESENTATION',
            'BUILDING_DESIGN_SCHEMA.' +
            'FACETED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
        'BUILDING_DESIGN_SCHEMA.FACETED_GEOMETRIC_SHAPE_REPRESENTATION',
            'BUILDING_DESIGN_SCHEMA.' +
            'FACETED_WIRE_SHAPE_REPRESENTATION']) = 1)) )) = 0)) )) = 0;
END_ENTITY; -- positive_component

ENTITY product;
    id                : identifier;
    name              : label;
    description       : text;
    frame_of_reference : SET [1:?] OF product_context;
  UNIQUE
    ur1 : id;
END_ENTITY; -- product

ENTITY product_category;
    name        : label;
    description : OPTIONAL text;
END_ENTITY; -- product_category

ENTITY product_category_relationship;
    name         : label;
    description  : text;
    category     : product_category;
    sub_category : product_category;
  WHERE
    wr1: acyclic_product_category_relationship(SELF,[SELF.sub_category]);
END_ENTITY; -- product_category_relationship

ENTITY product_context
  SUBTYPE OF (application_context_element);
```

```
    discipline_type : label;
END_ENTITY; -- product_context

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

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

ENTITY product_definition_formation;
    id          : identifier;
    description : text;
    of_product  : product;
  UNIQUE
    ur1 : id, of_product;
END_ENTITY; -- product_definition_formation

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

ENTITY product_definition_shape
  SUBTYPE OF (property_definition);
  UNIQUE
    ur1 : definition;
  WHERE
    wr1: 'BUILDING_DESIGN_SCHEMA.CHARACTERIZED_PRODUCT_DEFINITION' IN
            TYPEOF(SELF\property_definition.definition);
END_ENTITY; -- product_definition_shape

ENTITY product_definition_usage
  SUPERTYPE OF (assembly_component_usage)
  SUBTYPE OF (product_definition_relationship);
  UNIQUE
    ur1 : id, relating_product_definition, related_product_definition;
  WHERE
    wr1: acyclic_product_definition_relationship(SELF,[SELF\
              product_definition_relationship.related_product_definition],
              'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_USAGE.' +
              'RELATED_PRODUCT_DEFINITION');
END_ENTITY; -- product_definition_usage

ENTITY product_related_product_category
  SUBTYPE OF (product_category);
    products : SET [1:?] OF product;
END_ENTITY; -- product_related_product_category

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

ENTITY property_definition_representation;
```

```
    definition          : property_definition;
    used_representation : representation;
END_ENTITY; -- property_definition_representation

ENTITY quasi_uniform_curve
  SUBTYPE OF (b_spline_curve);
END_ENTITY; -- quasi_uniform_curve

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

ENTITY rational_b_spline_curve
  SUBTYPE OF (b_spline_curve);
    weights_data : LIST [2:?] OF REAL;
  DERIVE
    weights : ARRAY [0:upper_index_on_control_points] OF REAL :=
                list_to_array(weights_data,0,
                upper_index_on_control_points);
  WHERE
    wr1: SIZEOF(weights_data) = SIZEOF(SELF\b_spline_curve.
           control_points_list);
    wr2: curve_weights_positive(SELF);
END_ENTITY; -- rational_b_spline_curve

ENTITY rational_b_spline_surface
  SUBTYPE OF (b_spline_surface);
    weights_data : LIST [2:?] OF LIST [2:?] OF REAL;
  DERIVE
    weights : ARRAY [0:u_upper] OF ARRAY [0:v_upper] OF REAL :=
                make_array_of_array(weights_data,0,u_upper,0,v_upper);
  WHERE
    wr1: (SIZEOF(weights_data) = SIZEOF(SELF\b_spline_surface.
           control_points_list)) AND (SIZEOF(weights_data[1]) = SIZEOF(
           SELF\b_spline_surface.control_points_list[1]));
    wr2: surface_weights_positive(SELF);
END_ENTITY; -- rational_b_spline_surface

ENTITY recess
  SUBTYPE OF (negative_component);
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
           'BUILDING_COMPONENT_CLASSIFICATION_ASSIGNMENT.ITEMS')) = 1;
    wr2: SIZEOF(QUERY ( bcca <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
           'BUILDING_COMPONENT_CLASSIFICATION_ASSIGNMENT.ITEMS') | (
           NOT (bcca\group_assignment.assigned_group.description =
           'recess')) )) = 0;
END_ENTITY; -- recess

ENTITY rectangular_composite_surface
  SUBTYPE OF (bounded_surface);
    segments : LIST [1:?] OF LIST [1:?] OF surface_patch;
  DERIVE
    n_u : INTEGER := SIZEOF(segments);
    n_v : INTEGER := SIZEOF(segments[1]);
  WHERE
    wr1: [] = QUERY ( s <* segments | (n_v <> SIZEOF(s)) );
    wr2: constraints_rectangular_composite_surface(SELF);
END_ENTITY; -- rectangular_composite_surface

ENTITY rectangular_trimmed_surface
  SUBTYPE OF (bounded_surface);
    basis_surface : surface;
    u1            : parameter_value;
```

```
    u2            : parameter_value;
    v1            : parameter_value;
    v2            : parameter_value;
    usense        : BOOLEAN;
    vsense        : BOOLEAN;
  WHERE
    wr1: u1 <> u2;
    wr2: v1 <> v2;
    wr3: (('BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF(
             basis_surface)) AND (NOT ('BUILDING_DESIGN_SCHEMA.PLANE' IN
             TYPEOF(basis_surface)))) OR (
             'BUILDING_DESIGN_SCHEMA.SURFACE_OF_REVOLUTION' IN TYPEOF(
             basis_surface)) OR (usense = (u2 > u1));
    wr4: ('BUILDING_DESIGN_SCHEMA.SPHERICAL_SURFACE' IN TYPEOF(
             basis_surface)) OR (
             'BUILDING_DESIGN_SCHEMA.TOROIDAL_SURFACE' IN TYPEOF(
             basis_surface)) OR (vsense = (v2 > v1));
END_ENTITY; -- rectangular_trimmed_surface

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

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

ENTITY representation_item;
    name : label;
  WHERE
    wr1: SIZEOF(using_representations(SELF)) > 0;
END_ENTITY; -- representation_item

ENTITY representation_map;
    mapping_origin        : representation_item;
    mapped_representation : representation;
  INVERSE
    map_usage : SET [1:?] OF mapped_item FOR mapping_source;
  WHERE
    wr1: item_in_context(SELF.mapping_origin,SELF.mapped_representation.
             context_of_items);
END_ENTITY; -- representation_map

ENTITY representation_relationship;
    name        : label;
    description : text;
    rep_1       : representation;
    rep_2       : representation;
END_ENTITY; -- representation_relationship

ENTITY representation_relationship_with_transformation
  SUBTYPE OF (representation_relationship);
    transformation_operator : transformation;
  WHERE
    wr1: SELF\representation_relationship.rep_1.context_of_items :<>:
             SELF\representation_relationship.rep_2.context_of_items;
END_ENTITY; -- representation_relationship_with_transformation
```

```
ENTITY revolved_area_solid
  SUBTYPE OF (swept_area_solid);
    axis  : axis1_placement;
    angle : plane_angle_measure;
  DERIVE
    axis_line : line := line(axis.location,vector(axis.z,1));
END_ENTITY; -- revolved_area_solid

ENTITY right_circular_cone
  SUBTYPE OF (geometric_representation_item);
    position   : axis1_placement;
    height     : positive_length_measure;
    radius     : length_measure;
    semi_angle : plane_angle_measure;
  WHERE
    wr1: radius >= 0;
END_ENTITY; -- right_circular_cone

ENTITY right_circular_cylinder
  SUBTYPE OF (geometric_representation_item);
    position : axis1_placement;
    height   : positive_length_measure;
    radius   : positive_length_measure;
END_ENTITY; -- right_circular_cylinder

ENTITY service_element
  SUBTYPE OF (product_definition);
  WHERE
    wr1: SIZEOF(QUERY ( pdr <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
         'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION')
              | ('BUILDING_DESIGN_SCHEMA.BUILDING_SECTION' IN TYPEOF(pdr.
             relating_product_definition)) )) = 1;
    wr2: SIZEOF(USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
             'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
    wr3: SIZEOF(QUERY ( pds <* QUERY ( pd <* USEDIN(SELF,
             'BUILDING_DESIGN_SCHEMA.PROPERTY_DEFINITION.DEFINITION') | (
             'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN TYPEOF(
             pd)) ) | (SIZEOF(QUERY ( sa <* USEDIN(pds,
             'BUILDING_DESIGN_SCHEMA.' + 'SHAPE_ASPECT.OF_SHAPE') | ((
             'BUILDING_DESIGN_SCHEMA.POSITIVE_COMPONENT' IN TYPEOF(sa))
             AND (sa.description = 'main component')) )) = 1) )) = 1;
END_ENTITY; -- service_element

ENTITY shape_aspect;
    name                 : label;
    description          : text;
    of_shape             : product_definition_shape;
    product_definitional : LOGICAL;
END_ENTITY; -- shape_aspect

ENTITY shape_aspect_relationship;
    name                  : label;
    description           : text;
    relating_shape_aspect : shape_aspect;
    related_shape_aspect  : shape_aspect;
END_ENTITY; -- shape_aspect_relationship

ENTITY shape_definition_representation
  SUBTYPE OF (property_definition_representation);
  WHERE
    wr1: ('BUILDING_DESIGN_SCHEMA.SHAPE_DEFINITION' IN TYPEOF(SELF.
             definition.definition)) OR (
             'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN TYPEOF(
             SELF.definition));
```

```
    wr2: 'BUILDING_DESIGN_SCHEMA.SHAPE_REPRESENTATION' IN TYPEOF(SELF.
            used_representation);
END_ENTITY; -- shape_definition_representation

ENTITY shape_representation
  SUBTYPE OF (representation);
END_ENTITY; -- shape_representation

ENTITY si_unit
  SUBTYPE OF (named_unit);
    prefix : OPTIONAL si_prefix;
    name   : si_unit_name;
  DERIVE
    SELF\named_unit.dimensions : dimensional_exponents :=
                                    dimensions_for_si_unit(SELF.name);
END_ENTITY; -- si_unit

ENTITY site
  SUBTYPE OF (characterized_object, product_definition);
  WHERE
    wr1: (SIZEOF(USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
            'PROPERTY_DEFINITION.DEFINITION')) >= 1) AND (SIZEOF(
            QUERY ( pd <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
            'PROPERTY_DEFINITION.DEFINITION') | (SIZEOF(USEDIN(pd,
            'BUILDING_DESIGN_SCHEMA.' +
            'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) = 0) )) =
            0);
    wr2: SIZEOF(QUERY ( pd <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
            'PROPERTY_DEFINITION.DEFINITION') | (NOT (SIZEOF(
            QUERY ( pdr <* USEDIN(pd,'BUILDING_DESIGN_SCHEMA.' +
            'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') | (NOT (
            'BUILDING_DESIGN_SCHEMA.SITE_REPRESENTATION' IN TYPEOF(pdr.
            used_representation))) )) = 0)) )) = 0;
END_ENTITY; -- site

ENTITY site_representation
  SUBTYPE OF (shape_representation);
  WHERE
    wr1: SIZEOF(QUERY ( pdr <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
            'PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTATION') |
            (NOT ('BUILDING_DESIGN_SCHEMA.SITE' IN TYPEOF(pdr.definition
            .definition))) )) = 0;
    wr2: SIZEOF(QUERY ( item <* SELF.items | (NOT (SIZEOF([
            'BUILDING_DESIGN_SCHEMA.CONNECTED_FACE_SET',
            'BUILDING_DESIGN_SCHEMA.GEOMETRIC_CURVE_SET'] *
             TYPEOF(item))
            = 1)) )) = 1;
    wr3: SIZEOF(QUERY ( cfs <* QUERY ( item <* SELF.items | (
            'BUILDING_DESIGN_SCHEMA.CONNECTED_FACE_SET'
            IN TYPEOF(item)) )
             | (NOT (SIZEOF(QUERY ( fcs <* cfs\connected_face_set.
            cfs_faces | (NOT (SIZEOF(QUERY ( bnds <* fcs.bounds | (NOT (
            'BUILDING_DESIGN_SCHEMA.POLY_LOOP'
            IN TYPEOF(bnds.bound))) ))
            = 0)) )) = 0)) )) = 0;
    wr4: SIZEOF(QUERY ( cfs <* QUERY ( item <* SELF.items | (
            'BUILDING_DESIGN_SCHEMA.CONNECTED_FACE_SET'
            IN TYPEOF(item)) )
             | (NOT (SIZEOF(QUERY ( fcs <* cfs\connected_face_set.
            cfs_faces | (NOT (SIZEOF(QUERY ( bnds <* fcs.bounds | (NOT (
            SIZEOF(bnds.bound\poly_loop.polygon) = 3)) )) = 0)) ))
            = 0)) )) = 0;
    wr5: SIZEOF(QUERY ( gcs <* QUERY ( item <* SELF.items | (
            'BUILDING_DESIGN_SCHEMA.GEOMETRIC_CURVE_SET'
```

```
              IN TYPEOF(item)) )
                | (NOT (SIZEOF(QUERY ( el <* gcs\geometric_set.elements | (
               NOT (SIZEOF(['BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT',
               'BUILDING_DESIGN_SCHEMA.POLYLINE'] * TYPEOF(el)) = 1)) )) =
               0)) )) = 0;
      wr6: SIZEOF(QUERY ( gcs <* QUERY ( item <* SELF.items | (
               'BUILDING_DESIGN_SCHEMA.GEOMETRIC_CURVE_SET'
               IN TYPEOF(item)) )
                | (NOT (SIZEOF(QUERY ( el <* gcs\geometric_set.elements | (
               'BUILDING_DESIGN_SCHEMA.CARTESIAN_POINT' IN TYPEOF(el)) ))
               >= 1)) )) = 0;
      wr7: SIZEOF(QUERY ( gcs <* QUERY ( item <* SELF.items | (
               'BUILDING_DESIGN_SCHEMA.GEOMETRIC_CURVE_SET'
               IN TYPEOF(item)) )
                | (NOT (SIZEOF(QUERY ( pline <* QUERY ( el <* gcs\
               geometric_set.elements | ('BUILDING_DESIGN_SCHEMA.POLYLINE'
               IN TYPEOF(el)) ) | (NOT (SIZEOF(QUERY ( pline_pt <* pline\
               polyline.points | (NOT (pline_pt IN gcs\geometric_set.
               elements)) )) = 0)) )) = 0)) )) = 0;
  END_ENTITY; -- site_representation

  ENTITY solid_model
    SUPERTYPE OF (ONEOF (csg_solid,manifold_solid_brep,swept_area_solid))
    SUBTYPE OF (geometric_representation_item);
  END_ENTITY; -- solid_model

  ENTITY space_element
    SUBTYPE OF (product_definition);
    WHERE
      wr1: SIZEOF(QUERY ( pdr <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
           'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION')
                | ('BUILDING_DESIGN_SCHEMA.BUILDING_SECTION' IN TYPEOF(pdr.
               relating_product_definition)) )) = 1;
      wr2: SIZEOF(USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
               'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
      wr3: (SIZEOF(QUERY ( pd <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
               'PROPERTY_DEFINITION.DEFINITION') | (
               'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN TYPEOF(
               pd)) )) >= 1) AND (SIZEOF(QUERY ( pds <* QUERY ( pd <*
               USEDIN(SELF,
               'BUILDING_DESIGN_SCHEMA.PROPERTY_DEFINITION.DEFINITION') | (
               'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN TYPEOF(
               pd)) ) | (NOT (SIZEOF(USEDIN(pds,'BUILDING_DESIGN_SCHEMA.' +
                'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) >= 1)) ))
              = 0);
      wr4: SIZEOF(QUERY ( pd <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
               'PROPERTY_DEFINITION.DEFINITION') | (NOT (SIZEOF(
               QUERY ( pdr <* USEDIN(pd,'BUILDING_DESIGN_SCHEMA.' +
               'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION') | (NOT (
               SIZEOF(TYPEOF(pdr.used_representation) * [
               'BUILDING_DESIGN_SCHEMA.' +
               'ADVANCED_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
               'BUILDING_DESIGN_SCHEMA.' +
               'ELEMENTARY_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
               'BUILDING_DESIGN_SCHEMA.' +
               'FACETED_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
               'BUILDING_DESIGN_SCHEMA.' +
               'GROUND_FACE_SPACE_BOUNDARY_SHAPE_REPRESENTATION']) = 1)) ))
              = 0)) )) = 0;
  END_ENTITY; -- space_element

  ENTITY sphere
    SUBTYPE OF (geometric_representation_item);
      radius : positive_length_measure;
```

```
    centre : point;
END_ENTITY; -- sphere

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

ENTITY structure_enclosure_element
  SUBTYPE OF (product_definition);
  WHERE
    wr1: SIZEOF(QUERY ( pdr <* USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
         'PRODUCT_DEFINITION_RELATIONSHIP.RELATED_PRODUCT_DEFINITION')
              | ('BUILDING_DESIGN_SCHEMA.BUILDING_SECTION' IN TYPEOF(pdr.
             relating_product_definition)) )) = 1;
    wr2: SIZEOF(USEDIN(SELF,'BUILDING_DESIGN_SCHEMA.' +
             'BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT.ITEM')) = 1;
    wr3: SIZEOF(QUERY ( pds <* QUERY ( pd <* USEDIN(SELF,
             'BUILDING_DESIGN_SCHEMA.PROPERTY_DEFINITION.DEFINITION') | (
             'BUILDING_DESIGN_SCHEMA.PRODUCT_DEFINITION_SHAPE' IN TYPEOF(
             pd)) ) | (SIZEOF(QUERY ( sa <* USEDIN(pds,
             'BUILDING_DESIGN_SCHEMA.' + 'SHAPE_ASPECT.OF_SHAPE') | ((
             'BUILDING_DESIGN_SCHEMA.POSITIVE_COMPONENT' IN TYPEOF(sa))
             AND (sa.description = 'main component')) )) = 1) )) = 1;
END_ENTITY; -- structure_enclosure_element

ENTITY surface
  SUPERTYPE OF (ONEOF (elementary_surface,swept_surface,bounded_surface))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- surface

ENTITY surface_curve
  SUPERTYPE OF (intersection_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: ('BUILDING_DESIGN_SCHEMA.PCURVE' IN TYPEOF(associated_geometry[
             1])) OR (master_representation <> pcurve_s1);
    wr3: ('BUILDING_DESIGN_SCHEMA.PCURVE' IN TYPEOF(associated_geometry[
             2])) OR (master_representation <> pcurve_s2);
    wr4: NOT ('BUILDING_DESIGN_SCHEMA.PCURVE' IN TYPEOF(curve_3d));
END_ENTITY; -- surface_curve

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

ENTITY surface_of_revolution
  SUBTYPE OF (swept_surface);
    axis_position : axis1_placement;
  DERIVE
    axis_line : line := line(axis_position.location,vector(axis_position
                  .z,1));
END_ENTITY; -- surface_of_revolution

ENTITY surface_patch;
    parent_surface : bounded_surface;
    u_transition   : transition_code;
```

```
    v_transition   : transition_code;
    u_sense        : BOOLEAN;
    v_sense        : BOOLEAN;
  INVERSE
    using_surfaces : BAG [1:?] OF rectangular_composite_surface FOR
                        segments;
  WHERE
    wr1: NOT ('BUILDING_DESIGN_SCHEMA.CURVE_BOUNDED_SURFACE' IN TYPEOF(
              parent_surface));
END_ENTITY; -- surface_patch

ENTITY swept_area_solid
  SUPERTYPE OF (ONEOF (revolved_area_solid,extruded_area_solid))
  SUBTYPE OF (solid_model);
    swept_area : curve_bounded_surface;
  WHERE
    wr1: 'BUILDING_DESIGN_SCHEMA.PLANE' IN TYPEOF(swept_area.
              basis_surface);
END_ENTITY; -- swept_area_solid

ENTITY swept_surface
  SUPERTYPE OF (ONEOF (surface_of_linear_extrusion,
    surface_of_revolution))
  SUBTYPE OF (surface);
    swept_curve : curve;
END_ENTITY; -- swept_surface

ENTITY topological_representation_item
  SUPERTYPE OF (ONEOF (vertex,edge,face_bound,face,connected_face_set,
      loop ANDOR path))
  SUBTYPE OF (representation_item);
END_ENTITY; -- topological_representation_item

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

ENTITY torus
  SUBTYPE OF (geometric_representation_item);
    position     : axis1_placement;
    major_radius : positive_length_measure;
    minor_radius : positive_length_measure;
  WHERE
    wr1: major_radius > minor_radius;
END_ENTITY; -- torus

ENTITY trimmed_curve
  SUBTYPE OF (bounded_curve);
    basis_curve          : curve;
    trim_1               : SET [1:2] OF trimming_select;
    trim_2               : SET [1:2] OF trimming_select;
    sense_agreement      : BOOLEAN;
    master_representation : trimming_preference;
  WHERE
    wr1: (HIINDEX(trim_1) = 1) XOR (TYPEOF(trim_1[1])
         <> TYPEOF(trim_1[2]));
    wr2: (HIINDEX(trim_2) = 1) XOR (TYPEOF(trim_2[1])
         <> TYPEOF(trim_2[2]));
END_ENTITY; -- trimmed_curve

ENTITY truncated_pyramid
  SUBTYPE OF (boolean_result);
```

```
END_ENTITY; -- truncated_pyramid

ENTITY uniform_curve
  SUBTYPE OF (b_spline_curve);
END_ENTITY; -- uniform_curve

ENTITY uniform_surface
  SUBTYPE OF (b_spline_surface);
END_ENTITY; -- uniform_surface

ENTITY vector
  SUBTYPE OF (geometric_representation_item);
    orientation : direction;
    magnitude   : length_measure;
  WHERE
    wr1: magnitude >= 0;
END_ENTITY; -- vector

ENTITY versioned_action_request;
    id          : identifier;
    version     : label;
    purpose     : text;
    description : text;
END_ENTITY; -- versioned_action_request

ENTITY vertex
  SUBTYPE OF (topological_representation_item);
END_ENTITY; -- vertex

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

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

ENTITY week_of_year_and_day_date
  SUBTYPE OF (date);
    week_component : week_in_year_number;
    day_component  : OPTIONAL day_in_week_number;
END_ENTITY; -- week_of_year_and_day_date

RULE application_context_requires_ap_definition FOR (application_context,
                        application_protocol_definition);
WHERE
  wr1: SIZEOF(QUERY ( ac <* application_context | (NOT (SIZEOF(
           QUERY ( apd <* application_protocol_definition | ((ac :=: apd.
           application) AND (apd.
           application_interpreted_model_schema_name =
           'building_design_schema')) )) = 1)) )) = 0;

END_RULE; -- application_context_requires_ap_definition

RULE building_element_maps_into_building_section FOR (mapped_item,
           representation_relationship_with_transformation);

WHERE
  wr1: SIZEOF(QUERY ( mi <* mapped_item | ((SIZEOF(QUERY ( pdr <*
           USEDIN(mi.mapping_source.mapped_representation,
           'BUILDING_DESIGN_SCHEMA.' +
           'PROPERTY_DEFINITION_REPRESENTATION.' + 'USED_REPRESENTATION')
```

```
            | (SIZEOF(TYPEOF(pdr.definition.definition) * [
          'BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
          'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
          'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
          'BUILDING_DESIGN_SCHEMA.SPACE_ELEMENT',
          'BUILDING_DESIGN_SCHEMA.STRUCTURE_ENCLOSURE_ELEMENT']) = 1)))
          >= 1) AND (NOT (SIZEOF(QUERY ( r <* USEDIN(mi,
          'BUILDING_DESIGN_SCHEMA.' + 'REPRESENTATION.ITEMS') | (SIZEOF(
          QUERY ( pdr <* USEDIN(r,'BUILDING_DESIGN_SCHEMA.' +
          'PROPERTY_DEFINITION_REPRESENTATION.USED_REPRESENTATION') | (
          'BUILDING_DESIGN_SCHEMA.BUILDING_SECTION' IN TYPEOF(pdr.
          definition.definition)) )) >= 1) )) = 1))) )) = 0;
  wr2: SIZEOF(QUERY ( rrwt <*
          representation_relationship_with_transformation | ((SIZEOF(
          QUERY ( pdr <* USEDIN(rrwt\representation_relationship.rep_2,
          'BUILDING_DESIGN_SCHEMA.' +
          'PROPERTY_DEFINITION_REPRESENTATION.' + 'USED_REPRESENTATION')
            | (SIZEOF(TYPEOF(pdr.definition.definition) * [
          'BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
          'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
          'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
          'BUILDING_DESIGN_SCHEMA.SPACE_ELEMENT',
          'BUILDING_DESIGN_SCHEMA.STRUCTURE_ENCLOSURE_ELEMENT']) = 1)))
          >= 1) AND (NOT (SIZEOF(QUERY ( pdr <* USEDIN(rrwt\
          representation_relationship.rep_1,
          'BUILDING_DESIGN_SCHEMA.PROPERTY_DEFINITION_' +
          'REPRESENTATION.USED_REPRESENTATION') | (
          'BUILDING_DESIGN_SCHEMA.BUILDING_SECTION' IN TYPEOF(pdr.
          definition.definition)) )) = 1))) )) = 0;
  wr3: SIZEOF(QUERY ( rrwt_1 <*
          representation_relationship_with_transformation | (SIZEOF(
          QUERY ( rrwt_2 <*
          representation_relationship_with_transformation | ((SIZEOF(
          QUERY ( pdr <*
            USEDIN(rrwt_1\representation_relationship.rep_2,
          'BUILDING_DESIGN_SCHEMA.PROPERTY_DEFINITION_' +
          'REPRESENTATION.USED_REPRESENTATION') | (SIZEOF(TYPEOF(pdr.
          definition.definition) * [
          'BUILDING_DESIGN_SCHEMA.BUILDING_ELEMENT',
          'BUILDING_DESIGN_SCHEMA.FIXTURE_EQUIPMENT_ELEMENT',
          'BUILDING_DESIGN_SCHEMA.SERVICE_ELEMENT',
          'BUILDING_DESIGN_SCHEMA.SPACE_ELEMENT',
          'BUILDING_DESIGN_SCHEMA.' + 'STRUCTURE_ENCLOSURE_ELEMENT']) =
          1) )) >= 1) AND (rrwt_1\representation_relationship.rep_2 =
          rrwt_2\representation_relationship.rep_2)) )) >= 1) )) = 0;
END_RULE; -- building_element_maps_into_building_section

RULE compatible_dimension FOR (cartesian_point, direction,
          representation_context, geometric_representation_context);
WHERE
  wr1: SIZEOF(QUERY ( x <* cartesian_point | (SIZEOF(QUERY ( y <*
          geometric_representation_context | (item_in_context(x,y) AND (
          HIINDEX(x.coordinates) <> y.coordinate_space_dimension)) )) >
          0) )) = 0;
  wr2: SIZEOF(QUERY ( x <* direction | (SIZEOF(QUERY ( y <*
          geometric_representation_context | (item_in_context(x,y) AND (
          HIINDEX(x.direction_ratios) <>
          y.coordinate_space_dimension)) ))
          > 0) )) = 0;

END_RULE; -- compatible_dimension

RULE geometric_representation_context_3d FOR (
          geometric_representation_context);
```

```
  WHERE
    wr1: SIZEOF(QUERY ( grc <* geometric_representation_context | (NOT (
              grc.coordinate_space_dimension = 3)) )) = 0;

  END_RULE; -- geometric_representation_context_3d

  RULE restrict_application_context FOR (application_context);

  WHERE
    wr1: SIZEOF(QUERY ( ac <* application_context | (NOT (ac.application =
               'building shape composition')) )) = 0;

  END_RULE; -- restrict_application_context

  RULE restrict_origin_and_target FOR (mapped_item);

  WHERE
    wr1: SIZEOF(QUERY ( mi <* mapped_item | (NOT ((
              'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D' IN TYPEOF(mi.
              mapping_target)) AND (
              'BUILDING_DESIGN_SCHEMA.AXIS2_PLACEMENT_3D' IN TYPEOF(mi.
              mapping_source.mapping_origin)))) )) = 0;

  END_RULE; -- restrict_origin_and_target

  RULE shape_representation_subtype_exclusiveness FOR (
              shape_representation);
  WHERE
    wr1: SIZEOF(QUERY ( sr <* shape_representation | (NOT (SIZEOF(TYPEOF(
              sr) * ['BUILDING_DESIGN_SCHEMA.' +
              'ADVANCED_BREP_BUILDING_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.ADVANCED_CSG_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'ADVANCED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'ADVANCED_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.ADVANCED_WIRE_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.ELEMENTARY_CSG_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'ELEMENTARY_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
         'BUILDING_DESIGN_SCHEMA.ELEMENTARY_GEOMETRIC_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'ELEMENTARY_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'ELEMENTARY_WIRE_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.FACETED_CSG_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'FACETED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
            'BUILDING_DESIGN_SCHEMA.FACETED_GEOMETRIC_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'FACETED_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.FACETED_WIRE_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'GROUND_FACE_SPACE_BOUNDARY_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.SITE_REPRESENTATION']) <= 1)) )) = 0;
  END_RULE; -- shape_representation_subtype_exclusiveness

  RULE subtype_mandatory_geometric_set FOR (geometric_set);
  WHERE
    wr1: SIZEOF(QUERY ( gs <* geometric_set | (NOT (
              'BUILDING_DESIGN_SCHEMA.GEOMETRIC_CURVE_SET'
              IN TYPEOF(gs))) ))
              = 0;
  END_RULE; -- subtype_mandatory_geometric_set
```

```
  RULE subtype_mandatory_group FOR (group);
  WHERE
    wr1: SIZEOF(QUERY ( g <* group | (NOT
        ('BUILDING_DESIGN_SCHEMA.BUILDING_COMPONENT_CLASSIFICATION_GROUP'
             IN TYPEOF(g))) )) = 0;
  END_RULE; -- subtype_mandatory_group

  RULE subtype_mandatory_solid_model FOR (solid_model);
  WHERE
    wr1: SIZEOF(QUERY ( sm <* solid_model | (NOT (SIZEOF(TYPEOF(sm) * [
              'BUILDING_DESIGN_SCHEMA.CSG_SOLID',
              'BUILDING_DESIGN_SCHEMA.MANIFOLD_SOLID_BREP',
              'BUILDING_DESIGN_SCHEMA.EXTRUDED_AREA_SOLID',
              'BUILDING_DESIGN_SCHEMA.REVOLVED_AREA_SOLID',
              'BUILDING_DESIGN_SCHEMA.' +
              'ADVANCED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'ELEMENTARY_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
              'BUILDING_DESIGN_SCHEMA.' +
              'FACETED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION']) = 1)) ))
              = 0;
  END_RULE; -- subtype_mandatory_solid_model

  FUNCTION acyclic_curve_replica(
               rep: curve_replica;
               parent: curve
      ): BOOLEAN;
    IF NOT ('BUILDING_DESIGN_SCHEMA.CURVE_REPLICA' IN TYPEOF(parent))
         THEN
      RETURN(TRUE);
    END_IF;
    IF parent :=: rep THEN
      RETURN(FALSE);
    ELSE
      RETURN(acyclic_curve_replica(rep,parent\curve_replica.parent_curve));
    END_IF;

  END_FUNCTION; -- acyclic_curve_replica

  FUNCTION acyclic_mapped_representation(
               parent_set: SET OF representation;
               children_set: SET OF representation_item
      ): BOOLEAN;

    LOCAL
      i : INTEGER;
      x : SET OF representation_item;
      y : SET OF representation_item;
    END_LOCAL;
    x := QUERY ( z <* children_set | ('BUILDING_DESIGN_SCHEMA.MAPPED_ITEM'
        IN TYPEOF(z)) );
    IF SIZEOF(x) > 0 THEN
      REPEAT i := 1 TO HIINDEX(x) BY 1;
        IF x[i]\mapped_item.mapping_source.mapped_representation IN
            parent_set THEN
          RETURN(FALSE);
        END_IF;
        IF NOT acyclic_mapped_representation(parent_set + x[i]\mapped_item
            .mapping_source.mapped_representation,x[i]\mapped_item.
            mapping_source.mapped_representation.items) THEN
          RETURN(FALSE);
        END_IF;
      END_REPEAT;
    END_IF;
```

```
    x := children_set - x;
    IF SIZEOF(x) > 0 THEN
      REPEAT i := 1 TO HIINDEX(x) BY 1;
        y := QUERY ( z <* bag_to_set(USEDIN(x[i],'')) | (
            'BUILDING_DESIGN_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z)) );
        IF NOT acyclic_mapped_representation(parent_set,y) THEN
          RETURN(FALSE);
        END_IF;
      END_REPEAT;
    END_IF;
    RETURN(TRUE);

  END_FUNCTION; -- acyclic_mapped_representation

  FUNCTION acyclic_product_category_relationship(
              relation: product_category_relationship;
              children: SET OF product_category
      ): LOGICAL;

    LOCAL
      i              : INTEGER;
      x              : SET OF product_category_relationship;
      local_children : SET OF product_category;
    END_LOCAL;
    REPEAT i := 1 TO HIINDEX(children) BY 1;
      IF relation.category :=: children[i] THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
    x := bag_to_set(USEDIN(relation.category,'BUILDING_DESIGN_SCHEMA.' +
        'PRODUCT_CATEGORY_RELATIONSHIP.SUB_CATEGORY'));
    local_children := children + relation.category;
    IF SIZEOF(x) > 0 THEN
      REPEAT i := 1 TO HIINDEX(x) BY 1;
        IF NOT acyclic_product_category_relationship(x[i],local_children)
            THEN
          RETURN(FALSE);
        END_IF;
      END_REPEAT;
    END_IF;
    RETURN(TRUE);

  END_FUNCTION; -- acyclic_product_category_relationship

  FUNCTION acyclic_product_definition_relationship(
              relation: product_definition_relationship;
              relatives: SET OF product_definition;
              specific_relation: STRING
      ): LOGICAL;

    LOCAL
      i              : INTEGER;
      x              : SET OF product_definition_relationship;
      local_relatives : SET OF product_definition;
    END_LOCAL;
    REPEAT i := 1 TO HIINDEX(relatives) BY 1;
      IF relation.relating_product_definition :=: relatives[i] THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
    x := bag_to_set(USEDIN(relation.relating_product_definition,
        specific_relation));
    local_relatives := relatives + relation.relating_product_definition;
    IF SIZEOF(x) > 0 THEN
```

```
      REPEAT i := 1 TO HIINDEX(x) BY 1;
        IF NOT acyclic_product_definition_relationship(x[i],
            local_relatives,specific_relation) THEN
          RETURN(FALSE);
        END_IF;
      END_REPEAT;
    END_IF;
    RETURN(TRUE);

  END_FUNCTION; -- acyclic_product_definition_relationship

  FUNCTION associated_surface(
              arg: pcurve_or_surface
      ): surface;

    LOCAL
      surf : surface;
    END_LOCAL;
    IF 'BUILDING_DESIGN_SCHEMA.PCURVE' IN TYPEOF(arg) THEN
      surf := arg.basis_surface;
    ELSE
      surf := arg;
    END_IF;
    RETURN(surf);

  END_FUNCTION; -- associated_surface

  FUNCTION bag_to_set(
              the_bag: BAG OF GENERIC:intype
      ): SET OF GENERIC:intype;

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

  END_FUNCTION; -- bag_to_set

  FUNCTION base_axis(
              dim: INTEGER;
              axis1, axis2, axis3: direction
      ): LIST [2:3] OF direction;

    LOCAL
      u      : LIST [2:3] OF direction;
      vec    : direction;
      factor : REAL;
    END_LOCAL;
    IF dim = 3 THEN
      u[3] := NVL(normalise(axis3),direction([0,0,1]));
      u[1] := first_proj_axis(u[3],axis1);
      u[2] := second_proj_axis(u[3],u[1],axis2);
    ELSE
      u[3] := ?;
      IF EXISTS(axis1) THEN
        u[1] := normalise(axis1);
        u[2] := orthogonal_complement(u[1]);
        IF EXISTS(axis2) THEN
```

```
        factor := dot_product(axis2,u[2]);
        IF factor < 0 THEN
          u[2].direction_ratios[1] := -u[2].direction_ratios[1];
          u[2].direction_ratios[2] := -u[2].direction_ratios[2];
        END_IF;
      END_IF;
    ELSE
      IF EXISTS(axis2) THEN
        u[2] := normalise(axis2);
        u[1] := orthogonal_complement(u[2]);
        u[1].direction_ratios[1] := -u[1].direction_ratios[1];
        u[1].direction_ratios[2] := -u[1].direction_ratios[2];
      ELSE
        u[1].direction_ratios[1] := 1;
        u[1].direction_ratios[2] := 0;
        u[2].direction_ratios[1] := 0;
        u[2].direction_ratios[2] := 1;
      END_IF;
    END_IF;
  END_IF;
  RETURN(u);

END_FUNCTION; -- base_axis

FUNCTION boolean_choose(
            b: BOOLEAN;
            choice1, choice2: GENERIC:item
    ): GENERIC:item;
  IF b THEN
    RETURN(choice1);
  ELSE
    RETURN(choice2);
  END_IF;

END_FUNCTION; -- boolean_choose

FUNCTION build_2axes(
            ref_direction: direction
    ): LIST [2:2] OF direction;

  LOCAL
    u : LIST [2:2] OF direction;
  END_LOCAL;
  u[1] := NVL(normalise(ref_direction),direction([1,0]));
  u[2] := orthogonal_complement(u[1]);
  RETURN(u);

END_FUNCTION; -- build_2axes

FUNCTION build_axes(
            axis, ref_direction: direction
    ): LIST [3:3] OF direction;

  LOCAL
    u : LIST [3:3] OF direction;
  END_LOCAL;
  u[3] := NVL(normalise(axis),direction([0,0,1]));
  u[1] := first_proj_axis(u[3],ref_direction);
  u[2] := normalise(cross_product(u[3],u[1])).orientation;
  RETURN(u);

END_FUNCTION; -- build_axes

FUNCTION conditional_reverse(
```

```
            p: BOOLEAN;
            an_item: reversible_topology
    ): reversible_topology;
  IF p THEN
    RETURN(an_item);
  ELSE
    RETURN(topology_reversed(an_item));
  END_IF;

END_FUNCTION; -- conditional_reverse

FUNCTION constraints_composite_curve_on_surface(
            c: composite_curve_on_surface
    ): BOOLEAN;

  LOCAL
    n_segments : INTEGER := SIZEOF(c.segments);
  END_LOCAL;
  REPEAT k := 1 TO n_segments BY 1;
    IF (NOT ('BUILDING_DESIGN_SCHEMA.PCURVE' IN TYPEOF(c\composite_curve
        .segments[k].parent_curve))) AND (NOT (
        'BUILDING_DESIGN_SCHEMA.SURFACE_CURVE' IN TYPEOF(c\
        composite_curve.segments[k].parent_curve))) AND (NOT (
        'BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE_ON_SURFACE' IN TYPEOF(c\
        composite_curve.segments[k].parent_curve))) THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  RETURN(TRUE);

END_FUNCTION; -- constraints_composite_curve_on_surface

FUNCTION constraints_param_b_spline(
            degree, up_knots, up_cp: INTEGER;
            knot_mult: LIST OF INTEGER;
            knots: LIST OF parameter_value
    ): BOOLEAN;

  LOCAL
    k      : INTEGER;
    l      : INTEGER;
    sum    : INTEGER;
    result : BOOLEAN := TRUE;
  END_LOCAL;
  sum := knot_mult[1];
  REPEAT i := 2 TO up_knots BY 1;
    sum := sum + knot_mult[i];
  END_REPEAT;
  IF (degree < 1) OR (up_knots < 2) OR (up_cp < degree) OR (sum <> (
      degree + up_cp + 2)) THEN
    result := FALSE;
    RETURN(result);
  END_IF;
  k := knot_mult[1];
  IF (k < 1) OR (k > (degree + 1)) THEN
    result := FALSE;
    RETURN(result);
  END_IF;
  REPEAT i := 2 TO up_knots BY 1;
    IF (knot_mult[i] < 1) OR (knots[i] <= knots[i - 1]) THEN
      result := FALSE;
      RETURN(result);
    END_IF;
    k := knot_mult[i];
```

```
      IF (i < up_knots) AND (k > degree) THEN
        result := FALSE;
        RETURN(result);
      END_IF;
      IF (i = up_knots) AND (k > (degree + 1)) THEN
        result := FALSE;
        RETURN(result);
      END_IF;
    END_REPEAT;
    RETURN(result);

  END_FUNCTION; -- constraints_param_b_spline

  FUNCTION constraints_rectangular_composite_surface(
              s: rectangular_composite_surface
      ): BOOLEAN;
    REPEAT i := 1 TO s.n_u BY 1;
      REPEAT j := 1 TO s.n_v BY 1;
        IF NOT (('BUILDING_DESIGN_SCHEMA.B_SPLINE_SURFACE' IN TYPEOF(s.
            segments[i][j].parent_surface)) OR (
            'BUILDING_DESIGN_SCHEMA.RECTANGULAR_TRIMMED_SURFACE' IN TYPEOF(
            s.segments[i][j].parent_surface))) THEN
          RETURN(FALSE);
        END_IF;
      END_REPEAT;
    END_REPEAT;
    REPEAT i := 1 TO s.n_u - 1 BY 1;
      REPEAT j := 1 TO s.n_v BY 1;
        IF s.segments[i][j].u_transition = discontinuous THEN
          RETURN(FALSE);
        END_IF;
      END_REPEAT;
    END_REPEAT;
    REPEAT i := 1 TO s.n_u BY 1;
      REPEAT j := 1 TO s.n_v - 1 BY 1;
        IF s.segments[i][j].v_transition = discontinuous THEN
          RETURN(FALSE);
        END_IF;
      END_REPEAT;
    END_REPEAT;
    RETURN(TRUE);

  END_FUNCTION; -- constraints_rectangular_composite_surface

  FUNCTION cross_product(
              arg1, arg2: direction
      ): vector;

    LOCAL
      v2     : LIST [3:3] OF REAL;
      v1     : LIST [3:3] OF REAL;
      mag    : REAL;
      res    : direction;
      result : vector;
    END_LOCAL;
    IF (NOT EXISTS(arg1)) OR (arg1.dim = 2) OR (NOT EXISTS(arg2)) OR (arg2
       .dim = 2) THEN
      RETURN(?);
    ELSE
      BEGIN
        v1 := normalise(arg1).direction_ratios;
        v2 := normalise(arg2).direction_ratios;
        res.direction_ratios[1] := (v1[2] * v2[3]) - (v1[3] * v2[2]);
        res.direction_ratios[2] := (v1[3] * v2[1]) - (v1[1] * v2[3]);
```

```
        res.direction_ratios[3] := (v1[1] * v2[2]) - (v1[2] * v2[1]);
        mag := 0;
        REPEAT i := 1 TO 3 BY 1;
          mag := mag + (res.direction_ratios[i] * res.direction_ratios[i]);
        END_REPEAT;
        IF mag > 0 THEN
          result.orientation := res;
          result.magnitude := SQRT(mag);
        ELSE
          result.orientation := arg1;
          result.magnitude := 0;
        END_IF;
        RETURN(result);
      END;
    END_IF;

  END_FUNCTION; -- cross_product

  FUNCTION curve_weights_positive(
              b: rational_b_spline_curve
      ): BOOLEAN;

    LOCAL
      result : BOOLEAN := TRUE;
    END_LOCAL;
    REPEAT i := 0 TO b.upper_index_on_control_points BY 1;
      IF b.weights[i] <= 0 THEN
        result := FALSE;
        RETURN(result);
      END_IF;
    END_REPEAT;
    RETURN(result);

  END_FUNCTION; -- curve_weights_positive

  FUNCTION derive_dimensional_exponents(
              x: unit
      ): dimensional_exponents;

    LOCAL
      i      : INTEGER;
      result : dimensional_exponents := dimensional_exponents(0,0,0,0,0,0,
                0);
    END_LOCAL;
    IF 'BUILDING_DESIGN_SCHEMA.DERIVED_UNIT' IN TYPEOF(x) THEN
      REPEAT i := LOINDEX(x.elements) TO HIINDEX(x.elements) BY 1;
        result.length_exponent := result.length_exponent + (x.elements[i].
            exponent * x.elements[i].unit.dimensions.length_exponent);
        result.mass_exponent := result.mass_exponent + (x.elements[i].
            exponent * x.elements[i].unit.dimensions.mass_exponent);
        result.time_exponent := result.time_exponent + (x.elements[i].
            exponent * x.elements[i].unit.dimensions.time_exponent);
        result.electric_current_exponent := result.
            electric_current_exponent + (x.elements[i].exponent * x.
            elements[i].unit.dimensions.electric_current_exponent);
        result.thermodynamic_temperature_exponent := result.
            thermodynamic_temperature_exponent + (x.elements[i].exponent *
            x.elements[i].unit.dimensions.
            thermodynamic_temperature_exponent);
        result.amount_of_substance_exponent := result.
            amount_of_substance_exponent + (x.elements[i].exponent * x.
            elements[i].unit.dimensions.amount_of_substance_exponent);
        result.luminous_intensity_exponent := result.
            luminous_intensity_exponent + (x.elements[i].exponent * x.
```

```
            elements[i].unit.dimensions.luminous_intensity_exponent);
      END_REPEAT;
    ELSE
      result := x.dimensions;
    END_IF;
    RETURN(result);

  END_FUNCTION; -- derive_dimensional_exponents

  FUNCTION dimension_of(
              item: geometric_representation_item
      ): dimension_count;

    LOCAL
      x : SET OF representation;
      y : representation_context;
    END_LOCAL;
    x := using_representations(item);
    y := x[1].context_of_items;
    RETURN(y\geometric_representation_context.coordinate_space_dimension);

  END_FUNCTION; -- dimension_of

  FUNCTION dimensions_for_si_unit(
              n: si_unit_name
      ): dimensional_exponents;
    CASE n OF
      metre          :    RETURN(dimensional_exponents(1,0,0,0,0,0,0));
      gram           :    RETURN(dimensional_exponents(0,1,0,0,0,0,0));
      second         :    RETURN(dimensional_exponents(0,0,1,0,0,0,0));
      ampere         :    RETURN(dimensional_exponents(0,0,0,1,0,0,0));
      kelvin         :    RETURN(dimensional_exponents(0,0,0,0,1,0,0));
      mole           :    RETURN(dimensional_exponents(0,0,0,0,0,1,0));
      candela        :    RETURN(dimensional_exponents(0,0,0,0,0,0,1));
      radian         :    RETURN(dimensional_exponents(0,0,0,0,0,0,0));
      steradian      :    RETURN(dimensional_exponents(0,0,0,0,0,0,0));
      hertz          :    RETURN(dimensional_exponents(0,0,-1,0,0,0,0));
      newton         :    RETURN(dimensional_exponents(1,1,-2,0,0,0,0));
      pascal         :    RETURN(dimensional_exponents(-1,1,-2,0,0,0,0));
      joule          :    RETURN(dimensional_exponents(2,1,-2,0,0,0,0));
      watt           :    RETURN(dimensional_exponents(2,1,-3,0,0,0,0));
      coulomb        :    RETURN(dimensional_exponents(0,0,1,1,0,0,0));
      volt           :    RETURN(dimensional_exponents(2,1,-3,-1,0,0,0));
      farad          :    RETURN(dimensional_exponents(-2,-1,4,1,0,0,0));
      ohm            :    RETURN(dimensional_exponents(2,1,-3,-2,0,0,0));
      siemens        :    RETURN(dimensional_exponents(-2,-1,3,2,0,0,0));
      weber          :    RETURN(dimensional_exponents(2,1,-2,-1,0,0,0));
      tesla          :    RETURN(dimensional_exponents(0,1,-2,-1,0,0,0));
      henry          :    RETURN(dimensional_exponents(2,1,-2,-2,0,0,0));
      degree_celsius  :     RETURN(dimensional_exponents(0,0,0,0,1,0,0));
      lumen          :    RETURN(dimensional_exponents(0,0,0,0,0,0,1));
      lux            :    RETURN(dimensional_exponents(-2,0,0,0,0,0,1));
      becquerel      :    RETURN(dimensional_exponents(0,0,-1,0,0,0,0));
      gray           :    RETURN(dimensional_exponents(2,0,-2,0,0,0,0));
      sievert        :    RETURN(dimensional_exponents(2,0,-2,0,0,0,0));
      END_CASE;

  END_FUNCTION; -- dimensions_for_si_unit

  FUNCTION dot_product(
              arg1, arg2: direction
      ): REAL;

    LOCAL
```

```
      ndim   : INTEGER;
      scalar : REAL;
      vec1   : direction;
      vec2   : direction;
    END_LOCAL;
    IF (NOT EXISTS(arg1)) OR (NOT EXISTS(arg2)) THEN
      scalar := ?;
    ELSE
      IF arg1.dim <> arg2.dim THEN
        scalar := ?;
      ELSE
        BEGIN
          vec1 := normalise(arg1);
          vec2 := normalise(arg2);
          ndim := arg1.dim;
          scalar := 0;
          REPEAT i := 1 TO ndim BY 1;
            scalar := scalar + (vec1.direction_ratios[i] * vec2.
                direction_ratios[i]);
          END_REPEAT;
        END;
      END_IF;
    END_IF;
    RETURN(scalar);

  END_FUNCTION; -- dot_product

  FUNCTION edge_reversed(
              an_edge: edge
      ): edge;

    LOCAL
      the_reverse : edge;
    END_LOCAL;
    IF 'BUILDING_DESIGN_SCHEMA.ORIENTED_EDGE' IN TYPEOF(an_edge) THEN
      the_reverse := oriented_edge(an_edge\oriented_edge.edge_element,NOT
          an_edge\oriented_edge.orientation);
    ELSE
      the_reverse := oriented_edge(an_edge,FALSE);
    END_IF;
    RETURN(the_reverse);

  END_FUNCTION; -- edge_reversed

  FUNCTION face_bound_reversed(
              a_face_bound: face_bound
      ): face_bound;

    LOCAL
      the_reverse : face_bound;
    END_LOCAL;
    IF 'BUILDING_DESIGN_SCHEMA.FACE_OUTER_BOUND' IN TYPEOF(a_face_bound)
        THEN
      the_reverse := face_bound(a_face_bound\face_bound.bound,NOT
          a_face_bound\face_bound.orientation);
    ELSE
      the_reverse := face_bound(a_face_bound.bound,NOT a_face_bound.
          orientation);
    END_IF;
    RETURN(the_reverse);

  END_FUNCTION; -- face_bound_reversed

  FUNCTION face_reversed(
```

```
              a_face: face
    ): face;

  LOCAL
    the_reverse : face;
  END_LOCAL;
  IF 'BUILDING_DESIGN_SCHEMA.ORIENTED_FACE' IN TYPEOF(a_face) THEN
    the_reverse := oriented_face(a_face\oriented_face.face_element,NOT
        a_face\oriented_face.orientation);
  ELSE
    the_reverse := oriented_face(a_face,FALSE);
  END_IF;
  RETURN(the_reverse);

END_FUNCTION; -- face_reversed

FUNCTION first_proj_axis(
              z_axis, arg: direction
    ): direction;

  LOCAL
    x_vec  : vector;
    v      : direction;
    z      : direction;
    x_axis : direction;
  END_LOCAL;
  IF (NOT EXISTS(z_axis)) OR (NOT EXISTS(arg)) OR (arg.dim <> 3) THEN
    x_axis := ?;
  ELSE
    z_axis := normalise(z_axis);
    IF NOT EXISTS(arg) THEN
      IF z_axis <> direction([1,0,0]) THEN
        v := direction([1,0,0]);
      ELSE
        v := direction([0,1,0]);
      END_IF;
    ELSE
      IF cross_product(arg,z).magnitude = 0 THEN
        RETURN(?);
      ELSE
        v := normalise(arg);
      END_IF;
    END_IF;
    x_vec := scalar_times_vector(dot_product(v,z),z_axis);
    x_axis := vector_difference(v,x_vec).orientation;
    x_axis := normalise(x_axis);
  END_IF;
  RETURN(x_axis);

END_FUNCTION; -- first_proj_axis

FUNCTION get_basis_surface(
              c: curve_on_surface
    ): SET [0:2] OF surface;

  LOCAL
    surfs : SET [0:2] OF surface;
    n     : INTEGER;
  END_LOCAL;
  surfs := [];
  IF 'BUILDING_DESIGN_SCHEMA.PCURVE' IN TYPEOF(c) THEN
    surfs := [c\pcurve.basis_surface];
  ELSE
    IF 'BUILDING_DESIGN_SCHEMA.SURFACE_CURVE' IN TYPEOF(c) THEN
```

```
      n := SIZEOF(c\surface_curve.associated_geometry);
      REPEAT i := 1 TO n BY 1;
        surfs := surfs + associated_surface(c\surface_curve.
            associated_geometry[i]);
      END_REPEAT;
    END_IF;
  END_IF;
  IF 'BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE_ON_SURFACE' IN TYPEOF(c)
       THEN
    n := SIZEOF(c\composite_curve_on_surface.segments);
    surfs := get_basis_surface(c\composite_curve_on_surface.segments[1].
        parent_curve);
    IF n > 1 THEN
      REPEAT i := 2 TO n BY 1;
        surfs := surfs * get_basis_surface(c\composite_curve_on_surface.
            segments[i].parent_curve);
      END_REPEAT;
    END_IF;
  END_IF;
  RETURN(surfs);

END_FUNCTION; -- get_basis_surface

FUNCTION item_in_context(
             item: representation_item;
             cntxt: representation_context
    ): BOOLEAN;

  LOCAL
    i : INTEGER;
    y : BAG OF representation_item;
  END_LOCAL;
  IF SIZEOF(USEDIN(item,'BUILDING_DESIGN_SCHEMA.REPRESENTATION.ITEMS') *
      cntxt.representations_in_context) > 0 THEN
    RETURN(TRUE);
  ELSE
    y := QUERY ( z <* USEDIN(item,'') | (
        'BUILDING_DESIGN_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z)) );
    IF SIZEOF(y) > 0 THEN
      REPEAT i := 1 TO HIINDEX(y) BY 1;
        IF item_in_context(y[i],cntxt) THEN
          RETURN(TRUE);
        END_IF;
      END_REPEAT;
    END_IF;
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- item_in_context

FUNCTION leap_year(
             year: year_number
    ): BOOLEAN;
  IF (((year MOD 4) = 0) AND ((year MOD 100) <> 0)) OR ((year MOD 400) =
      0) THEN
    RETURN(TRUE);
  ELSE
    RETURN(FALSE);
  END_IF;

END_FUNCTION; -- leap_year

FUNCTION list_face_loops(
             f: face
```

```
    ): LIST [0:?] OF loop;

  LOCAL
    loops : LIST [0:?] OF loop := [];
  END_LOCAL;
  REPEAT i := 1 TO SIZEOF(f.bounds) BY 1;
    loops := loops + f.bounds[i].bound;
  END_REPEAT;
  RETURN(loops);

END_FUNCTION; -- list_face_loops

FUNCTION list_of_topology_reversed(
            a_list: list_of_reversible_topology_item
    ): list_of_reversible_topology_item;

  LOCAL
    the_reverse : list_of_reversible_topology_item;
  END_LOCAL;
  the_reverse := [];
  REPEAT i := 1 TO SIZEOF(a_list) BY 1;
    the_reverse := topology_reversed(a_list[i]) + the_reverse;
  END_REPEAT;
  RETURN(the_reverse);

END_FUNCTION; -- list_of_topology_reversed

FUNCTION list_to_array(
            lis: LIST [0:?] OF GENERIC:t;
            low, u: INTEGER
    ): ARRAY [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
    REPEAT i := 1 TO n BY 1;
      res[(low + i) - 1] := lis[i];
    END_REPEAT;
    RETURN(res);
  END_IF;

END_FUNCTION; -- list_to_array

FUNCTION list_to_set(
            l: LIST [0:?] OF GENERIC:t
    ): SET OF GENERIC:t;

  LOCAL
    s : SET OF GENERIC:t := [];
  END_LOCAL;
  REPEAT i := 1 TO SIZEOF(l) BY 1;
    s := s + l[i];
  END_REPEAT;
  RETURN(s);

END_FUNCTION; -- list_to_set

FUNCTION make_array_of_array(
            lis: LIST [1:?] OF LIST [1:?] OF GENERIC:t;
```

```
              low1, u1, low2, u2: INTEGER
    ): ARRAY [low1:u1] OF ARRAY [low2:u2] OF GENERIC:t;

  LOCAL
    n2   : INTEGER;
    n1   : INTEGER;
    res  : ARRAY [low1:u1] OF ARRAY [low2:u2] OF GENERIC:t;
    resl : LIST [1:?] OF ARRAY [low2:u2] OF GENERIC:t;
  END_LOCAL;
  n1 := SIZEOF(lis);
  n2 := SIZEOF(lis[1]);
  IF (n1 <> ((u1 - low1) + 1)) AND (n2 <> ((u2 - low2) + 1)) THEN
    RETURN(?);
  END_IF;
  REPEAT i := 1 TO n1 BY 1;
    IF SIZEOF(lis[i]) <> n2 THEN
      RETURN(?);
    END_IF;
  END_REPEAT;
  REPEAT i := 1 TO n1 BY 1;
    resl[i] := list_to_array(lis[i],low2,u2);
  END_REPEAT;
  res := list_to_array(resl,low1,u1);
  RETURN(res);

END_FUNCTION; -- make_array_of_array

FUNCTION mixed_loop_type_set(
              l: SET [0:?] OF loop
    ): LOGICAL;

  LOCAL
    i              : INTEGER;
    poly_loop_type : LOGICAL;
  END_LOCAL;
  IF SIZEOF(l) <= 1 THEN
    RETURN(FALSE);
  END_IF;
  poly_loop_type := 'BUILDING_DESIGN_SCHEMA.POLY_LOOP' IN TYPEOF(l[1]);
  REPEAT i := 2 TO SIZEOF(l) BY 1;
    IF ('BUILDING_DESIGN_SCHEMA.POLY_LOOP' IN TYPEOF(l[i])) <>
        poly_loop_type THEN
      RETURN(TRUE);
    END_IF;
  END_REPEAT;
  RETURN(FALSE);

END_FUNCTION; -- mixed_loop_type_set

FUNCTION msb_shells(
              brep: manifold_solid_brep
    ): SET [1:?] OF closed_shell;
  IF SIZEOF(QUERY ( brtyp <* TYPEOF(brep) | (brtyp LIKE
      '*.BREP_WITH_VOIDS') )) >= 1 THEN
    RETURN(brep\brep_with_voids.voids + brep.outer);
  ELSE
    RETURN([brep.outer]);
  END_IF;

END_FUNCTION; -- msb_shells

FUNCTION normalise(
              arg: vector_or_direction
    ): vector_or_direction;
```

```
    LOCAL
      ndim   : INTEGER;
      v      : direction;
      vec    : vector;
      mag    : REAL;
      result : vector_or_direction;
    END_LOCAL;
    IF NOT EXISTS(arg) THEN
      result := ?;
    ELSE
      ndim := arg.dim;
      IF 'BUILDING_DESIGN_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
        BEGIN
          vec := arg;
          v := arg.orientation;
          IF arg.magnitude = 0 THEN
            RETURN(?);
          ELSE
            vec.magnitude := 1;
          END_IF;
        END;
      ELSE
        v := arg;
      END_IF;
      mag := 0;
      REPEAT i := 1 TO ndim BY 1;
        mag := mag + (v.direction_ratios[i] * v.direction_ratios[i]);
      END_REPEAT;
      IF mag > 0 THEN
        mag := SQRT(mag);
        REPEAT i := 1 TO ndim BY 1;
          v.direction_ratios[i] := v.direction_ratios[i] / mag;
        END_REPEAT;
        IF 'BUILDING_DESIGN_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
          vec.orientation := v;
          result := vec;
        ELSE
          result := v;
        END_IF;
      ELSE
        RETURN(?);
      END_IF;
    END_IF;
    RETURN(result);

  END_FUNCTION; -- normalise

  FUNCTION orthogonal_complement(
              vec: direction
    ): direction;

    LOCAL
      result : direction;
    END_LOCAL;
    IF (vec.dim <> 2) OR (NOT EXISTS(vec)) THEN
      RETURN(?);
    ELSE
      result.direction_ratios[1] := -vec.direction_ratios[2];
      result.direction_ratios[2] := vec.direction_ratios[1];
      RETURN(result);
    END_IF;

  END_FUNCTION; -- orthogonal_complement
```

```
  FUNCTION path_head_to_tail(
              a_path: path
      ): LOGICAL;

    LOCAL
      n : INTEGER;
      p : BOOLEAN := TRUE;
    END_LOCAL;
    n := SIZEOF(a_path.edge_list);
    REPEAT i := 2 TO n BY 1;
      p := p AND (a_path.edge_list[i - 1].edge_end :=: a_path.edge_list[i]
          .edge_start);
    END_REPEAT;
    RETURN(p);

  END_FUNCTION; -- path_head_to_tail

  FUNCTION path_reversed(
              a_path: path
      ): path;

    LOCAL
      the_reverse : path;
    END_LOCAL;
    IF 'BUILDING_DESIGN_SCHEMA.ORIENTED_PATH' IN TYPEOF(a_path) THEN
      the_reverse := oriented_path(a_path\oriented_path.path_element,NOT
          a_path\oriented_path.orientation);
    ELSE
      the_reverse := oriented_path(a_path,FALSE);
    END_IF;
    RETURN(the_reverse);

  END_FUNCTION; -- path_reversed

  FUNCTION scalar_times_vector(
              scalar: REAL;
              vec: vector_or_direction
      ): vector;

    LOCAL
      v      : direction;
      mag    : REAL;
      result : vector;
    END_LOCAL;
    IF (NOT EXISTS(scalar)) OR (NOT EXISTS(vec)) THEN
      result := ?;
    ELSE
      IF 'BUILDING_DESIGN_SCHEMA.VECTOR' IN TYPEOF(vec) THEN
        v := vec.orientation;
        mag := scalar * vec.magnitude;
      ELSE
        v := vec;
        mag := scalar;
      END_IF;
      IF mag < 0 THEN
        REPEAT i := 1 TO SIZEOF(v.direction_ratios) BY 1;
          v.direction_ratios[i] := -v.direction_ratios[i];
        END_REPEAT;
        mag := -mag;
      END_IF;
      result.orientation := normalise(v);
      result.magnitude := mag;
    END_IF;
    RETURN(result);
```

```
END_FUNCTION; -- scalar_times_vector

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

  LOCAL
    temp   : vector;
    v      : direction;
    y_axis : vector;
  END_LOCAL;
  IF NOT EXISTS(arg) THEN
    v := direction([0,1,0]);
  ELSE
    v := arg;
  END_IF;
  temp := scalar_times_vector(dot_product(v,z_axis),z_axis);
  y_axis := vector_difference(v,temp);
  temp := scalar_times_vector(dot_product(v,x_axis),x_axis);
  y_axis := vector_difference(y_axis,temp);
  y_axis := normalise(y_axis);
  RETURN(y_axis.orientation);

END_FUNCTION; -- second_proj_axis

FUNCTION set_of_topology_reversed(
            a_set: set_of_reversible_topology_item
    ): set_of_reversible_topology_item;

  LOCAL
    the_reverse : set_of_reversible_topology_item;
  END_LOCAL;
  the_reverse := [];
  REPEAT i := 1 TO SIZEOF(a_set) BY 1;
    the_reverse := the_reverse + topology_reversed(a_set[i]);
  END_REPEAT;
  RETURN(the_reverse);

END_FUNCTION; -- set_of_topology_reversed

FUNCTION shell_reversed(
            a_shell: shell
    ): shell;

  LOCAL
    the_reverse : shell;
  END_LOCAL;
  IF 'BUILDING_DESIGN_SCHEMA.ORIENTED_OPEN_SHELL' IN TYPEOF(a_shell)
      THEN
    the_reverse := oriented_open_shell(a_shell\oriented_open_shell.
        open_shell_element,NOT a_shell\oriented_open_shell.orientation);
  ELSE
    IF 'BUILDING_DESIGN_SCHEMA.OPEN_SHELL' IN TYPEOF(a_shell) THEN
      the_reverse := oriented_open_shell(a_shell,FALSE);
    ELSE
      IF 'BUILDING_DESIGN_SCHEMA.ORIENTED_CLOSED_SHELL' IN TYPEOF(
          a_shell) THEN
        the_reverse := oriented_closed_shell(a_shell\
            oriented_closed_shell.closed_shell_element,NOT a_shell\
            oriented_closed_shell.orientation);
      ELSE
        IF 'BUILDING_DESIGN_SCHEMA.CLOSED_SHELL' IN TYPEOF(a_shell)
            THEN
          the_reverse := oriented_closed_shell(a_shell,FALSE);
```

```
          ELSE
            the_reverse := ?;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
    RETURN(the_reverse);

  END_FUNCTION; -- shell_reversed

  FUNCTION surface_weights_positive(
              b: rational_b_spline_surface
      ): BOOLEAN;

    LOCAL
      result : BOOLEAN := TRUE;
    END_LOCAL;
    REPEAT i := 0 TO b.u_upper BY 1;
      REPEAT j := 0 TO b.v_upper BY 1;
        IF b.weights[i][j] <= 0 THEN
          result := FALSE;
          RETURN(result);
        END_IF;
      END_REPEAT;
    END_REPEAT;
    RETURN(result);

  END_FUNCTION; -- surface_weights_positive

  FUNCTION topology_reversed(
              an_item: reversible_topology
      ): reversible_topology;
    IF 'BUILDING_DESIGN_SCHEMA.EDGE' IN TYPEOF(an_item) THEN
      RETURN(edge_reversed(an_item));
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.PATH' IN TYPEOF(an_item) THEN
      RETURN(path_reversed(an_item));
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.FACE_BOUND' IN TYPEOF(an_item) THEN
      RETURN(face_bound_reversed(an_item));
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.FACE' IN TYPEOF(an_item) THEN
      RETURN(face_reversed(an_item));
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.SHELL' IN TYPEOF(an_item) THEN
      RETURN(shell_reversed(an_item));
    END_IF;
    IF 'SET' IN TYPEOF(an_item) THEN
      RETURN(set_of_topology_reversed(an_item));
    END_IF;
    IF 'LIST' IN TYPEOF(an_item) THEN
      RETURN(list_of_topology_reversed(an_item));
    END_IF;
    RETURN(?);

  END_FUNCTION; -- topology_reversed

  FUNCTION using_representations(
              item: representation_item
      ): SET OF representation;

    LOCAL
      results            : SET OF representation;
      i                  : INTEGER;
```

```
    intermediate_items : SET OF representation_item;
    result_bag         : BAG OF representation;
  END_LOCAL;
  result_bag := USEDIN(item,
      'BUILDING_DESIGN_SCHEMA.REPRESENTATION.ITEMS');
  IF SIZEOF(result_bag) > 0 THEN
    REPEAT i := 1 TO HIINDEX(result_bag) BY 1;
      results := results + result_bag[i];
    END_REPEAT;
  END_IF;
  intermediate_items := QUERY ( z <* bag_to_set(USEDIN(item,'')) | (
      'BUILDING_DESIGN_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z)) );
  IF SIZEOF(intermediate_items) > 0 THEN
    REPEAT i := 1 TO HIINDEX(intermediate_items) BY 1;
      results := results + using_representations(intermediate_items[i]);
    END_REPEAT;
  END_IF;
  RETURN(results);

END_FUNCTION; -- using_representations

FUNCTION valid_advanced_csg_tree(
            tree_element: boolean_operand
  ): BOOLEAN;
  IF SIZEOF(TYPEOF(tree_element) * ['BUILDING_DESIGN_SCHEMA.BLOCK',
      'BUILDING_DESIGN_SCHEMA.TORUS',
      'BUILDING_DESIGN_SCHEMA.RIGHT_CIRCULAR_CYLINDER',
      'BUILDING_DESIGN_SCHEMA.SPHERE',
      'BUILDING_DESIGN_SCHEMA.RIGHT_CIRCULAR_CONE',
      'BUILDING_DESIGN_SCHEMA.' +
      'ADVANCED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
      'BUILDING_DESIGN_SCHEMA.EXTRUDED_AREA_SOLID',
      'BUILDING_DESIGN_SCHEMA.REVOLVED_AREA_SOLID',
      'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID']) = 1 THEN
    RETURN(TRUE);
  ELSE
    IF 'BUILDING_DESIGN_SCHEMA.BOOLEAN_RESULT' IN TYPEOF(tree_element)
        THEN
      IF NOT (tree_element\boolean_result.operator IN
  [BOOLEAN_OPERATOR.UNION, BOOLEAN_OPERATOR.DIFFERENCE]) THEN
        RETURN(FALSE);
      END_IF;
      IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN TYPEOF(
          tree_element\boolean_result.first_operand) THEN
        IF 'BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF(
            tree_element\boolean_result.first_operand\half_space_solid.
            base_surface) THEN
          IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN TYPEOF(
              tree_element\boolean_result.second_operand) THEN
            IF 'BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF(
                tree_element\boolean_result.second_operand\
                half_space_solid.base_surface) THEN
              RETURN(TRUE);
            ELSE
              RETURN(FALSE);
            END_IF;
          ELSE
            RETURN(valid_advanced_csg_tree(tree_element\boolean_result.
                second_operand));
          END_IF;
        ELSE
          RETURN(FALSE);
        END_IF;
      ELSE
```

```
          IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN TYPEOF(
              tree_element\boolean_result.second_operand) THEN
            IF 'BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF(
                tree_element\boolean_result.second_operand\half_space_solid
                .base_surface) THEN
              RETURN(valid_advanced_csg_tree(tree_element\boolean_result.
                  first_operand));
            ELSE
              RETURN(FALSE);
            END_IF;
          ELSE
            RETURN(valid_advanced_csg_tree(tree_element\boolean_result.
                first_operand) AND valid_advanced_csg_tree(tree_element\
                boolean_result.second_operand));
          END_IF;
        END_IF;
      END_IF;
    END_IF;
    RETURN(FALSE);

  END_FUNCTION; -- valid_advanced_csg_tree

  FUNCTION valid_advanced_wire_composition(
              sw_element: curve
      ): BOOLEAN;
    IF SIZEOF(['BUILDING_DESIGN_SCHEMA.B_SPLINE_CURVE',
        'BUILDING_DESIGN_SCHEMA.CONIC','BUILDING_DESIGN_SCHEMA.LINE',
        'BUILDING_DESIGN_SCHEMA.POLYLINE'] * TYPEOF(sw_element)) = 1 THEN
      RETURN(TRUE);
    ELSE
      IF 'BUILDING_DESIGN_SCHEMA.CURVE_REPLICA' IN TYPEOF(sw_element)
           THEN
        RETURN(valid_advanced_wire_composition(sw_element\curve_replica.
            parent_curve));
      ELSE
        IF 'BUILDING_DESIGN_SCHEMA.OFFSET_CURVE_3D' IN TYPEOF(sw_element)
             THEN
          RETURN(valid_advanced_wire_composition(sw_element\
              offset_curve_3d.basis_curve));
        ELSE
          IF 'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(sw_element)
               THEN
            RETURN(valid_faceted_wire_composition(sw_element\trimmed_curve
                .basis_curve));
          END_IF;
        END_IF;
      END_IF;
    END_IF;
    RETURN(FALSE);

  END_FUNCTION; -- valid_advanced_wire_composition

  FUNCTION valid_calendar_date(
              date: calendar_date
      ): LOGICAL;
    IF NOT ((1 <= date.day_component) AND (date.day_component <= 31))
         THEN
      RETURN(FALSE);
    END_IF;
    CASE date.month_component OF
      4   :    RETURN((1 <= date.day_component) AND (date.
          day_component <= 30));
      6   :    RETURN((1 <= date.day_component) AND (date.
          day_component <= 30));
```

```
      9   :    RETURN((1 <= date.day_component) AND (date.
            day_component <= 30));
      11  :    RETURN((1 <= date.day_component) AND (date.
            day_component <= 30));
      2   :    BEGIN
          IF leap_year(date.year_component) THEN
            RETURN((1 <= date.day_component)
            AND (date.day_component <= 29));
          ELSE
            RETURN((1 <= date.day_component)
            AND (date.day_component <= 28));
          END_IF;
        END;
      OTHERWISE  :         RETURN(TRUE);
      END_CASE;

  END_FUNCTION; -- valid_calendar_date

  FUNCTION valid_elementary_csg_tree(
              tree_element: boolean_operand
      ): BOOLEAN;
    IF SIZEOF(TYPEOF(tree_element) * ['BUILDING_DESIGN_SCHEMA.BLOCK',
        'BUILDING_DESIGN_SCHEMA.TORUS',
        'BUILDING_DESIGN_SCHEMA.RIGHT_CIRCULAR_CYLINDER',
        'BUILDING_DESIGN_SCHEMA.SPHERE',
        'BUILDING_DESIGN_SCHEMA.RIGHT_CIRCULAR_CONE',
        'BUILDING_DESIGN_SCHEMA.' +
        'ELEMENTARY_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION',
        'BUILDING_DESIGN_SCHEMA.EXTRUDED_AREA_SOLID',
        'BUILDING_DESIGN_SCHEMA.REVOLVED_AREA_SOLID',
        'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID']) = 1 THEN
      RETURN(TRUE);
    ELSE
      IF 'BUILDING_DESIGN_SCHEMA.BOOLEAN_RESULT' IN TYPEOF(tree_element)
          THEN
        IF NOT (tree_element\boolean_result.operator IN
    [BOOLEAN_OPERATOR.UNION, BOOLEAN_OPERATOR.DIFFERENCE]) THEN
          RETURN(FALSE);
        END_IF;
        IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN TYPEOF(
            tree_element\boolean_result.first_operand) THEN
          IF 'BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF(
              tree_element\boolean_result.first_operand\half_space_solid.
              base_surface) THEN
            IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN TYPEOF(
                tree_element\boolean_result.second_operand) THEN
              IF 'BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF(
                  tree_element\boolean_result.second_operand\
                  half_space_solid.base_surface) THEN
                RETURN(TRUE);
              ELSE
                RETURN(FALSE);
              END_IF;
            ELSE
              RETURN(valid_elementary_csg_tree(tree_element\boolean_result
                  .second_operand));
            END_IF;
          ELSE
            RETURN(FALSE);
          END_IF;
        ELSE
          IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN TYPEOF(
              tree_element\boolean_result.second_operand) THEN
            IF 'BUILDING_DESIGN_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF(
```

```
              tree_element\boolean_result.second_operand\half_space_solid
              .base_surface) THEN
            RETURN(valid_elementary_csg_tree(tree_element\boolean_result
               .first_operand));
          ELSE
            RETURN(FALSE);
          END_IF;
        ELSE
          RETURN(valid_elementary_csg_tree(tree_element\boolean_result.
              first_operand) AND valid_elementary_csg_tree(tree_element\
              boolean_result.second_operand));
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- valid_elementary_csg_tree

FUNCTION valid_elementary_wire_composition(
            sw_element: curve
    ): BOOLEAN;
  IF SIZEOF(['BUILDING_DESIGN_SCHEMA.LINE',
      'BUILDING_DESIGN_SCHEMA.CONIC'] * TYPEOF(sw_element)) = 1 THEN
    RETURN(TRUE);
  ELSE
    IF 'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(sw_element)
        THEN
      RETURN(valid_elementary_wire_composition(sw_element\trimmed_curve.
          basis_curve));
    ELSE
      IF 'BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE' IN TYPEOF(sw_element)
          THEN
        RETURN(SIZEOF(QUERY ( ccs <* sw_element\composite_curve.segments
         | (NOT valid_elementary_wire_composition(ccs.parent_curve)) ))
          = 0);
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- valid_elementary_wire_composition

FUNCTION valid_faceted_csg_tree(
            tree_element: boolean_operand
    ): BOOLEAN;
  IF SIZEOF(['BUILDING_DESIGN_SCHEMA.BLOCK','BUILDING_DESIGN_SCHEMA.' +
      'FACETED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION'] * TYPEOF(
      tree_element)) = 1 THEN
    RETURN(TRUE);
  ELSE
    IF 'BUILDING_DESIGN_SCHEMA.BOOLEAN_RESULT' IN TYPEOF(tree_element)
        THEN
      IF NOT (tree_element\boolean_result.operator IN
  [BOOLEAN_OPERATOR.UNION, BOOLEAN_OPERATOR.DIFFERENCE]) THEN
        RETURN(FALSE);
      END_IF;
      IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN TYPEOF(
          tree_element\boolean_result.first_operand) THEN
        IF 'BUILDING_DESIGN_SCHEMA.PLANE' IN TYPEOF(tree_element\
            boolean_result.first_operand\half_space_solid.base_surface)
            THEN
          IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN TYPEOF(
              tree_element\boolean_result.second_operand) THEN
```

```
            IF 'BUILDING_DESIGN_SCHEMA.PLANE' IN TYPEOF(tree_element\
                boolean_result.second_operand\half_space_solid.
                base_surface) THEN
              RETURN(TRUE);
            ELSE
              RETURN(FALSE);
            END_IF;
          ELSE
            RETURN(valid_faceted_csg_tree(tree_element\boolean_result.
                second_operand));
          END_IF;
        ELSE
          RETURN(FALSE);
        END_IF;
      ELSE
        IF 'BUILDING_DESIGN_SCHEMA.HALF_SPACE_SOLID' IN TYPEOF(
            tree_element\boolean_result.second_operand) THEN
          IF 'BUILDING_DESIGN_SCHEMA.PLANE' IN TYPEOF(tree_element\
             boolean_result.second_operand\half_space_solid.base_surface)
               THEN
            RETURN(valid_faceted_csg_tree(tree_element\boolean_result.
                first_operand));
          ELSE
            RETURN(FALSE);
          END_IF;
        ELSE
          RETURN(valid_faceted_csg_tree(tree_element\boolean_result.
              first_operand) AND valid_faceted_csg_tree(tree_element\
              boolean_result.second_operand));
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);

END_FUNCTION; -- valid_faceted_csg_tree

FUNCTION valid_faceted_wire_composition(
            sw_element: curve
    ): BOOLEAN;
  IF 'BUILDING_DESIGN_SCHEMA.POLYLINE' IN TYPEOF(sw_element) THEN
    RETURN(TRUE);
  ELSE
    IF 'BUILDING_DESIGN_SCHEMA.TRIMMED_CURVE' IN TYPEOF(sw_element)
         THEN
      IF SIZEOF(['BUILDING_DESIGN_SCHEMA.LINE',
          'BUILDING_DESIGN_SCHEMA.CONIC'] * TYPEOF(sw_element\
          trimmed_curve.basis_curve)) = 1 THEN
        RETURN(TRUE);
      ELSE
        RETURN(valid_faceted_wire_composition(sw_element\trimmed_curve.
            basis_curve));
      END_IF;
    ELSE
      IF 'BUILDING_DESIGN_SCHEMA.COMPOSITE_CURVE' IN TYPEOF(sw_element)
           THEN
        RETURN(SIZEOF(QUERY ( ccs <* sw_element\composite_curve.segments
             | (NOT valid_faceted_wire_composition(ccs.parent_curve)) ))
            = 0);
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
```

```
  END_FUNCTION; -- valid_faceted_wire_composition

  FUNCTION valid_units(
              m: measure_with_unit
      ): BOOLEAN;
    IF 'BUILDING_DESIGN_SCHEMA.LENGTH_MEASURE' IN TYPEOF(m.value_component)
         THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(1,0,0,0,0,0,0) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.MASS_MEASURE' IN TYPEOF(m.value_component)
         THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(0,1,0,0,0,0,0) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.TIME_MEASURE' IN TYPEOF(m.value_component)
         THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(0,0,1,0,0,0,0) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.ELECTRIC_CURRENT_MEASURE' IN TYPEOF(m.
        value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(0,0,0,1,0,0,0) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.THERMODYNAMIC_TEMPERATURE_MEASURE' IN
        TYPEOF(m.value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(0,0,0,0,1,0,0) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.AMOUNT_OF_SUBSTANCE_MEASURE' IN TYPEOF(m.
        value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(0,0,0,0,0,1,0) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.LUMINOUS_INTENSITY_MEASURE' IN TYPEOF(m.
        value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(0,0,0,0,0,0,1) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.PLANE_ANGLE_MEASURE' IN TYPEOF(m.
        value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(0,0,0,0,0,0,0) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.SOLID_ANGLE_MEASURE' IN TYPEOF(m.
        value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
```

```
          dimensional_exponents(0,0,0,0,0,0,0) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.AREA_MEASURE' IN TYPEOF(m.value_component)
         THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(2,0,0,0,0,0,0) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.VOLUME_MEASURE' IN TYPEOF(m.value_component)
         THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(3,0,0,0,0,0,0) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.RATIO_MEASURE' IN TYPEOF(m.value_component)
         THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(0,0,0,0,0,0,0) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.POSITIVE_LENGTH_MEASURE' IN TYPEOF(m.
        value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(1,0,0,0,0,0,0) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
    IF 'BUILDING_DESIGN_SCHEMA.POSITIVE_PLANE_ANGLE_MEASURE' IN TYPEOF(m.
        value_component) THEN
      IF derive_dimensional_exponents(m.unit_component) <>
          dimensional_exponents(0,0,0,0,0,0,0) THEN
        RETURN(FALSE);
      END_IF;
    END_IF;
    RETURN(TRUE);

  END_FUNCTION; -- valid_units

  FUNCTION vector_difference(
              arg1, arg2: vector_or_direction
      ): vector;

    LOCAL
      ndim   : INTEGER;
      mag2   : REAL;
      mag1   : REAL;
      mag    : REAL;
      res    : direction;
      vec1   : direction;
      vec2   : direction;
      result : vector;
    END_LOCAL;
    IF (NOT EXISTS(arg1)) OR (NOT EXISTS(arg2)) OR (arg1.dim <> arg2.dim)
         THEN
      result := ?;
    ELSE
      BEGIN
        IF 'BUILDING_DESIGN_SCHEMA.VECTOR' IN TYPEOF(arg1) THEN
          mag1 := arg1.magnitude;
```

```
          vec1 := arg1.orientation;
        ELSE
          mag1 := 1;
          vec1 := arg1;
        END_IF;
        IF 'BUILDING_DESIGN_SCHEMA.VECTOR' IN TYPEOF(arg2) THEN
          mag2 := arg2.magnitude;
          vec2 := arg2.orientation;
        ELSE
          mag2 := 1;
          vec2 := arg2;
        END_IF;
        vec1 := normalise(vec1);
        vec2 := normalise(vec2);
        ndim := SIZEOF(vec1.direction_ratios);
        mag := 0;
        REPEAT i := 1 TO ndim BY 1;
          res.direction_ratios[i] := (mag1 * vec1.direction_ratios[i]) - (
              mag2 * vec2.direction_ratios[i]);
          mag := mag + (res.direction_ratios[i] * res.direction_ratios[i]);
        END_REPEAT;
        IF mag > 0 THEN
          result.magnitude := SQRT(mag);
          result.orientation := res;
        ELSE
          result.magnitude := 0;
          result.orientation := vec1;
        END_IF;
      END;
    END_IF;
    RETURN(result);

  END_FUNCTION; -- vector_difference

END_SCHEMA; -- building_design_schema
```

Annex B
(normative)

AIM short names of entities

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

Entity names	Short names
ACTION	ACTION
ACTION_ASSIGNMENT	ACTASS
ACTION_METHOD	ACTMTH
ACTION_REQUEST_SOLUTION	ACRQSL
ACTION_REQUEST_STATUS	ACRQST
ADVANCED_BREP_BUILDING_SHAPE_REPRESENTATION	ABBSR
ADVANCED_CSG_SHAPE_REPRESENTATION	ACSR
ADVANCED_FACE	ADVFC
ADVANCED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION	AFWTSR
ADVANCED_SPACE_BOUNDARY_SHAPE_REPRESENTATION	ASBSR
ADVANCED_WIRE_SHAPE_REPRESENTATION	AWSR
APPLICATION_CONTEXT	APPCNT
APPLICATION_CONTEXT_ELEMENT	APCNEL
APPLICATION_PROTOCOL_DEFINITION	APPRDF
APPROVAL	APPRVL
APPROVAL_ASSIGNMENT	APPASS
APPROVAL_DATE_TIME	APDTTM
APPROVAL_PERSON_ORGANIZATION	APPROR
APPROVAL_ROLE	APPRL
APPROVAL_STATUS	APPSTT
ASSEMBLY_COMPONENT_USAGE	ASCMUS
AXIS1_PLACEMENT	AX1PLC

Table B.1 - AIM short names (continued)

AXIS2_PLACEMENT_2D	A2PL2D
AXIS2_PLACEMENT_3D	A2PL3D
BEZIER_CURVE	BZRCRV
BEZIER_SURFACE	BZRSRF
BLOCK	BLOCK
BOOLEAN_RESULT	BLNRSL
BOUNDARY_CURVE	BNDCR
BOUNDED_CURVE	BNDCRV
BOUNDED_SURFACE	BNDSRF
BREP_WITH_VOIDS	BRWTVD
BUILDING	BLDNG
BUILDING_COMPLEX	BLDCMP
BUILDING_COMPONENT_CLASSIFICATION_ASSIGNMENT	BCCA
BUILDING_COMPONENT_CLASSIFICATION_GROUP	BCCG
BUILDING_DESIGN_APPROVAL	BLDSAP
BUILDING_DESIGN_DATE_ASSIGNMENT	BDDA
BUILDING_DESIGN_ORGANIZATION_ASSIGNMENT	BDOA
BUILDING_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT	BDPAOA
BUILDING_DESIGN_PERSON_ASSIGNMENT	BDPA
BUILDING_DOCUMENT_REFERENCE	BLDCRF
BUILDING_ELEMENT	BLDELM
BUILDING_ELEMENT_ASSEMBLY	BLELAS
BUILDING_ELEMENT_GROUP	BLELGR
BUILDING_ITEM_IDENTIFICATION_ASSIGNMENT	BIIA
BUILDING_LEVEL	BLDLVL
BUILDING_SECTION	BLDSCT
B_SPLINE_CURVE	BSPCR
B_SPLINE_CURVE_WITH_KNOTS	BSCWK
B_SPLINE_SURFACE	BSPSR

Table B.1 - AIM short names (continued)

B_SPLINE_SURFACE_WITH_KNOTS	BSSWK
CALENDAR_DATE	CLNDT
CARTESIAN_POINT	CRTPNT
CARTESIAN_TRANSFORMATION_OPERATOR	CRTROP
CARTESIAN_TRANSFORMATION_OPERATOR_3D	CTO3
CHANGE	CHANGE
CHARACTERIZED_OBJECT	CHROBJ
CIRCLE	CIRCLE
CLASSIFICATION_TABLE	CLSTBL
CLOSED_SHELL	CLSSHL
COMPOSITE_CURVE	CMPCRV
COMPOSITE_CURVE_ON_SURFACE	CCOS
COMPOSITE_CURVE_SEGMENT	CMCRSG
CONIC	CONIC
CONICAL_SURFACE	CNCSRF
CONNECTED_FACE_SET	CNFCST
CONVERSION_BASED_UNIT	CNBSUN
CSG_SOLID	CSGSLD
CURVE	CURVE
CURVE_BOUNDED_SURFACE	CRBNSR
CURVE_REPLICA	CRVRPL
CYLINDRICAL_SURFACE	CYLSRF
DATE	DATE
DATE_ASSIGNMENT	DTASS
DATE_ROLE	DTRL
DEFINITIONAL_REPRESENTATION	DFNRPR
DEGENERATE_TOROIDAL_SURFACE	DGTRSR
DERIVED_UNIT	DRVUNT
DERIVED_UNIT_ELEMENT	DRUNEL

Table B.1 - AIM short names (continued)

DESCRIPTIVE_REPRESENTATION_ITEM	DSRPIT
DIMENSIONAL_EXPONENTS	DMNEXP
DIRECTION	DRCTN
DOCUMENT	DCMNT
DOCUMENT_REFERENCE	DCMRFR
DOCUMENT_TYPE	DCMTYP
DOCUMENT_USAGE_CONSTRAINT	DCUSCN
EDGE	EDGE
EDGE_CURVE	EDGCRV
EDGE_LOOP	EDGLP
ELEMENTARY_CSG_SHAPE_REPRESENTATION	ECSR
ELEMENTARY_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION	EFWTSR
ELEMENTARY_GEOMETRIC_SHAPE_REPRESENTATION	EGSR
ELEMENTARY_SPACE_BOUNDARY_SHAPE_REPRESENTATION	ESBSR
ELEMENTARY_SURFACE	ELMSRF
ELEMENTARY_WIRE_SHAPE_REPRESENTATION	EWSR
ELLIPSE	ELLPS
EXTRUDED_AREA_SOLID	EXARSL
FACE	FACE
FACETED_BREP	FCTBR
FACETED_CSG_SHAPE_REPRESENTATION	FCSR
FACETED_FACE_WITH_THICKNESS_SHAPE_REPRESENTATION	FFWTSR
FACETED_GEOMETRIC_SHAPE_REPRESENTATION	FGSR
FACETED_SPACE_BOUNDARY_SHAPE_REPRESENTATION	FSBSR
FACETED_WIRE_SHAPE_REPRESENTATION	FWSR
FACE_BOUND	FCBND
FACE_OUTER_BOUND	FCOTBN
FACE_SURFACE	FCSRF
FIXTURE_EQUIPMENT_ELEMENT	FXEQEL

Table B.1 - AIM short names (continued)

FUNCTIONALLY_DEFINED_TRANSFORMATION	FNDFTR
GEOMETRIC_CURVE_SET	GMCRST
GEOMETRIC_REPRESENTATION_CONTEXT	GMRPCN
GEOMETRIC_REPRESENTATION_ITEM	GMRPIT
GEOMETRIC_SET	GMTST
GLOBAL_UNIT_ASSIGNED_CONTEXT	GUAC
GROUND_FACE_SPACE_BOUNDARY_SHAPE_REPRESENTATION	GFSBSR
GROUP	GROUP
GROUP_ASSIGNMENT	GRPASS
HALF_SPACE_SOLID	HLSPSL
HYPERBOLA	HYPRBL
INTERSECTION_CURVE	INTCRV
LENGTH_MEASURE_WITH_UNIT	LMWU
LENGTH_UNIT	LNGUNT
LINE	LINE
LOOP	LOOP
MANIFOLD_SOLID_BREP	MNSLBR
MAPPED_ITEM	MPPITM
MEASURE_REPRESENTATION_ITEM	MSRPIT
MEASURE_WITH_UNIT	MSWTUN
NAMED_UNIT	NMDUNT
NAME_ASSIGNMENT	NMASS
NEGATIVE_COMPONENT	NGTCMP
OFFSET_CURVE_3D	OFCR3D
OPENING	OPNNG
OPEN_SHELL	OPNSHL
ORDINAL_DATE	ORDDT
ORGANIZATION	ORGNZT
ORGANIZATIONAL_PROJECT	ORGPRJ

Table B.1 - AIM short names (continued)

ORGANIZATION_ASSIGNMENT	ORGASS
ORGANIZATION_ROLE	ORGRL
ORIENTED_CLOSED_SHELL	ORCLSH
ORIENTED_EDGE	ORNEDG
ORIENTED_FACE	ORNFC
ORIENTED_OPEN_SHELL	OROPSH
ORIENTED_PATH	ORNPTH
OUTER_BOUNDARY_CURVE	OTBNCR
PARABOLA	PRBL
PARAMETRIC_REPRESENTATION_CONTEXT	PRRPCN
PATH	PATH
PCURVE	PCURVE
PERSON	PERSON
PERSON_AND_ORGANIZATION	PRANOR
PERSON_AND_ORGANIZATION_ASSIGNMENT	PAOA
PERSON_AND_ORGANIZATION_ROLE	PAOR
PERSON_ASSIGNMENT	PRSASS
PERSON_ROLE	PRSRL
PLACEMENT	PLCMNT
PLANE	PLANE
PLANE_ANGLE_MEASURE_WITH_UNIT	PAMWU
PLANE_ANGLE_UNIT	PLANUN
POINT	POINT
POLYLINE	PLYLN
POLY_LOOP	PLYLP
POSITIVE_COMPONENT	PSTCMP
PRODUCT	PRDCT
PRODUCT_CATEGORY	PRDCTG
PRODUCT_CATEGORY_RELATIONSHIP	PRCTRL

Table B.1 - AIM short names (continued)

PRODUCT_CONTEXT	PRDCNT
PRODUCT_DEFINITION	PRDDFN
PRODUCT_DEFINITION_CONTEXT	PRDFCN
PRODUCT_DEFINITION_FORMATION	PRDFFR
PRODUCT_DEFINITION_RELATIONSHIP	PRDFRL
PRODUCT_DEFINITION_SHAPE	PRDFSH
PRODUCT_DEFINITION_USAGE	PRDFUS
PRODUCT_RELATED_PRODUCT_CATEGORY	PRPC
PROPERTY_DEFINITION	PRPDFN
PROPERTY_DEFINITION_REPRESENTATION	PRDFRP
QUASI_UNIFORM_CURVE	QSUNCR
QUASI_UNIFORM_SURFACE	QSUNSR
RATIONAL_B_SPLINE_CURVE	RBSC
RATIONAL_B_SPLINE_SURFACE	RBSS
RECESS	RECESS
RECTANGULAR_COMPOSITE_SURFACE	RCCMSR
RECTANGULAR_TRIMMED_SURFACE	RCTRSR
REPRESENTATION	RPRSNT
REPRESENTATION_CONTEXT	RPRCNT
REPRESENTATION_ITEM	RPRITM
REPRESENTATION_MAP	RPRMP
REPRESENTATION_RELATIONSHIP	RPRRLT
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION	RRWT
REVOLVED_AREA_SOLID	RVARSL
RIGHT_CIRCULAR_CONE	RGCRCN
RIGHT_CIRCULAR_CYLINDER	RGCRCY
SERVICE_ELEMENT	SRVELM
SHAPE_ASPECT	SHPASP
SHAPE_ASPECT_RELATIONSHIP	SHASRL

Table B.1 - AIM short names (continued)

SHAPE_DEFINITION_REPRESENTATION	SHDFRP
SHAPE_REPRESENTATION	SHPRPR
SITE	SITE
SITE_REPRESENTATION	STRPR
SI_UNIT	SUNT
SOLID_MODEL	SLDMDL
SPACE_ELEMENT	SPCELM
SPHERE	SPHERE
SPHERICAL_SURFACE	SPHSRF
STRUCTURE_ENCLOSURE_ELEMENT	STENEL
SURFACE	SRFC
SURFACE_CURVE	SRFCRV
SURFACE_OF_LINEAR_EXTRUSION	SL
SURFACE_OF_REVOLUTION	SROFRV
SURFACE_PATCH	SRFPTC
SWEPT_AREA_SOLID	SWARSL
SWEPT_SURFACE	SWPSRF
TOPOLOGICAL_REPRESENTATION_ITEM	TPRPIT
TOROIDAL_SURFACE	TRDSRF
TORUS	TORUS
TRIMMED_CURVE	TRMCRV
TRUNCATED_PYRAMID	TRNPYR
UNIFORM_CURVE	UNFCRV
UNIFORM_SURFACE	UNFSRF
VECTOR	VECTOR
VERSIONED_ACTION_REQUEST	VRACRQ
VERTEX	VERTEX
VERTEX_LOOP	VRTLP
VERTEX_POINT	VRTPNT

Table B.1 - AIM short names (concluded)

WEEK_OF_YEAR_AND_DAY_DATE	WOYADD

Annex C
(normative)

Implementation method specific requirements

The implementation method defines what types of exchange behaviour are required with respect to this part of ISO 10303. Conformance to this part of ISO 10303 shall be realized in an exchange structure. The file format shall be encoded according to the syntax and EXPRESS language mapping defined in ISO 10303-21 and the AIM defined in annex A of this part of ISO 10303. The header of the exchange structure shall identify the use of this part of ISO 10303 by the schema name ‘building_-design_schema’.

Annex D
(normative)

Protocol Implementation Conformance Statement (PICS) proforma

The Protocol Implementation Conformance Statement (PICS) proforma is supplied for completion by the person or organization (the client) requesting conformance testing. Its purpose is to ascertain the scope of claimed conformance to a particular application protocol by an implementation under test (IUT) using a defined implementation method. Through the completion of this form, the PICS Proforma becomes a PICS.

The information contained in the PICS is used to configure an appropriate executable test suite for use by the client.

Six conformance classes are identified in this part of ISO 10303. A conforming implementation shall support at least one conformance class. Each class specifies a subset of ISO 10303-225 AIM constructs. These classes are detailed in clause 6 of ISO 10303-225.

Questions:

1. Please provide an identifier for the product or system for which conformance is claimed:

   Product name and current version number:_____

2. Please indicate the implementation method chosen:

   — ISO 10303-21 Exchange Structure -- preprocessor
   Preprocessor name and current version number:_____

   — ISO 10303-21 Exchange Structure -- postprocessor
   Postprocessor name and current version number:_____

3. Please indicate the classes for which conformance is claimed:

Annex E
(normative)

Information object registration

E.1 Document identification

In order to provide for unambiguous identification of an information object in an open system, the object identifier

{ iso standard 10303 part(225) 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

In order to provide for unambiguous identification of the schema specifications given in this application protocol in an open information system, object identifiers are assigned as follows:

{ iso standard 10303 part(225) version(0) object(1) building-design-schema(1) }

is assigned to the building_design_schema expanded schema (see Annex A).

{ iso standard 10303 part(225) version(0) object(1) building-design-schema(2) }

is assigned to the building_design_schema short form schema (see 5.2).

The meaning of these values is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

Annex F
(informative)

Application activity model

The application activity model (AAM) is provided to aid in understanding the scope and information requirements defined in this application protocol. The model is presented as a set of figures that contain the activity diagrams and a set of definitions of the activities and their data.

The figures use modified IDEF0 function modelling.**[1]** Each activity may be decomposed to provide more detail. If an activity has been decomposed, a separate figure is included.

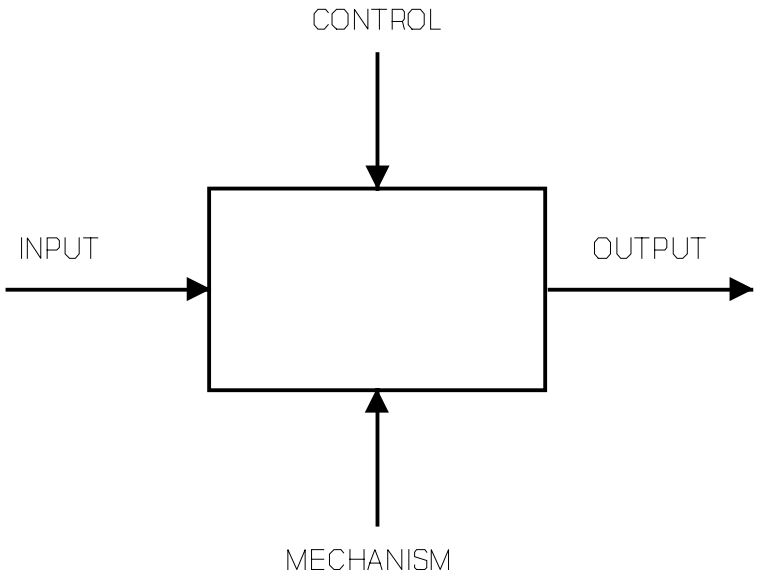


Figure F.1 - IDEF0 basic notation

As with any IDEF0 model, the application activity model is dependent on a particular viewpoint and purpose. The viewpoint of the application activity model is the user(s) of the designs for a buildings, including architect, engineer, and builder.

F.1 Application activity model definitions and abbreviations

The following terms are used in the application activity model. Terms marked with an asterisk are outside the scope of this application protocol.

The definitions given in this annex do not supersede the definitions given in the main body of the text.

F.1.1 Approved preliminary building design: a preliminary building design that has been approved by agents involved in the building design process, such as customer, architect, engineer, and authorities.

F.1.2 Approved detailed building design: a building design that has been approved by the customer, architect, engineer and the authorities responsible for evaluating the design.

F.1.3 As-built documentation: documentation of the building as it was erected. It is usually prepared when the building is finished and reflects all design changes.

F.1.4 Building:* a free-standing, man-made structure intended to shelter an occupancy from the environment, and may provide facilities to perform various functions.

NOTE - As defined here, the building is the physical structure that exists in the real world.

F.1.5 Building design practice and experience:* the knowledge and skills obtained over time by the individual participating in the design and construction of a building.

F.1.6 Building material:* the material components used to construct the building structure.

F.1.7 Building requirements:* the design and performance requirements of a building that are derived from customer requirements, engineering and construction principles.

F.1.8 Building standards and codes:* government and industry requirements for the performance or behavioral characteristics of a building. Most standards and codes prescribe safety requirements such as structural strength and fire resistance.

F.1.9 Building elements: the items installed or constructed within the building after the structure has been erected.

F.1.10 Construct building: the actions taken by the construction company to erect the building.

F.1.11 Construct, manage, and maintain building: the building is constructed by the construction company and conveyed to the customer. The customer is then responsible for the use, management and maintenance of the building.

F.1.12 Contract documents:* the legal documentation for a building that represents the agreement between design professionals, contractor and owner.

F.1.13 Create "as-built" documentation: after completion of the building construction, the constructed configuration of the building is documented for management and maintenance of building and serving as a baseline for changes to the original building design.

F.1.14 Create detailed design/award contract: completion of an approved building design with respect to specific details required for building construction, the solicitation of tenders, and the award of the construction contract.

F.1.15 Create and approve preliminary building design: the agents involved in the design of building create the preliminary building design based on the building requirements and obtain approval for the preliminary design from customer and authorities.

F.1.16 Create preliminary building design: the architect, in conjunction with building engineers, creates initial design for the building based on customer requirements. Preliminary design specifies the

primary functional characteristics (such as arrangement of rooms and main structural elements) of the building, but omits detailed information required for construction; initial design is subject to change.

F.1.17 Customer-approved preliminary building design: a preliminary building design that has met the customer requirements and has been approved by the customer.

F.1.18 Customer requirements:* the desires, guidelines, criteria, or specifications provided by the customer for the performance or appearance of a building.

F.1.19 Customer requirements for management and maintenance:* the desires, guidelines, criteria, or specifications provide by the customer for the management and maintenance of the building.

F.1.20 Design change requests: requests from customers or building authorities to alter to the building design.

F.1.21 Design, construct, and manage building complex: the activities necessary to design, build, and operate a building or complex of buildings.

F.1.22 Detailed building design: the approved building design that is completed with sufficient detail to construct the building.

F.1.23 Establish building requirements: the architect and customer establish the functional requirements for the building to be constructed.

F.1.24 Install/construct building elements:* the actions taken by the construction company or subcontractor to construct or install components of a building including systems (plumbing, electrical, HVAC), cladding, glazing, and painting/plastering.

F.1.25 Maintain building: actions taken the customer or subcontractor to ensure the continued satisfactory performance of the elements of the building, such as painting of metal structures to prevent rust, or preventive maintenance for air conditioning systems.

F.1.26 Manage building: the actions taken to control the use and maintenance of the building, such as cleaning and tenant relocation.

F.1.27 Obtain approval of authorities: government-sanctioned agents evaluate and approve the initial design based on architectural and engineering standards and building codes, such as structural requirements, fires codes, thermal insulation codes, and seismic safety.

F.1.28 Obtain customer approval of detailed design: detailed design submitted to, reviewed by, and approved by the customer. Requests for changes to the detailed design are sent to the architect/engineer.

F.1.29 Obtain customer approval of preliminary building design: initial design submitted to, reviewed by, and approved by the customer.

F.1.30 Preliminary building design: a building design that meets customer requirements. It does not show all the details, but only main aspects of the building to be constructed.

F.1.31 Produce detailed design: architects and engineers specify design details required by a construction company for preparation of tender, acquisition of building materials, and construction of building. Detailed design documents consists of drawings and building specifications. The customer also provides input to the design process.

F.1.32 Produce tender documents:* based on detailed design documents, additional tender documents, such as the bill of quantities, are prepared.

F.1.33 Solicit tenders and award contract:* architects, engineers, or customers identify potential construction companies and invite them to prepare tenders based on the detailed design documents. Once the tenders are received, they are reviewed, evaluated, and a construction company is selected to receive the contract award.

F.1.34 Specify design details for compliance to standards: the extension of the preliminary design by architects and engineers to meet standards and codes.

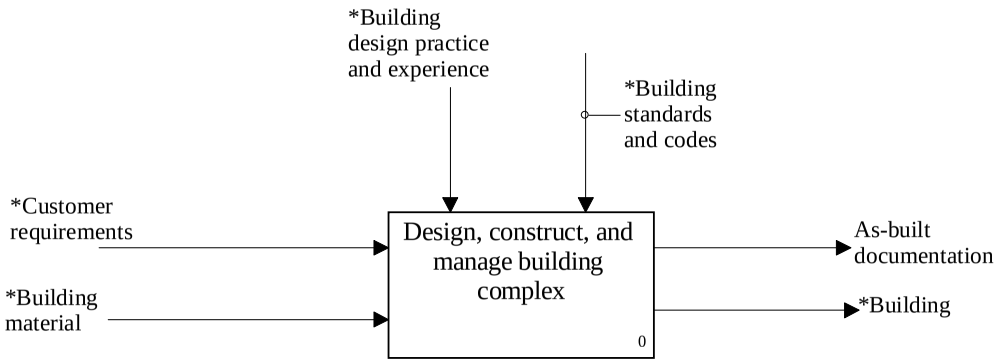
F.1.35 Standards for tender documents:* requirements for preparation and presentation of tender document.

F.1.36 Tender documents:* the documentation submitted to construction companies to solicit bids (i.e., tenders) for construction of a building. The tender documents include not only legal contracting documents, but the detailed design documents as well.

F.2 Application activity model diagrams

The application activity model is given in figures F.2 through F.5. Activities and data flows that are out of scope are marked with asterisks.

USED AT:	AUTHOR:Haas/Burkett	DATE:06/19/96		WORKING	READER DATE	CONTEXT:
	PROJECT: AP 225 Building Structures	REV:		DRAFT		
				RECOMMENDED		Top
	NOTES: 1 2 3 4 5 6 7 8 9 10		X	PUBLICATION		



Note: All inputs and controls are out of scope because all information within the scope of the AP originates within the activity Design, construct, and manage building complex.

NODE: A-0	TITLE: Design, construct, and manage building complex	NUMBER:

Figure F.2 - A-0: design, construct, and manage building in IDEF0

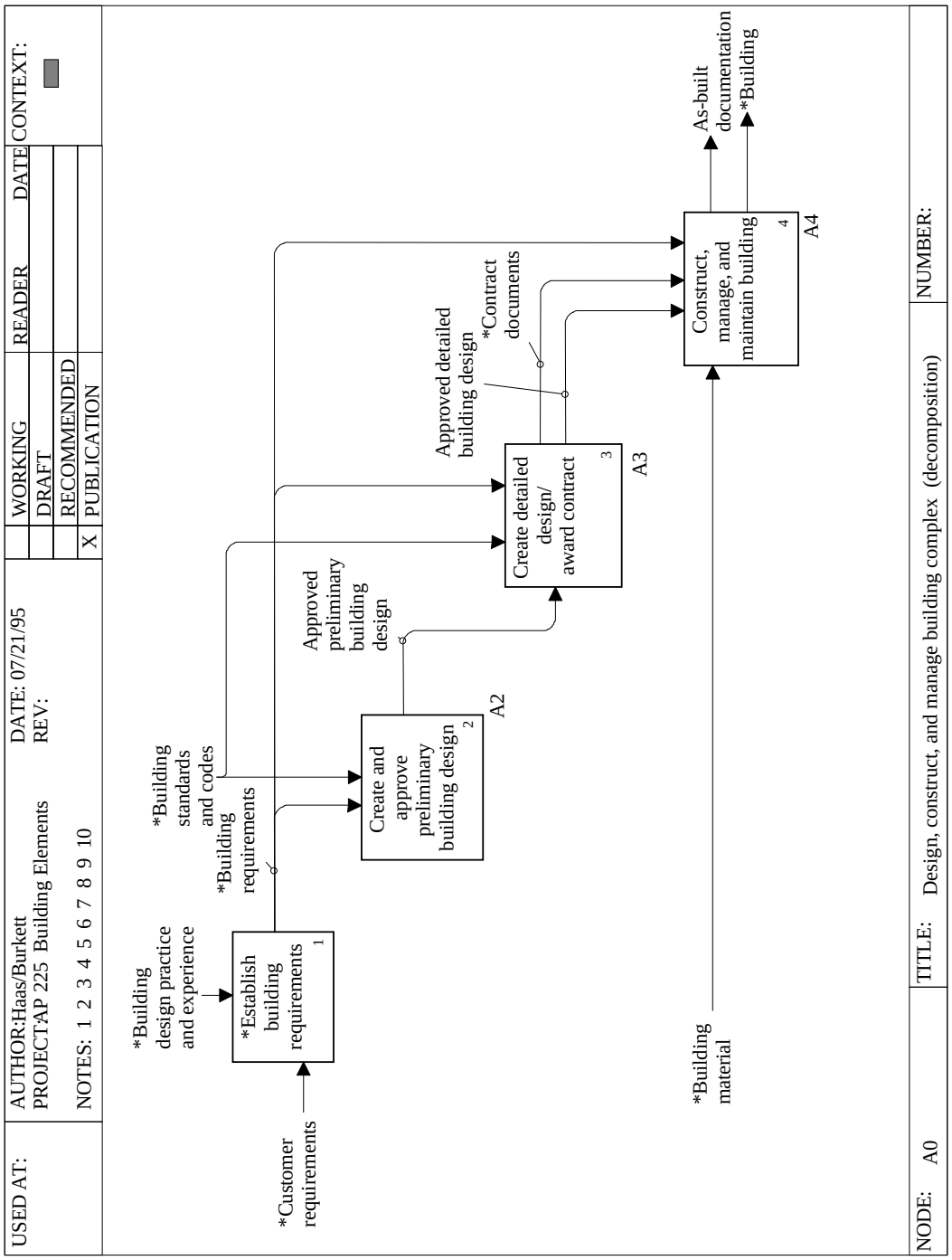


Figure F.3 - A0: design, construct, and manage building (decomposition) in IDEF0

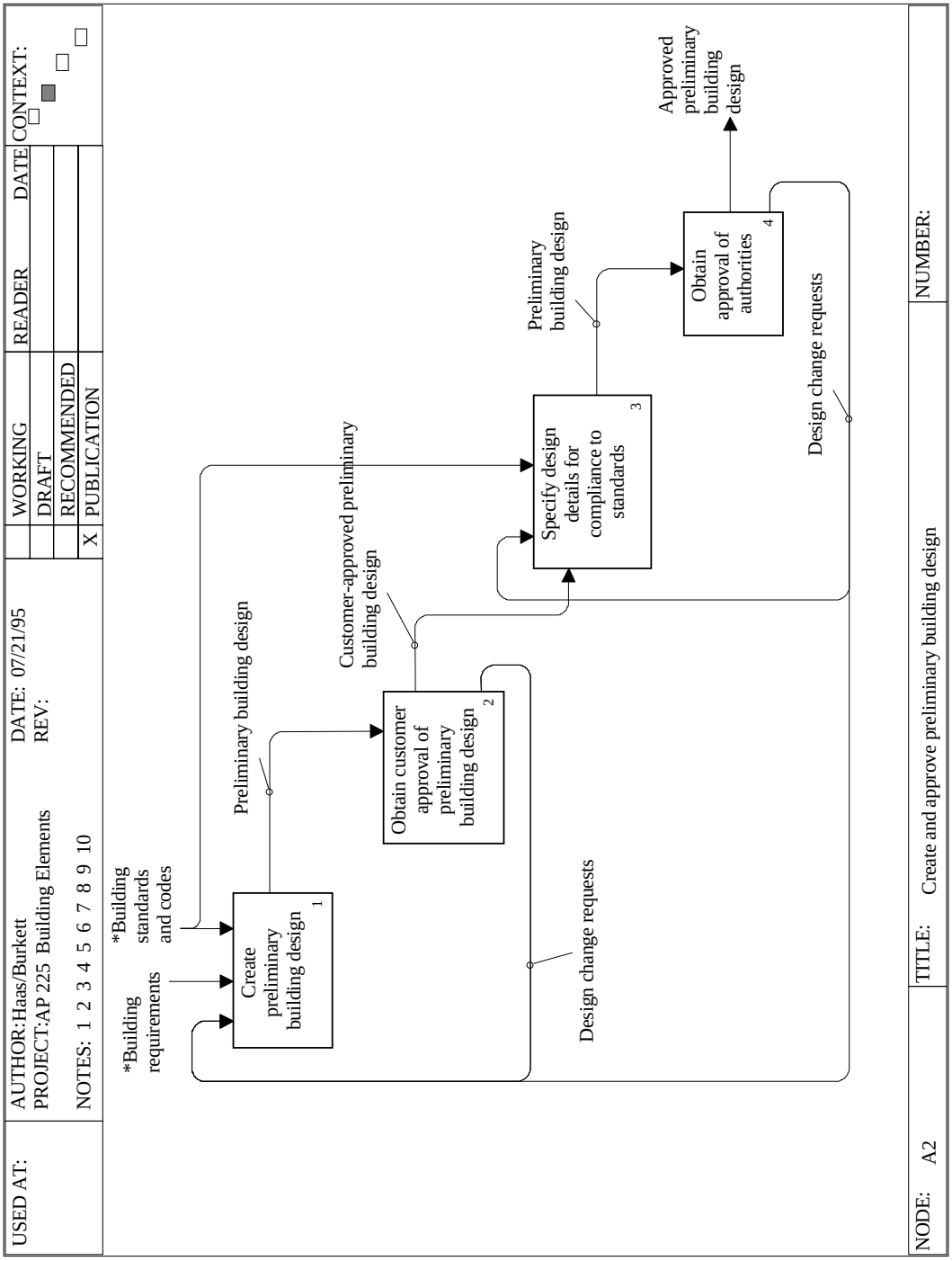


Figure F.4 - A2: create and approve preliminary building design in IDEF0

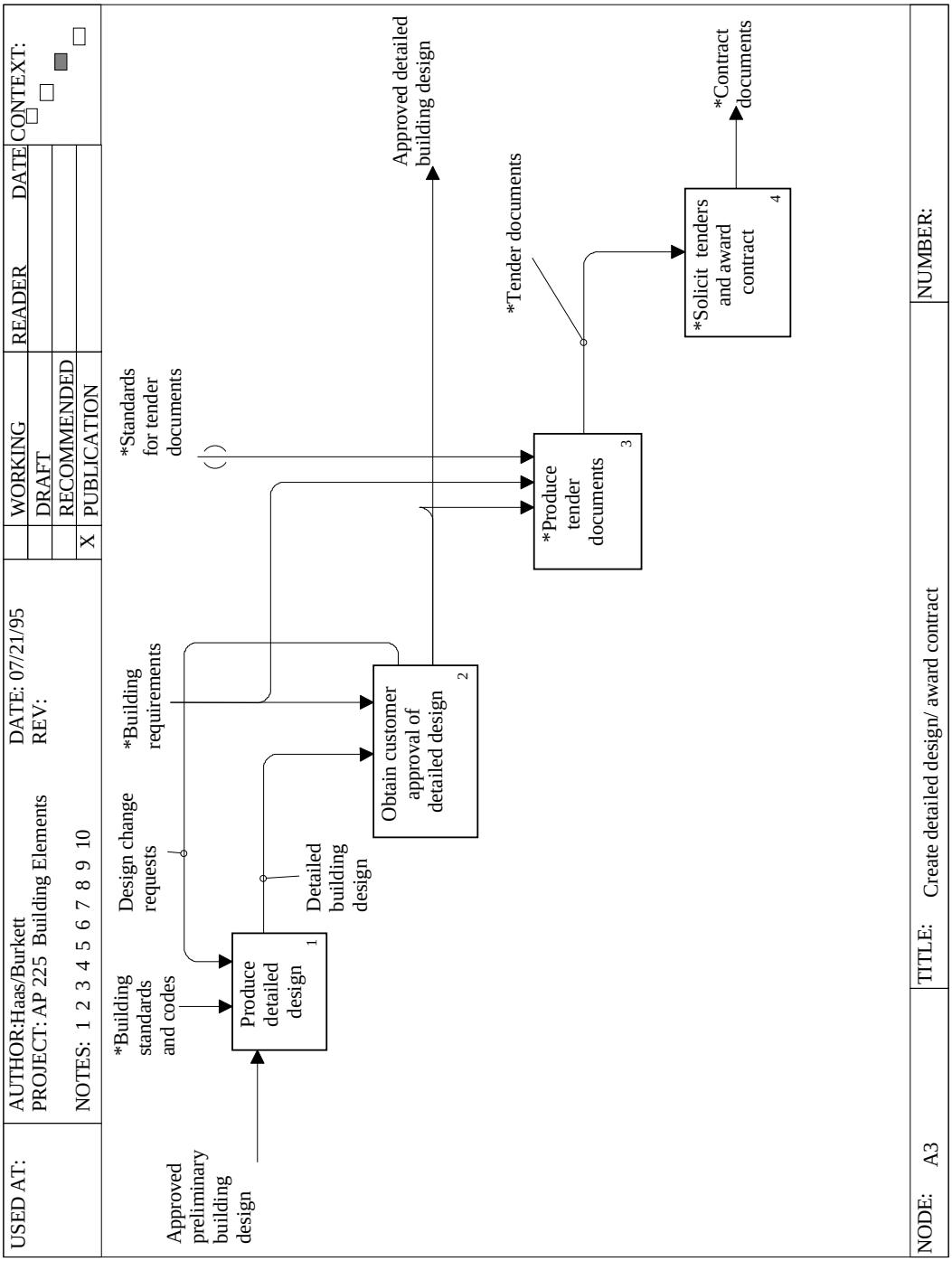


Figure F.5 - A3: create detailed design/award contract in IDEF0

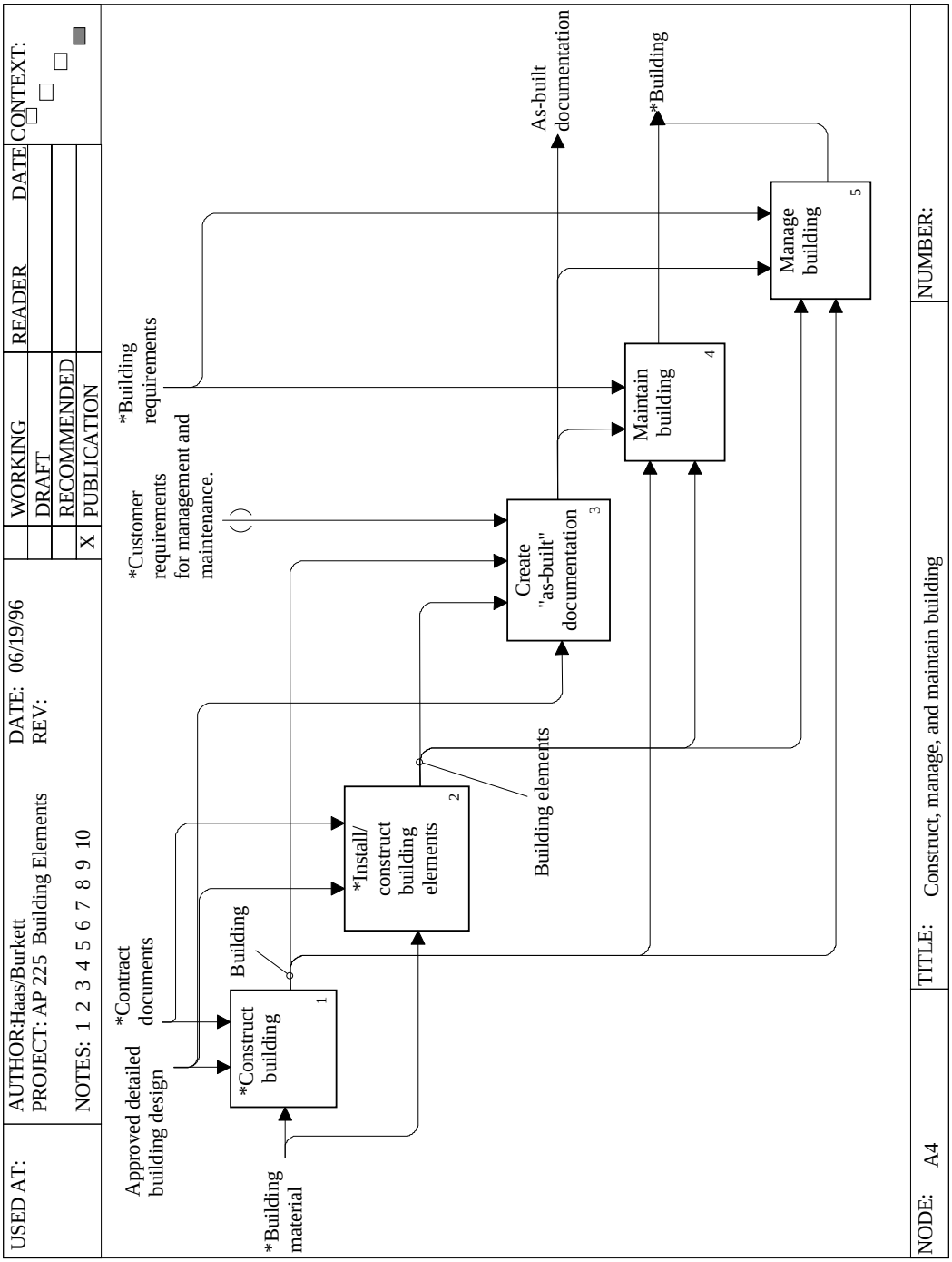


Figure F.6 - A4: construct, manage, and maintain building in IDEF0

Annex G
(informative)

Application reference model

This annex provides the application reference model for part of ISO 10303 and is given in figures G.1 through G.7. The application reference 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 EXPRESS-G modelling language. The application reference model is independent of any implementation method.

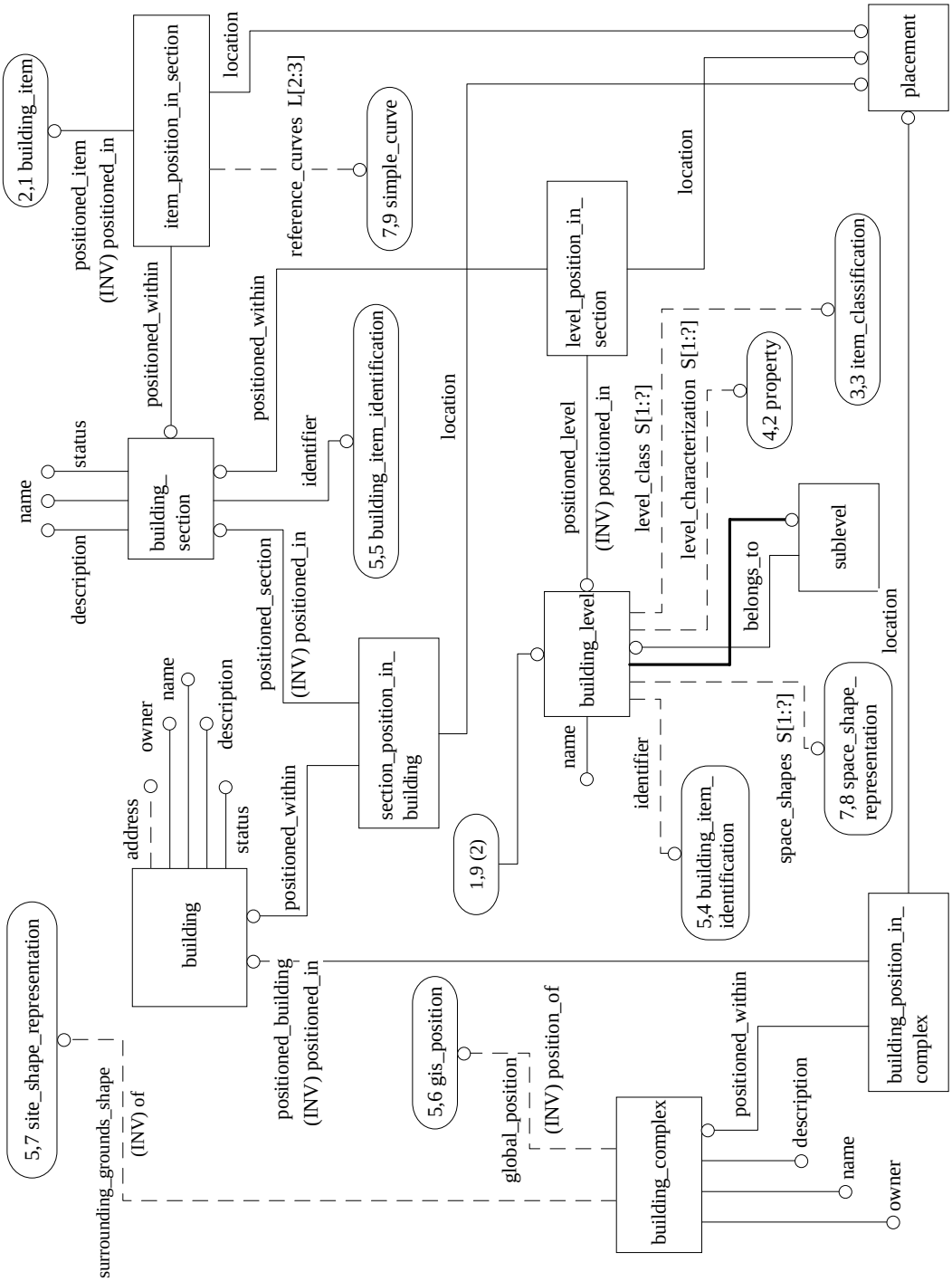


Figure G.1 - ARM diagram 1 of 7 in EXPRESS-G

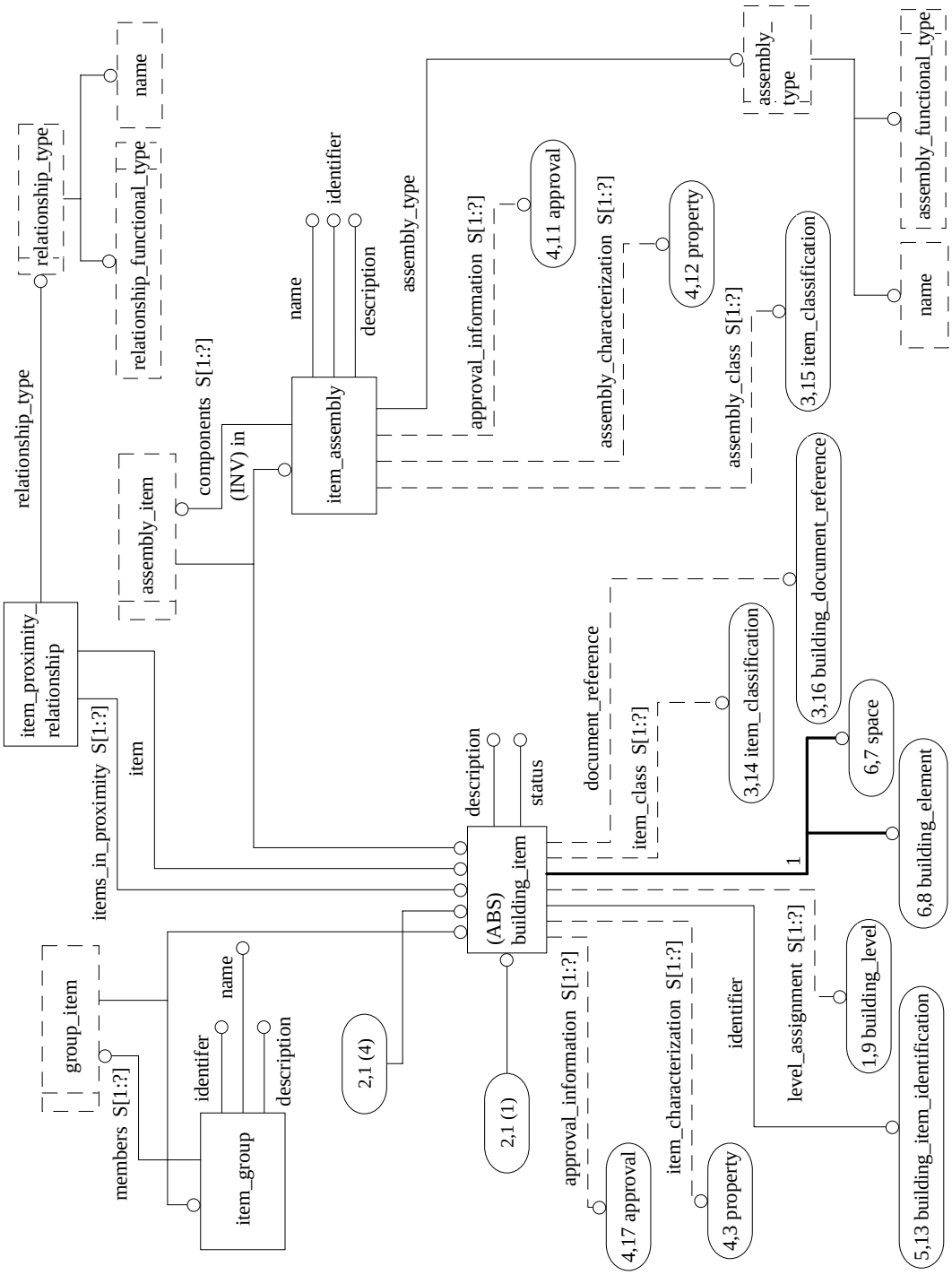


Figure G.2 - ARM diagram 2 of 7 in EXPRESS-G

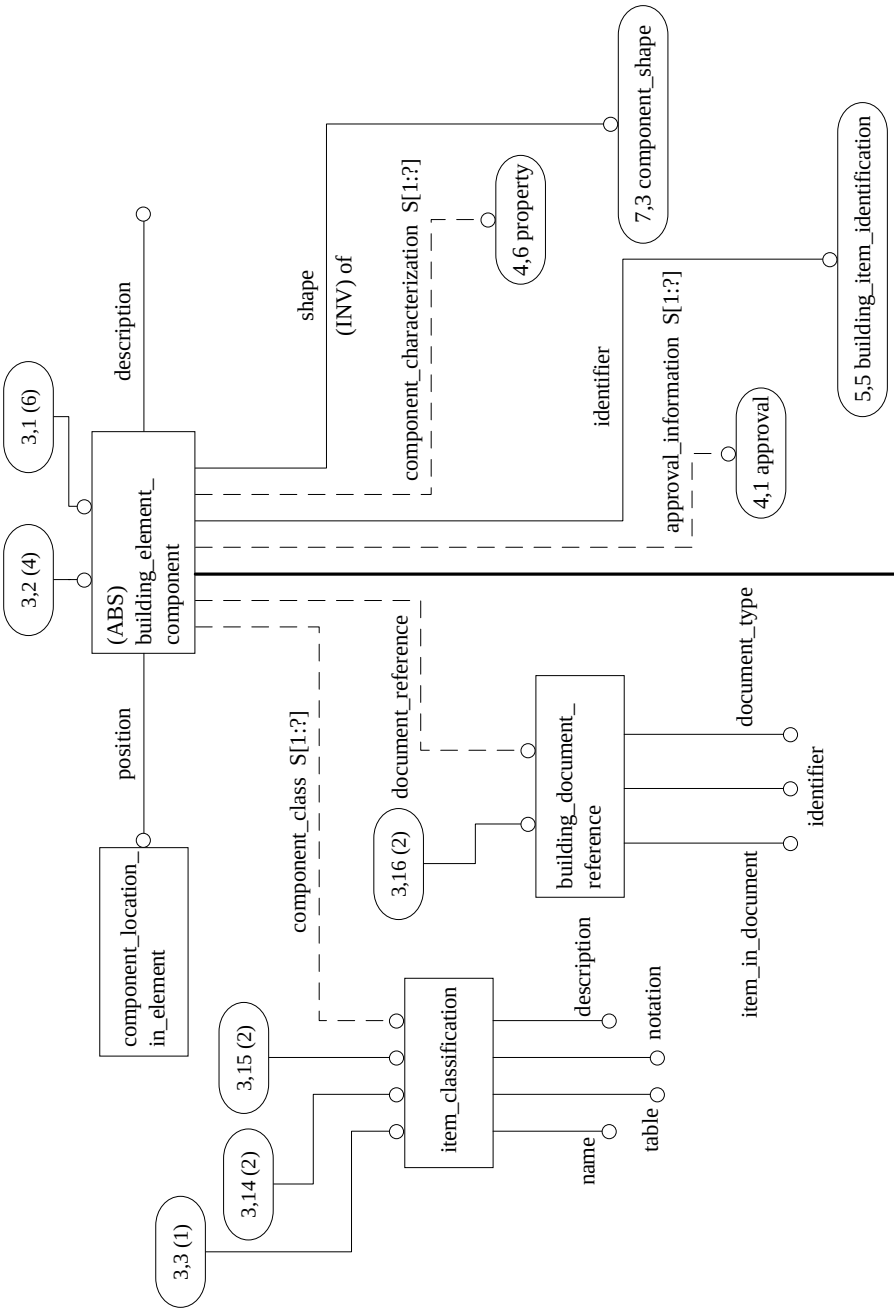


Figure G.3 - ARM diagram 3 of 7 in EXPRESS-G

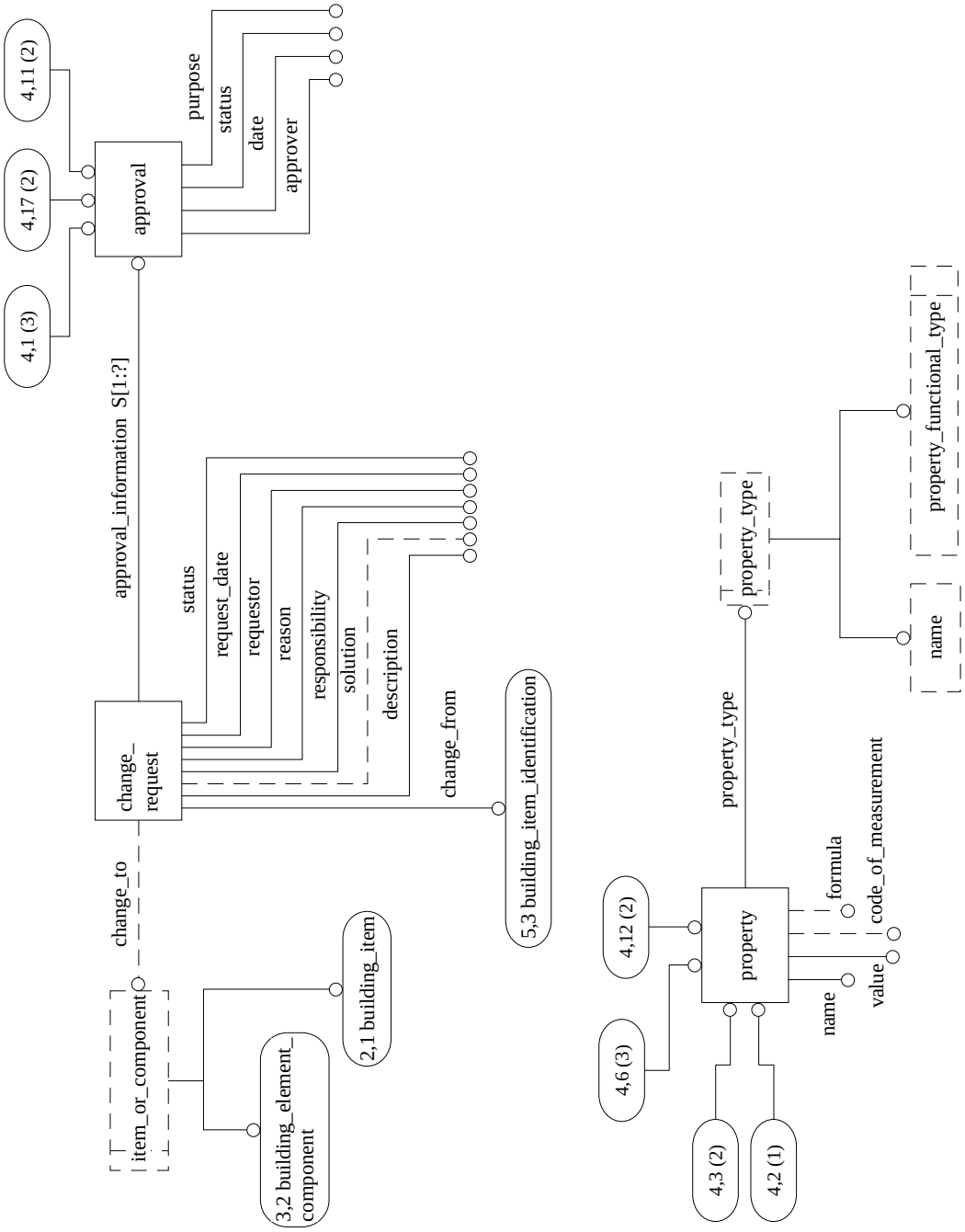


Figure G.4 - ARM diagram 4 of 7 in EXPRESS-G

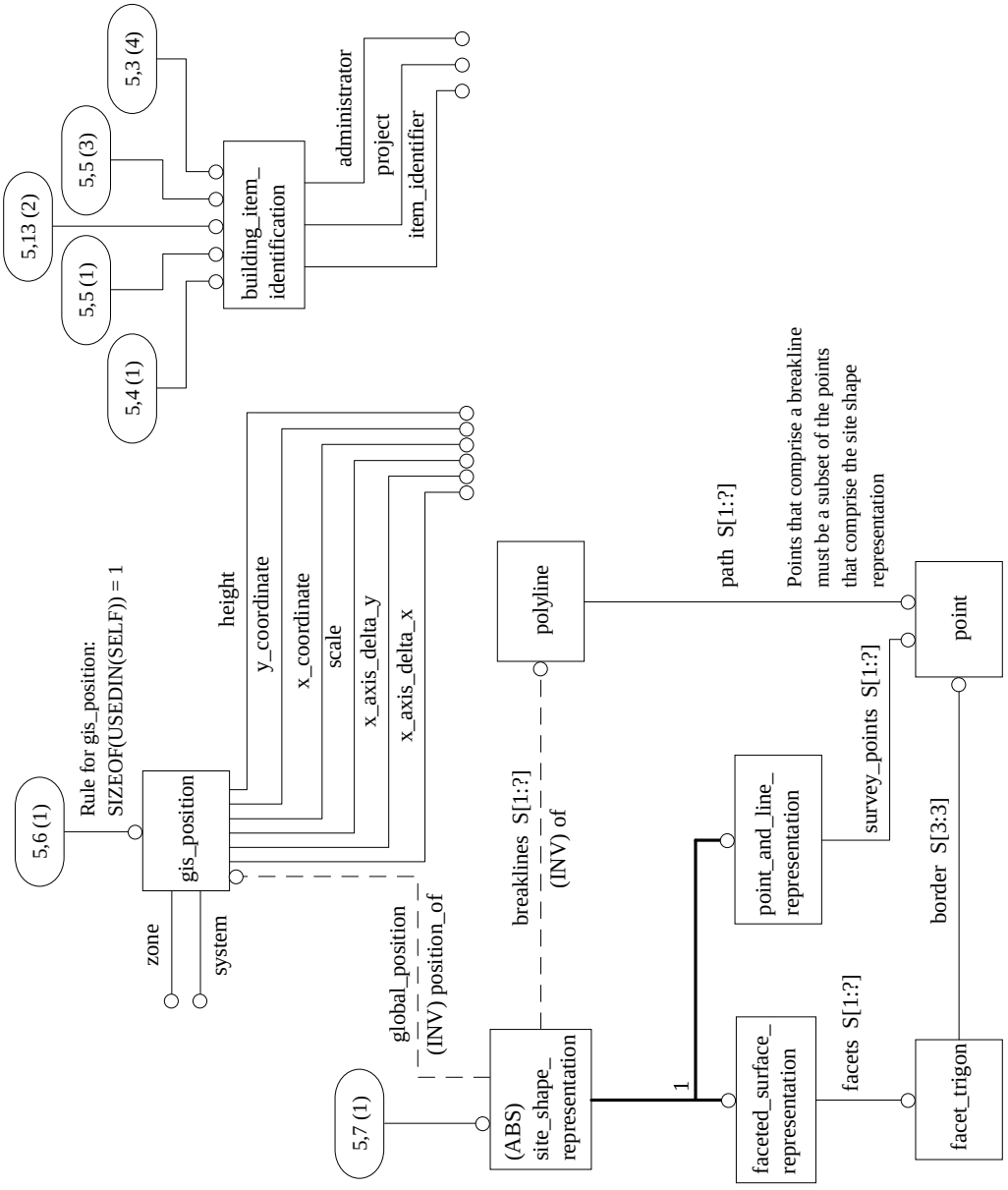


Figure G.5 - ARM diagram 5 of 7 in EXPRESS-G

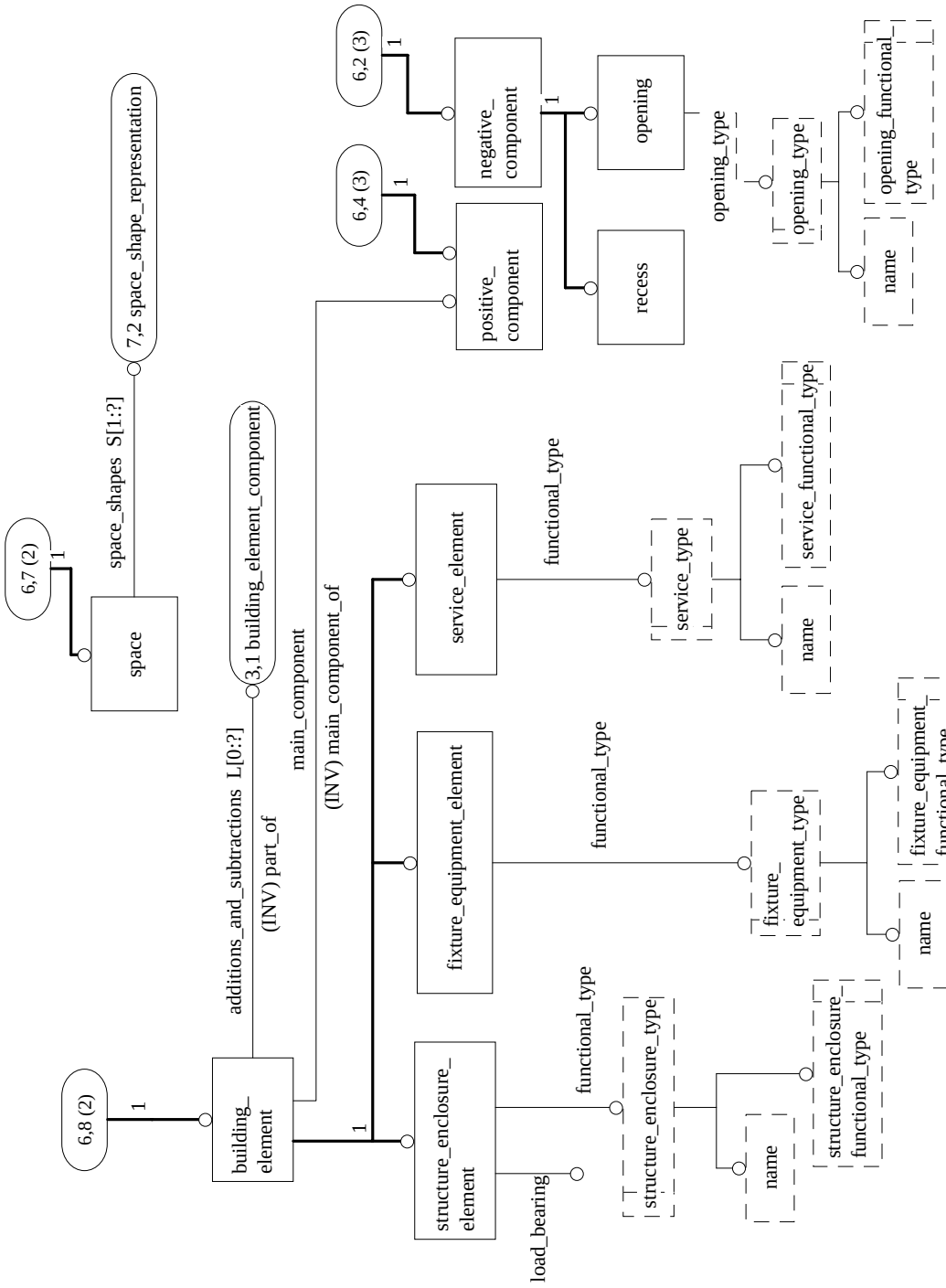


Figure G.6 - ARM diagram 6 of 7 in EXPRESS-G

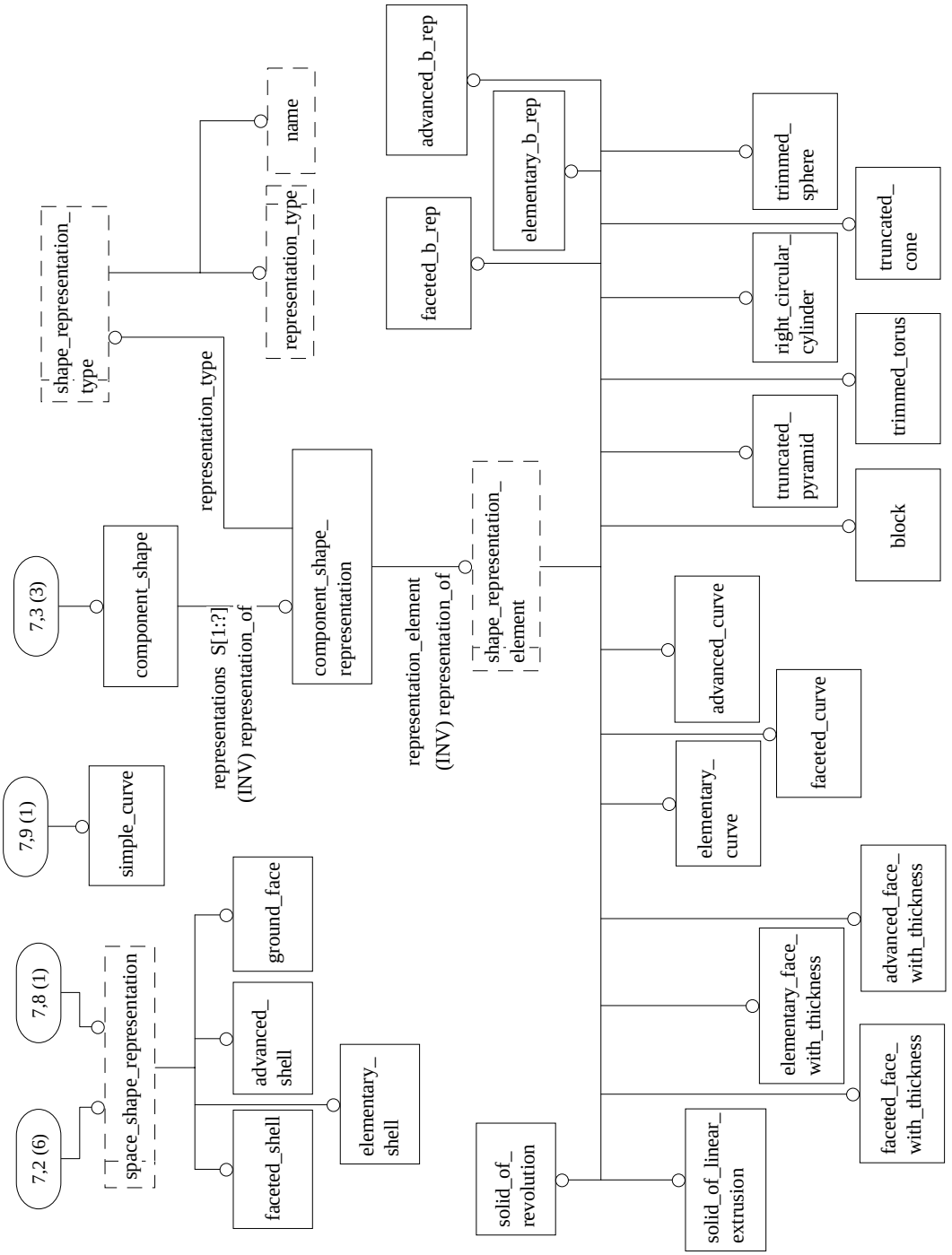


Figure G.7 - ARM diagram 7 of 7 in EXPRESS-G

Annex H
(informative)

AIM EXPRESS-G

Figures H.1 through H.28 correspond to the AIM EXPRESS expanded listing given in annex A. The figures use the EXPRESS-G graphical notation for the EXPRESS language. EXPRESS-G is defined in annex D of ISO 10303-11.

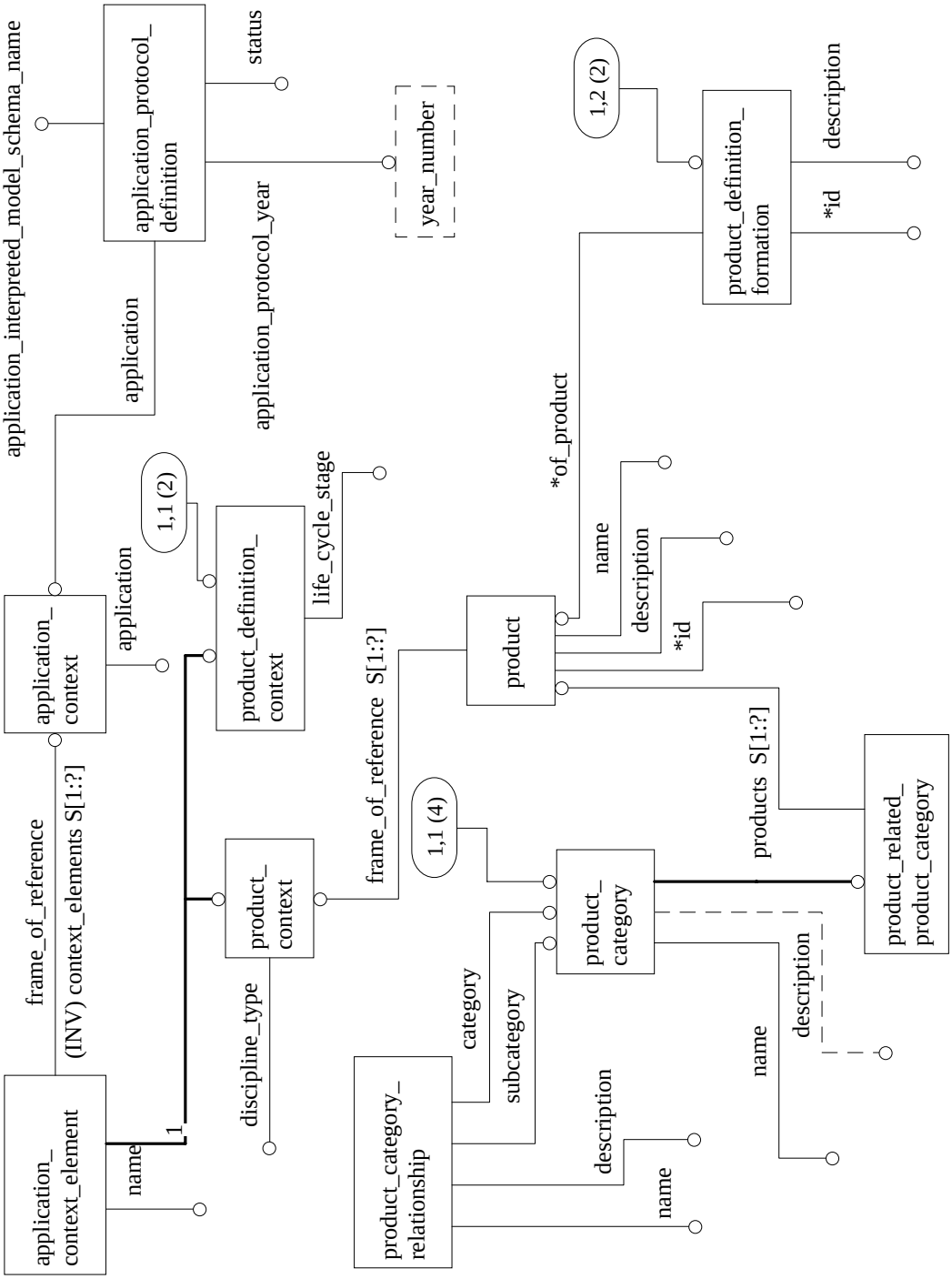


Figure H.1 - AIM diagram 1 of 29 in EXPRESS-G

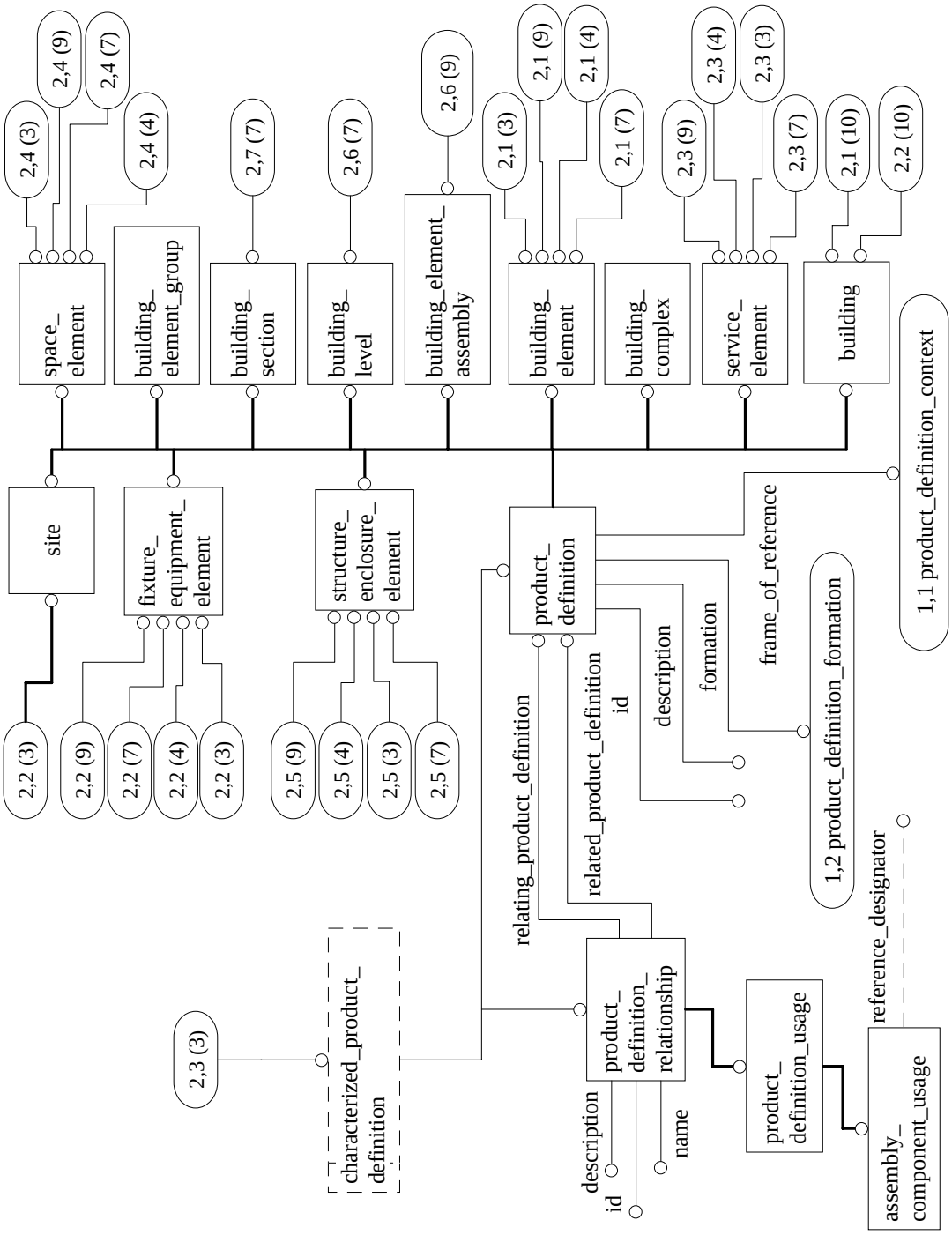


Figure H.2 - AIM diagram 2 of 29 in EXPRESS-G

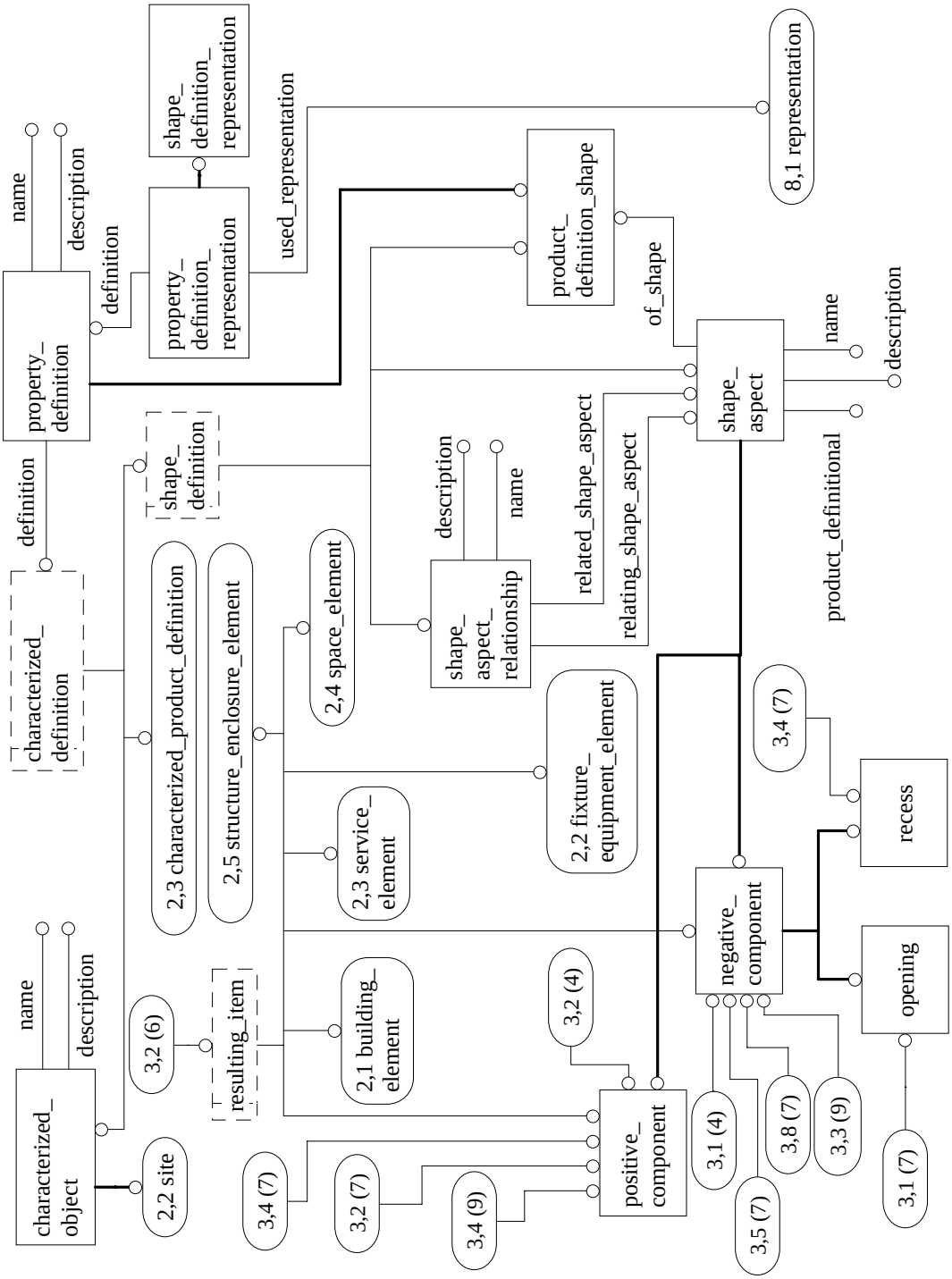


Figure H.3 - AIM diagram 3 of 29 in EXPRESS-G

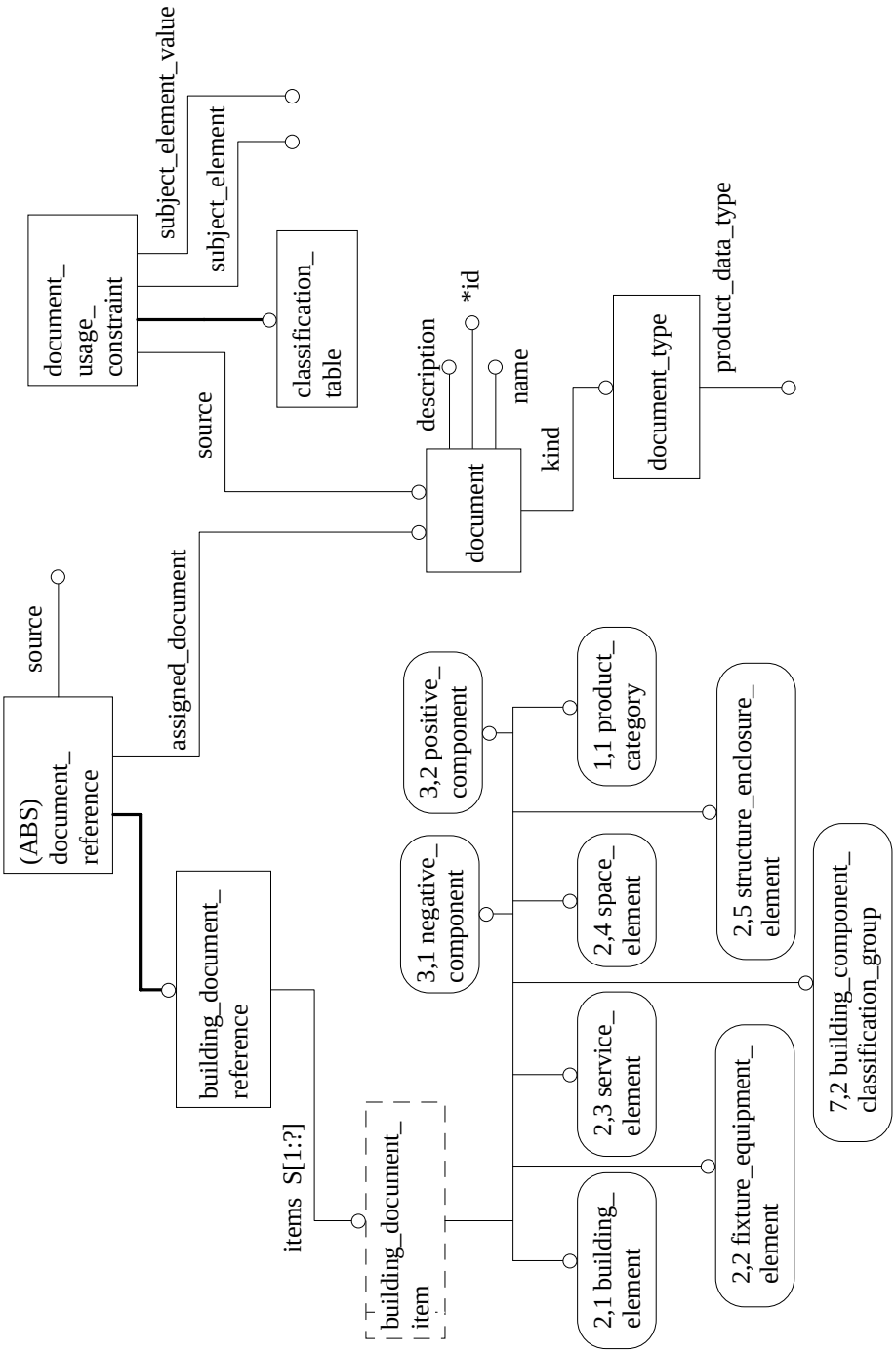


Figure H.4 - AIM diagram 4 of 29 in EXPRESS-G

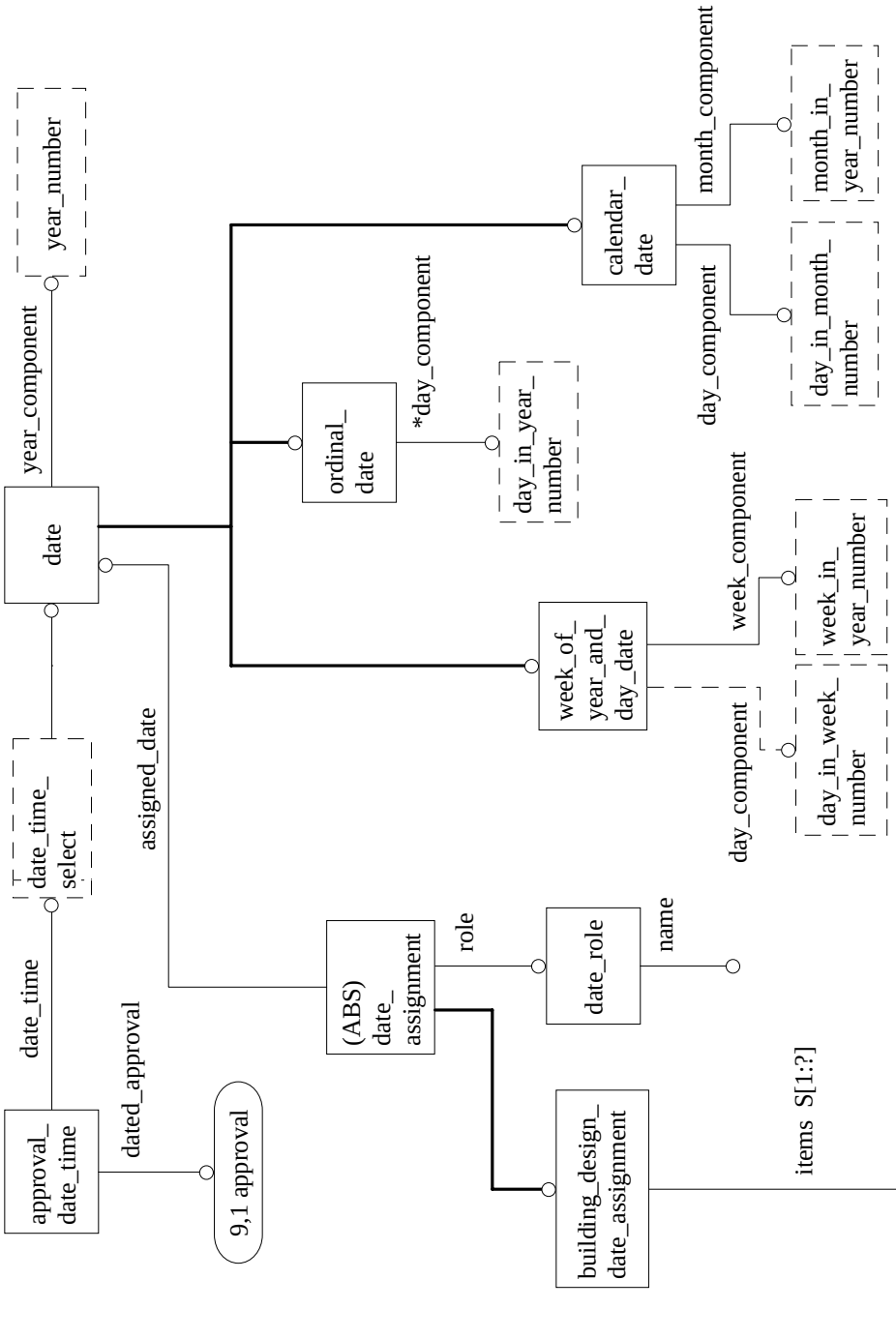


Figure H.5 - AIM diagram 5 of 29 in EXPRESS-G

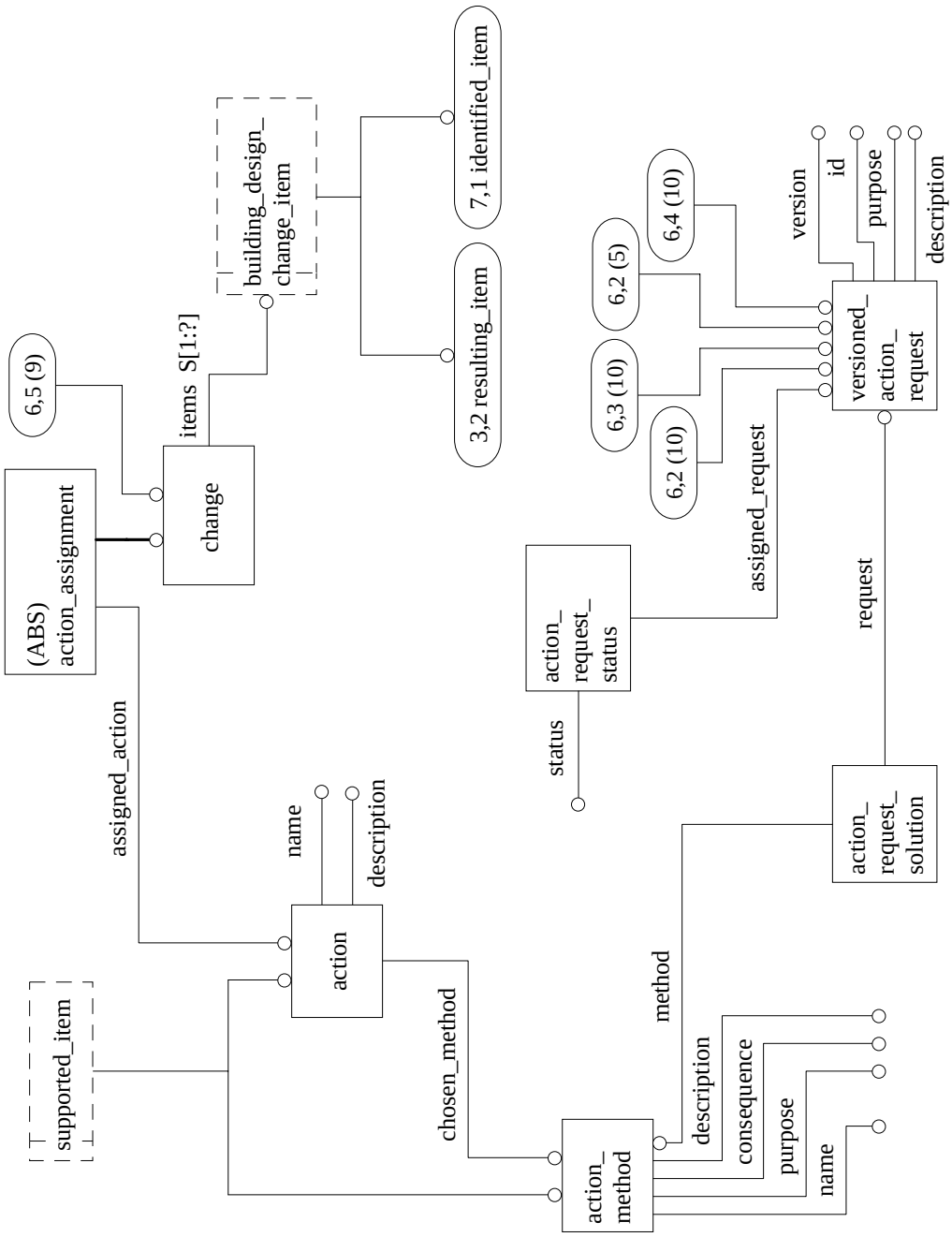


Figure H.6 - AIM diagram 6 of 29 in EXPRESS-G

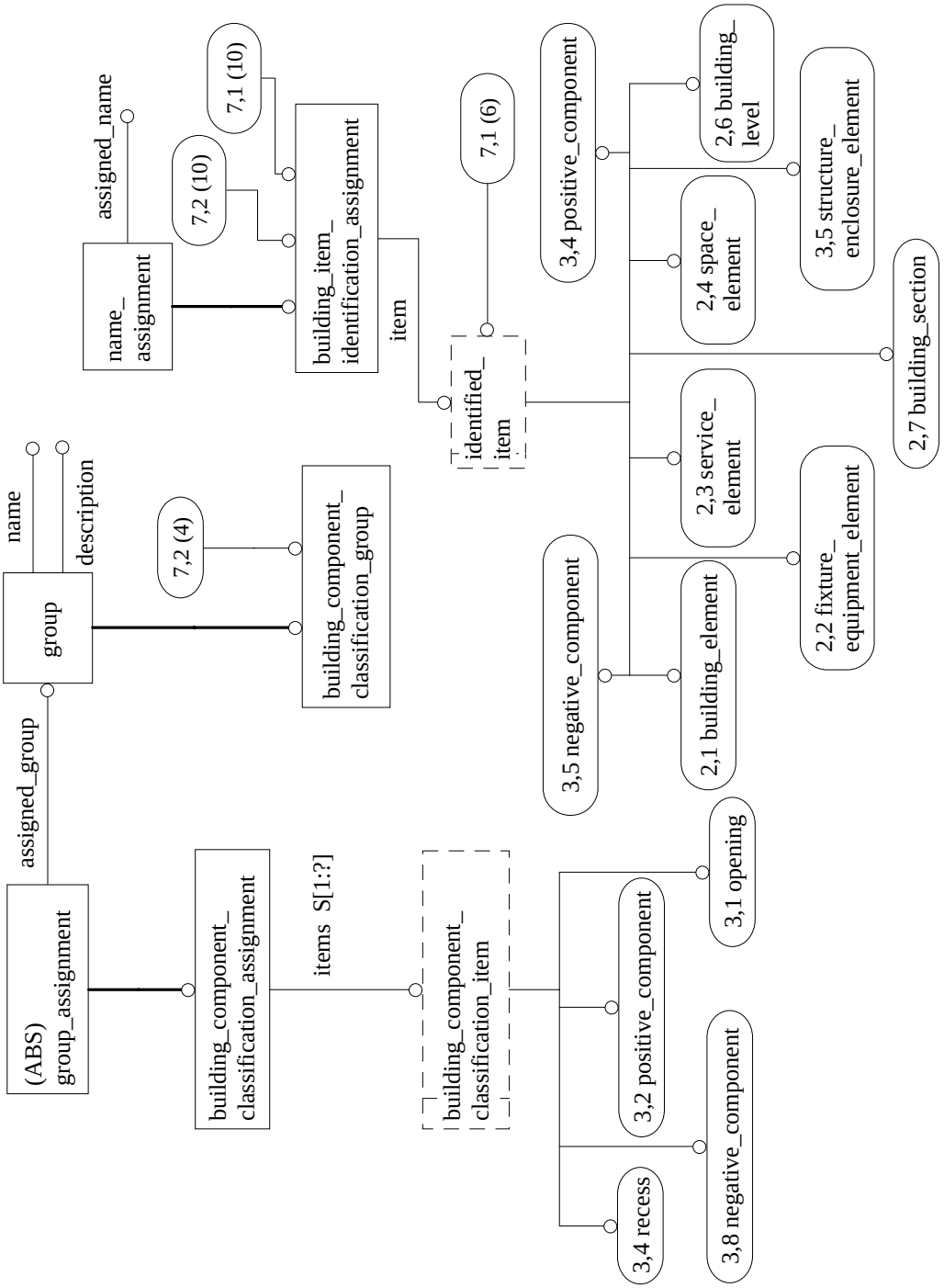


Figure H.7 - AIM diagram 7 of 29 in EXPRESS-G

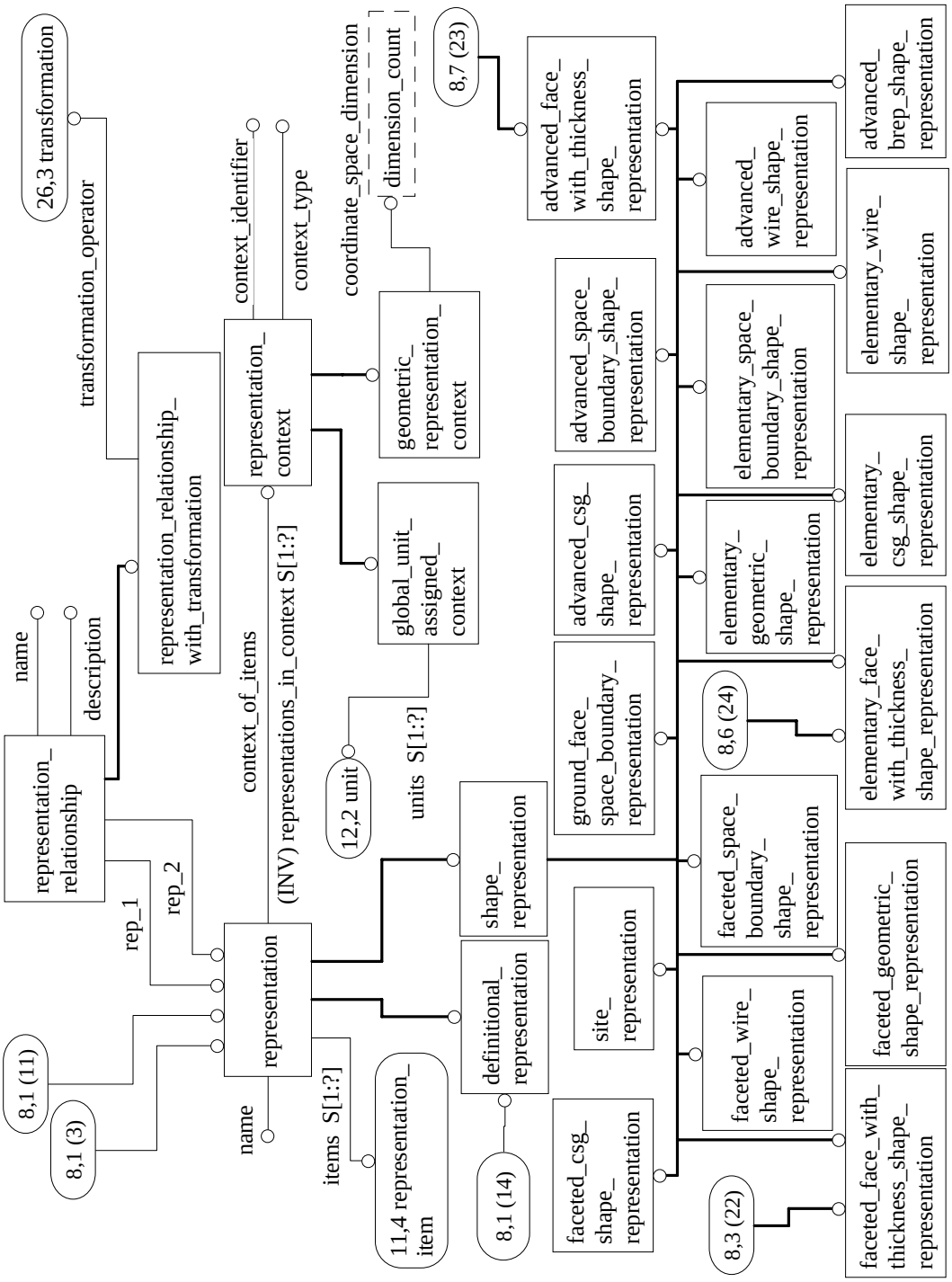


Figure H.8 - AIM diagram 8 of 29 in EXPRESS-G

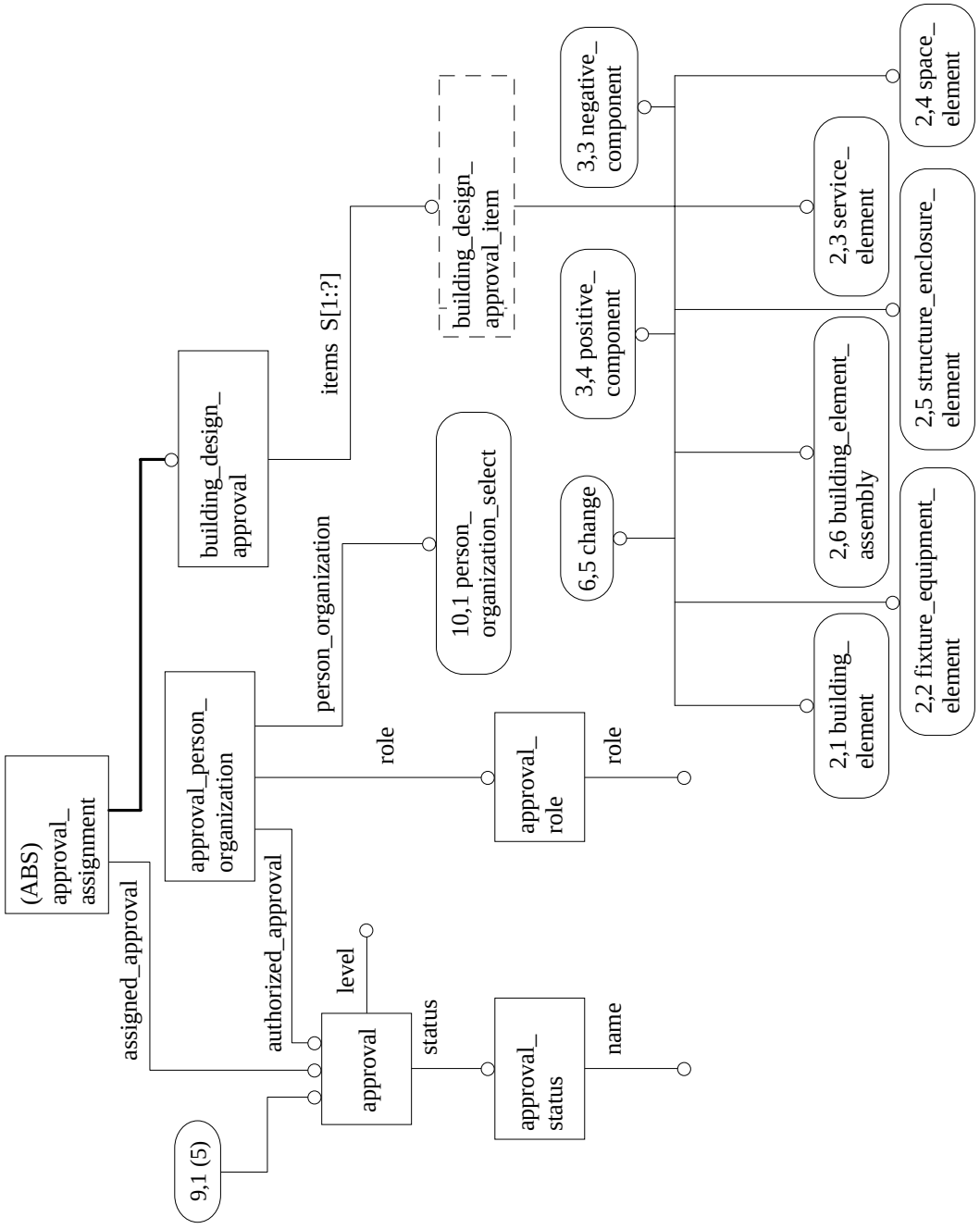


Figure H.9 - AIM diagram 9 of 29 in EXPRESS-G

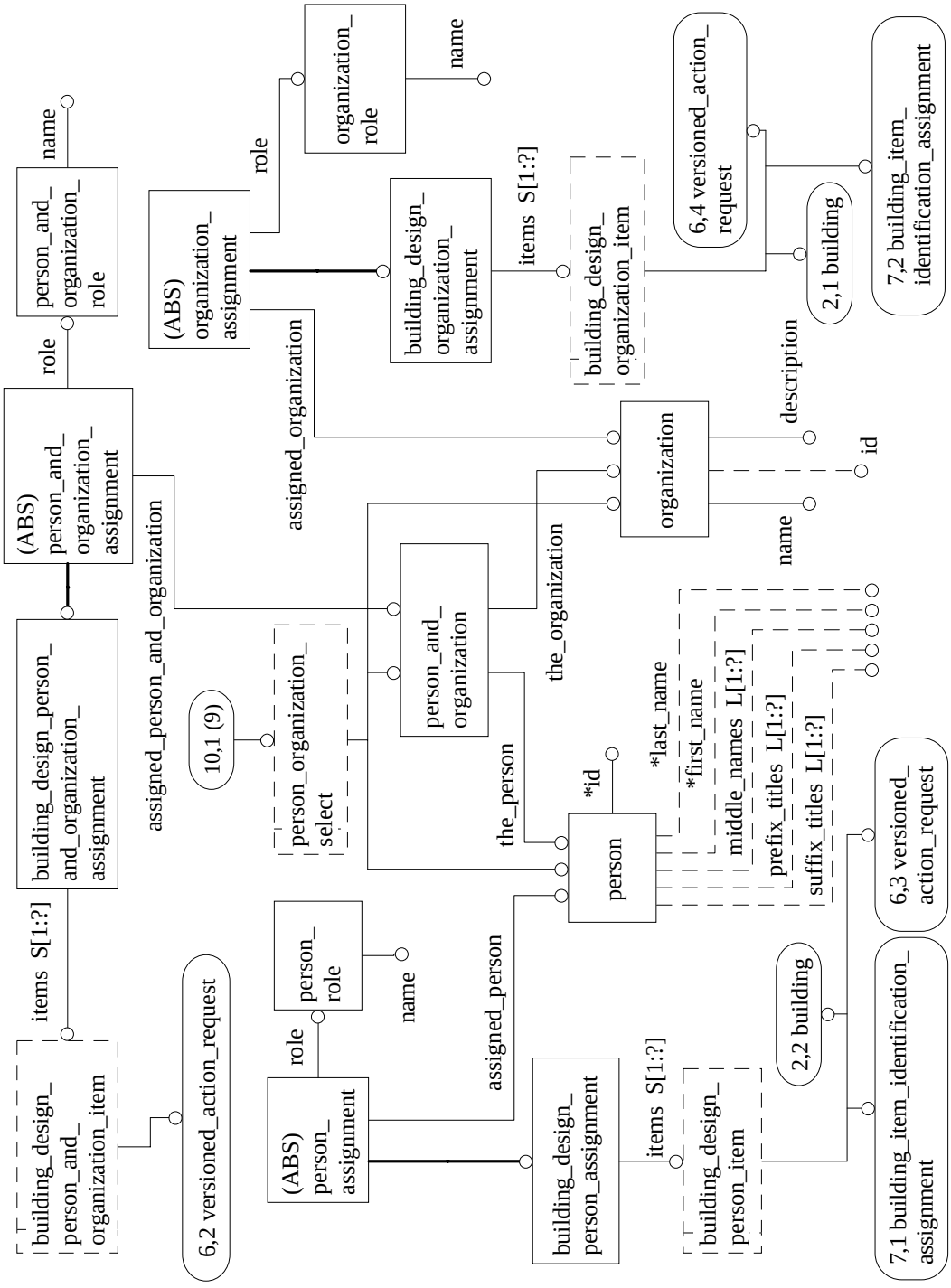


Figure H.10 - AIM diagram 10 of 29 in EXPRESS-G

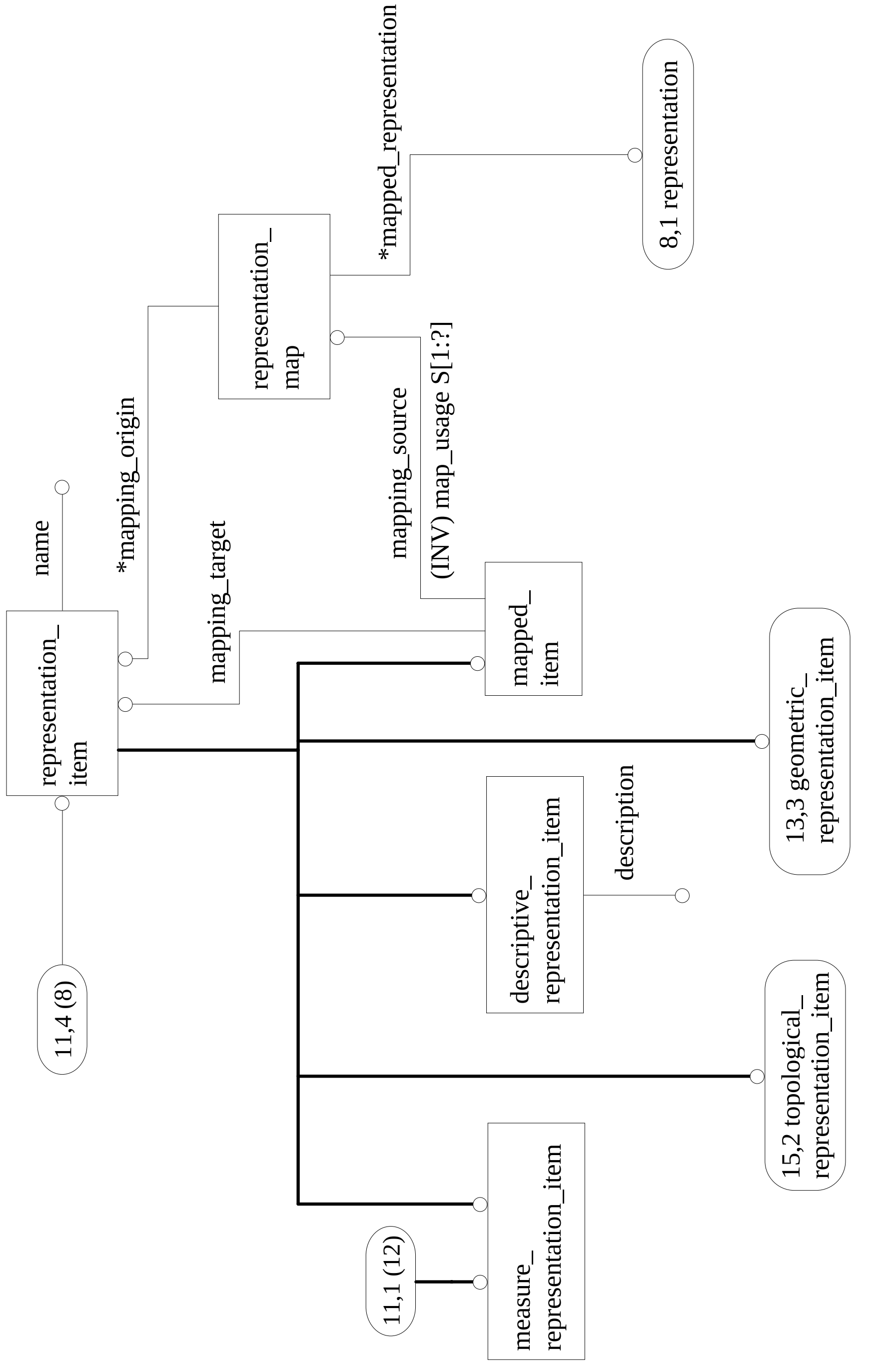


Figure H.11 - AIM diagram 11 of 29 in EXPRESS-G

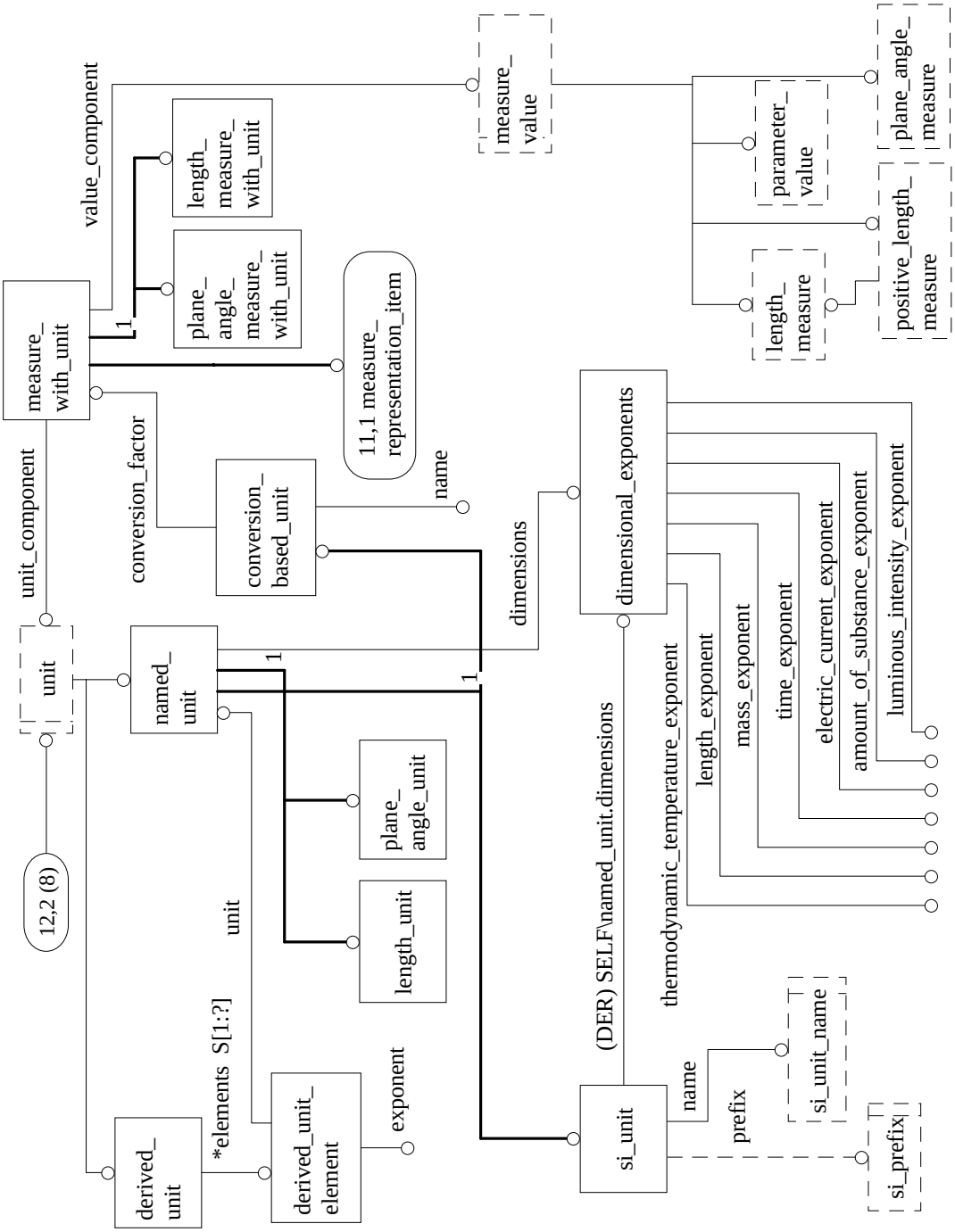


Figure H.12 - AIM diagram 12 of 29 in EXPRESS-G

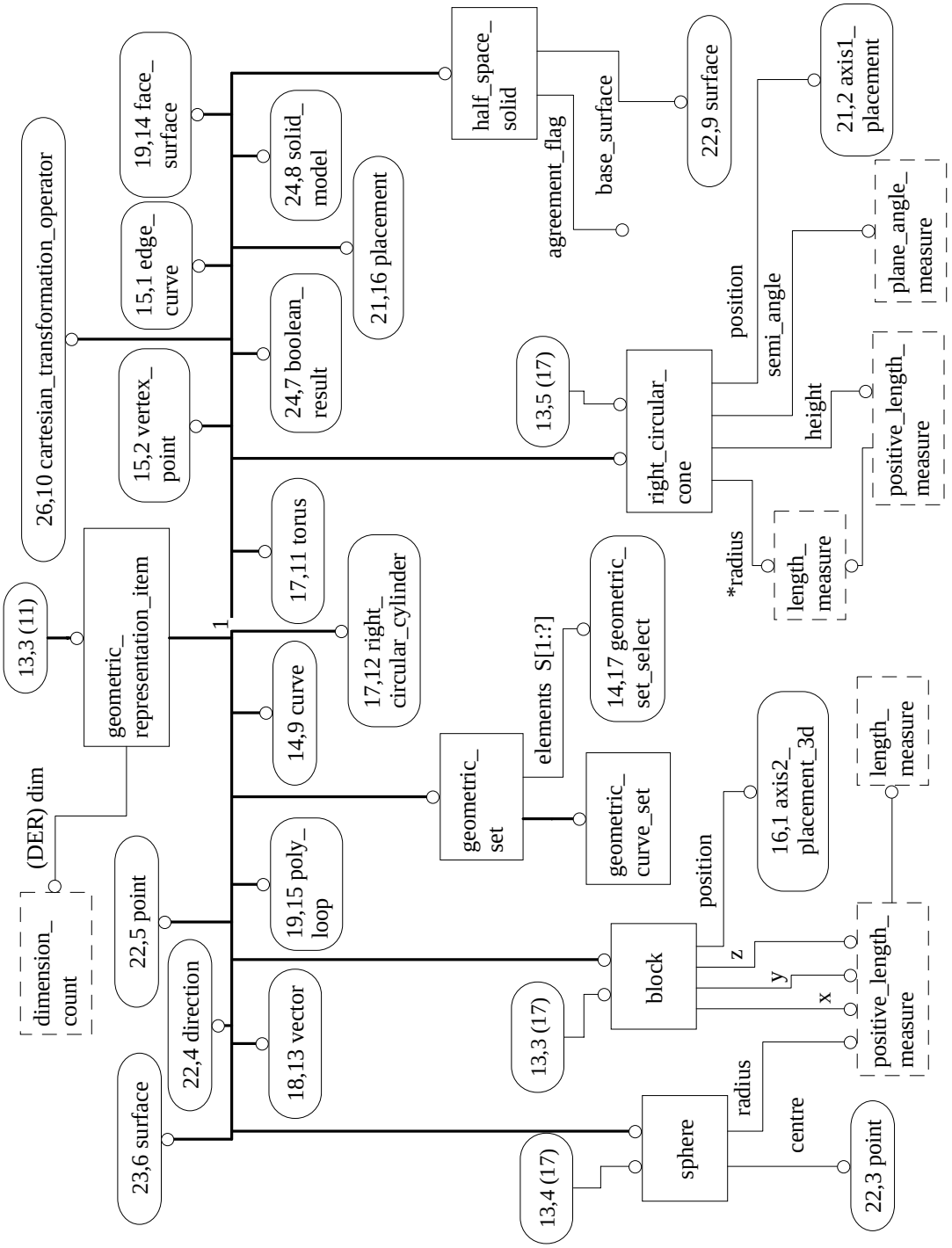


Figure H.13 - AIM diagram 13 of 29 in EXPRESS-G

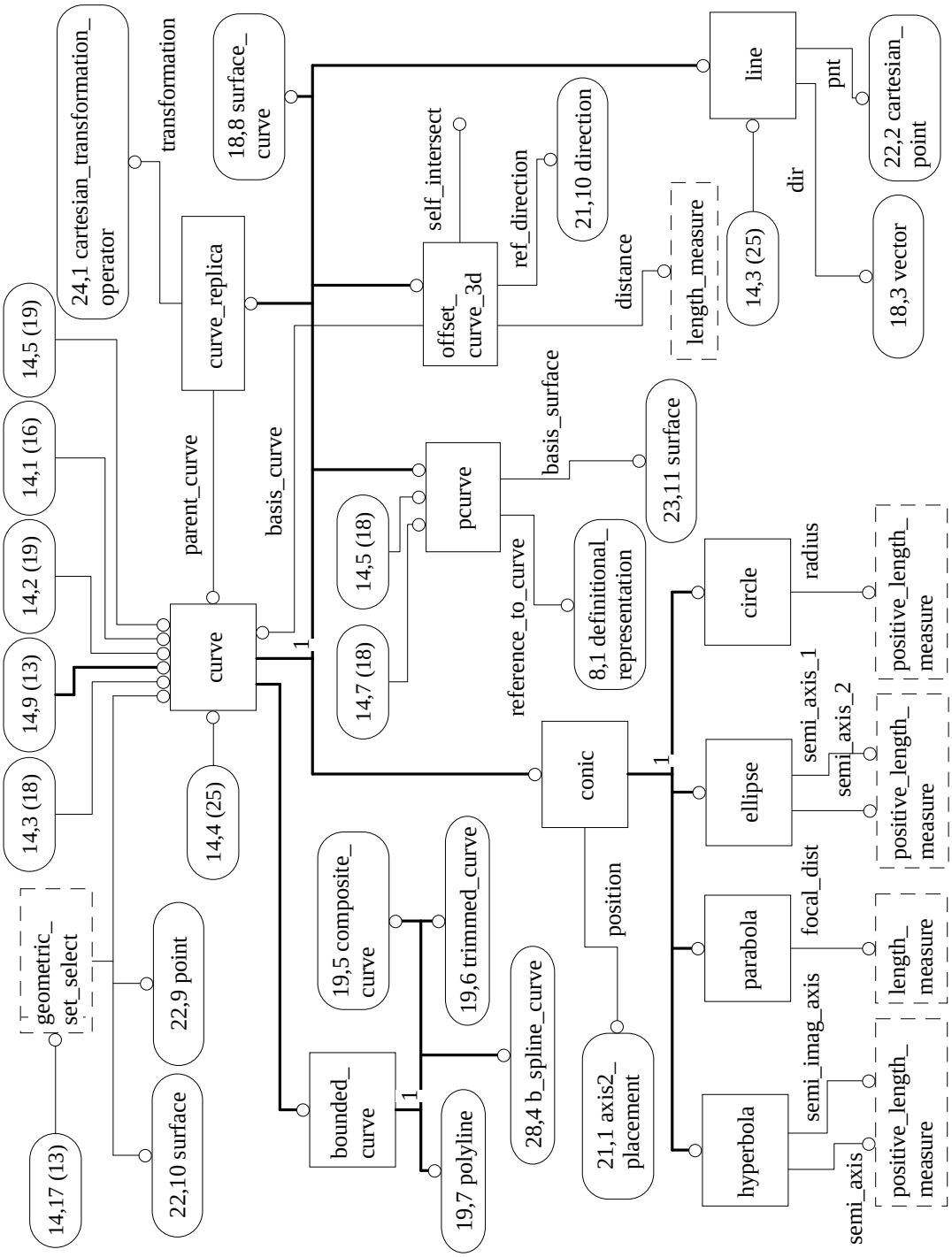


Figure H.14 - AIM diagram 14 of 29 in EXPRESS-G

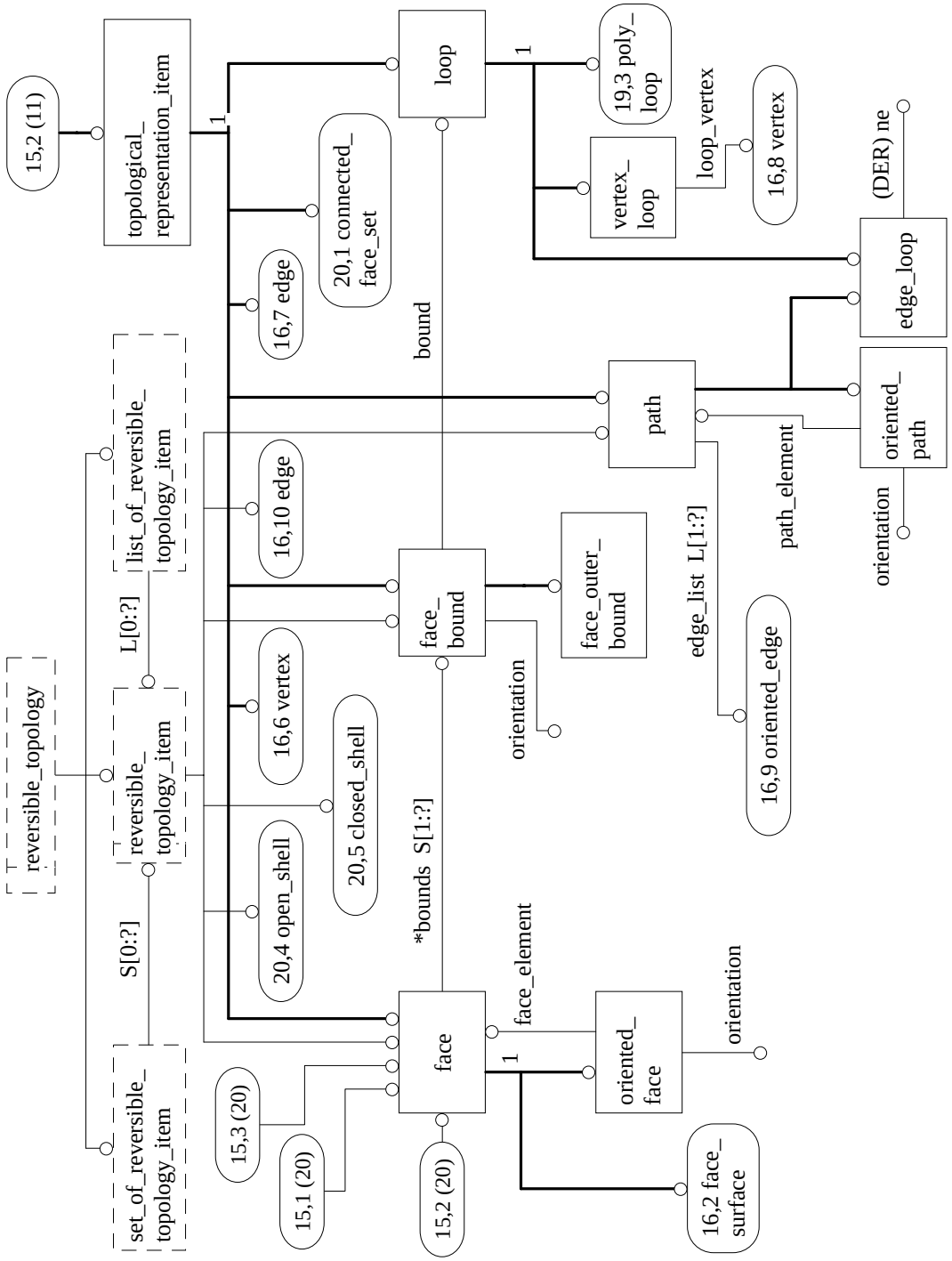


Figure H.15 - AIM diagram 15 of 29 in EXPRESS-G

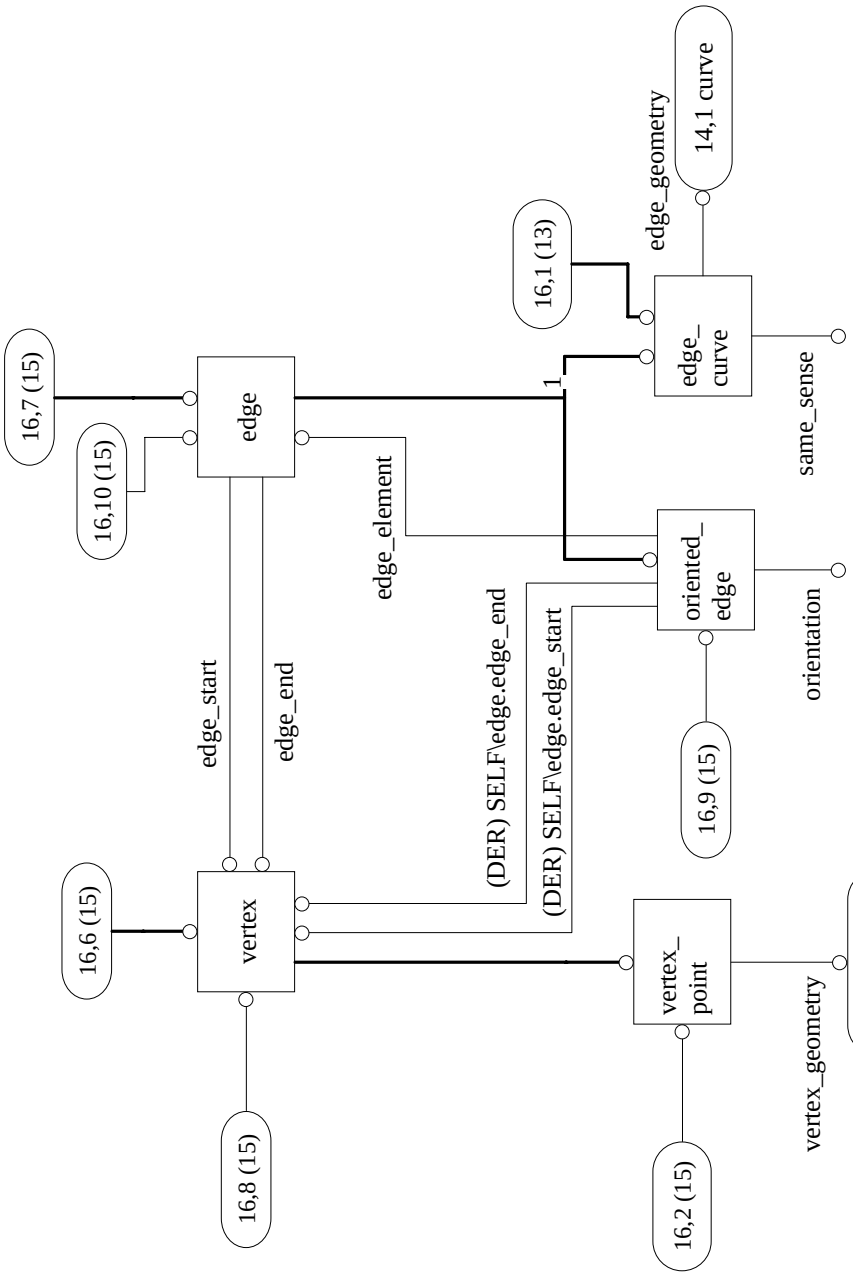


Figure H.16 - AIM diagram 16 of 29 in EXPRESS-G

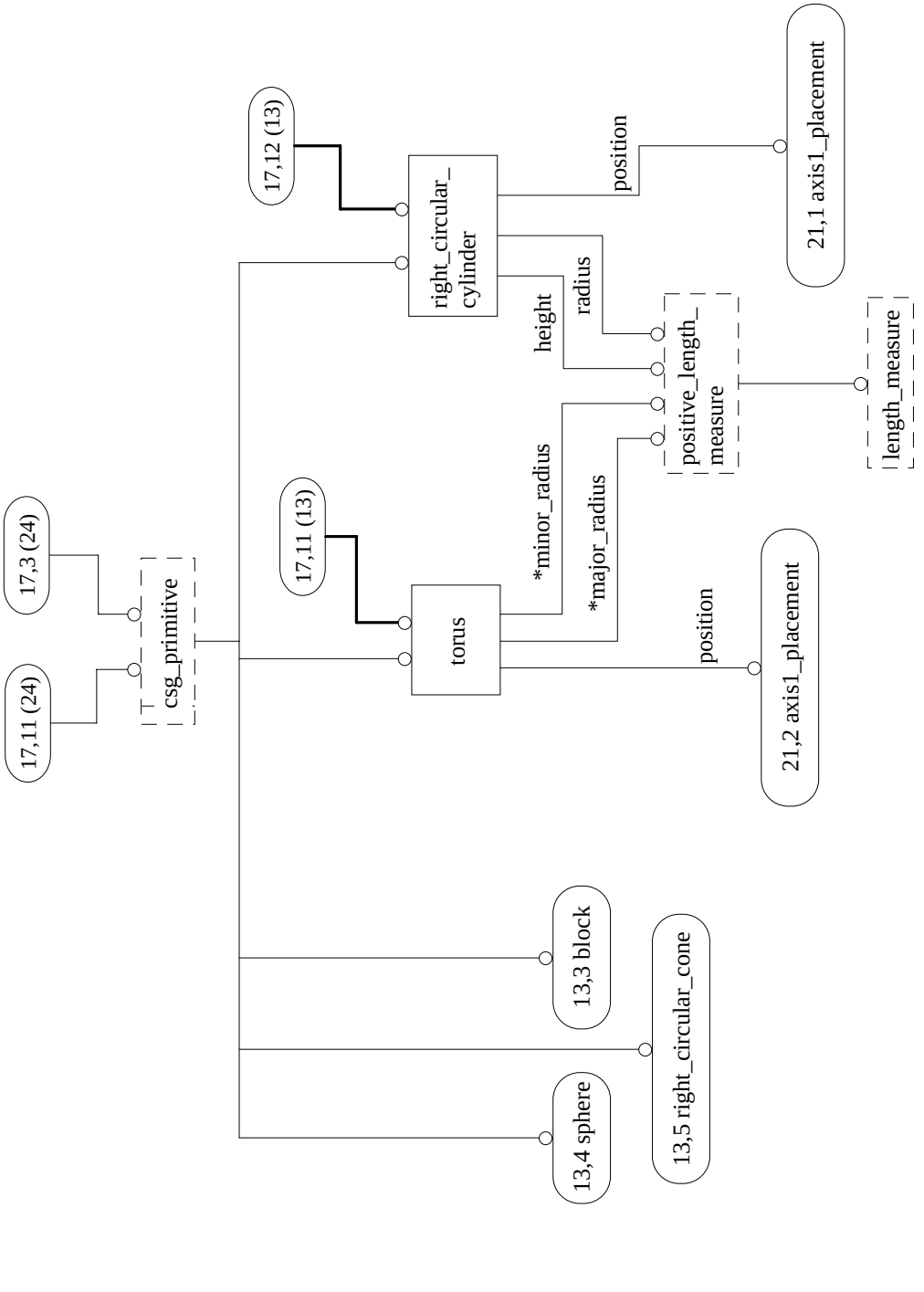


Figure H.17 - AIM diagram 17 of 29 in EXPRESS-G

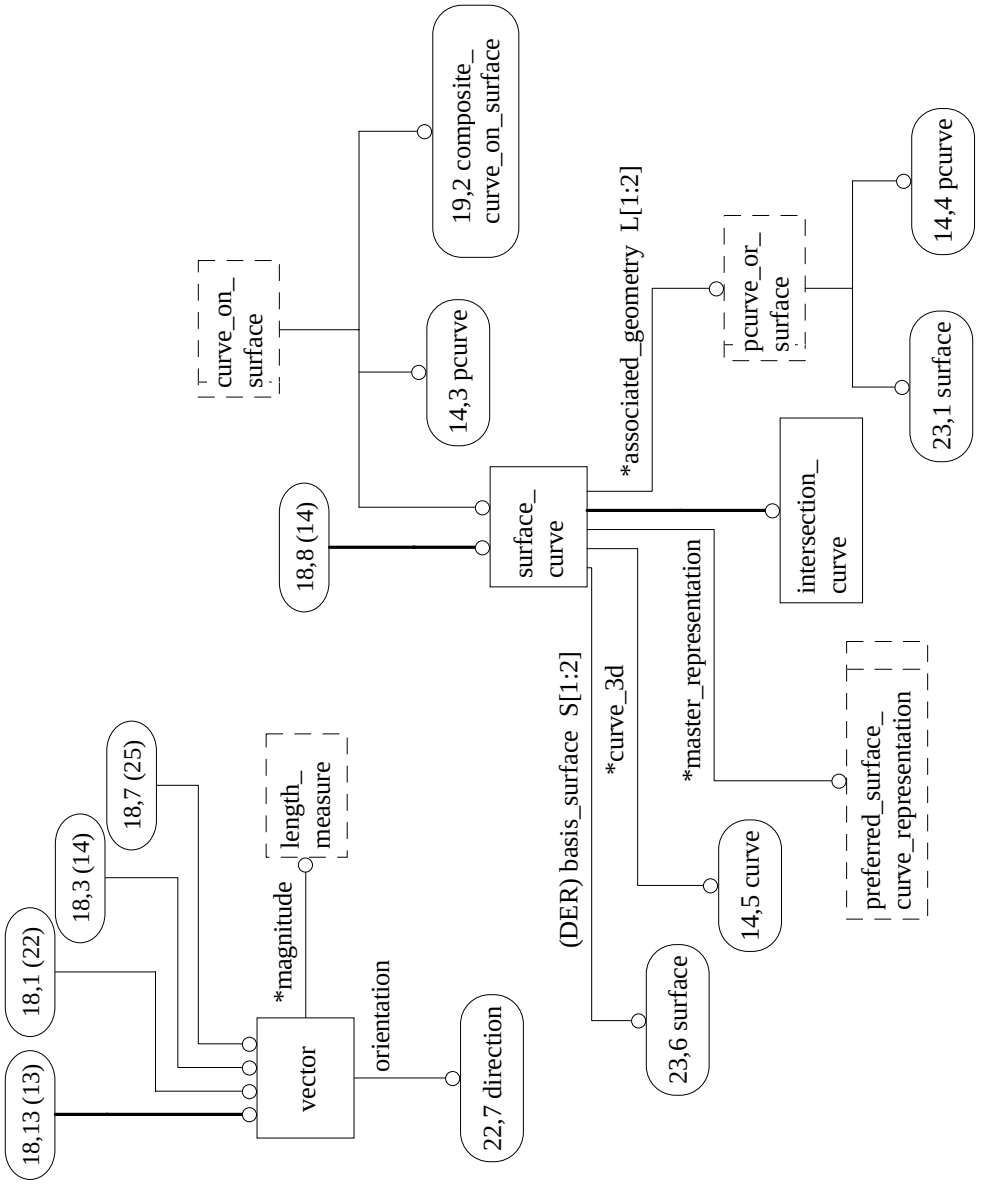


Figure H.18 - AIM diagram 18 of 29 in EXPRESS-G

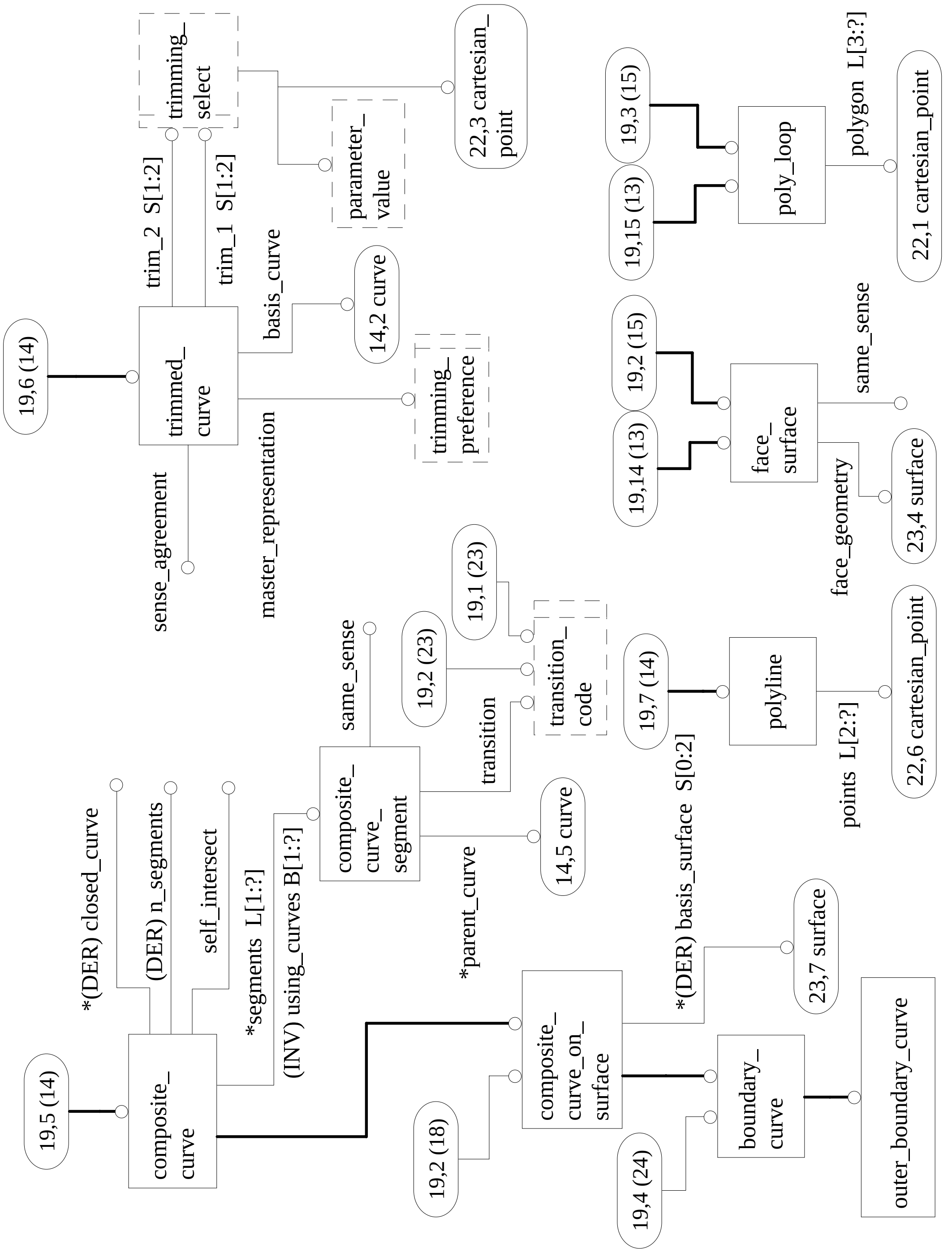


Figure H.19 - AIM diagram 19 of 29 in EXPRESS-G

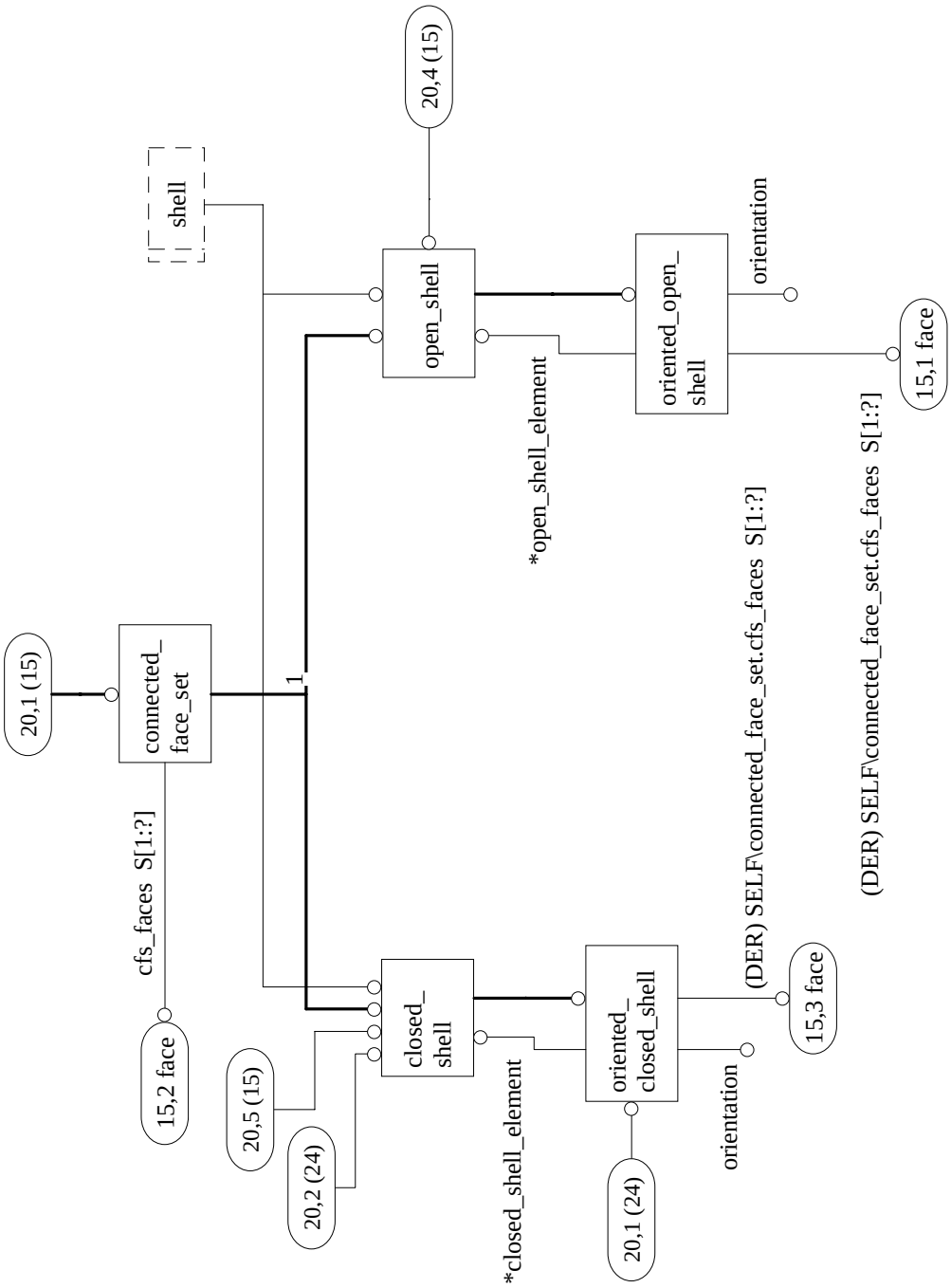


Figure H.20 - AIM diagram 20 of 29 in EXPRESS-G

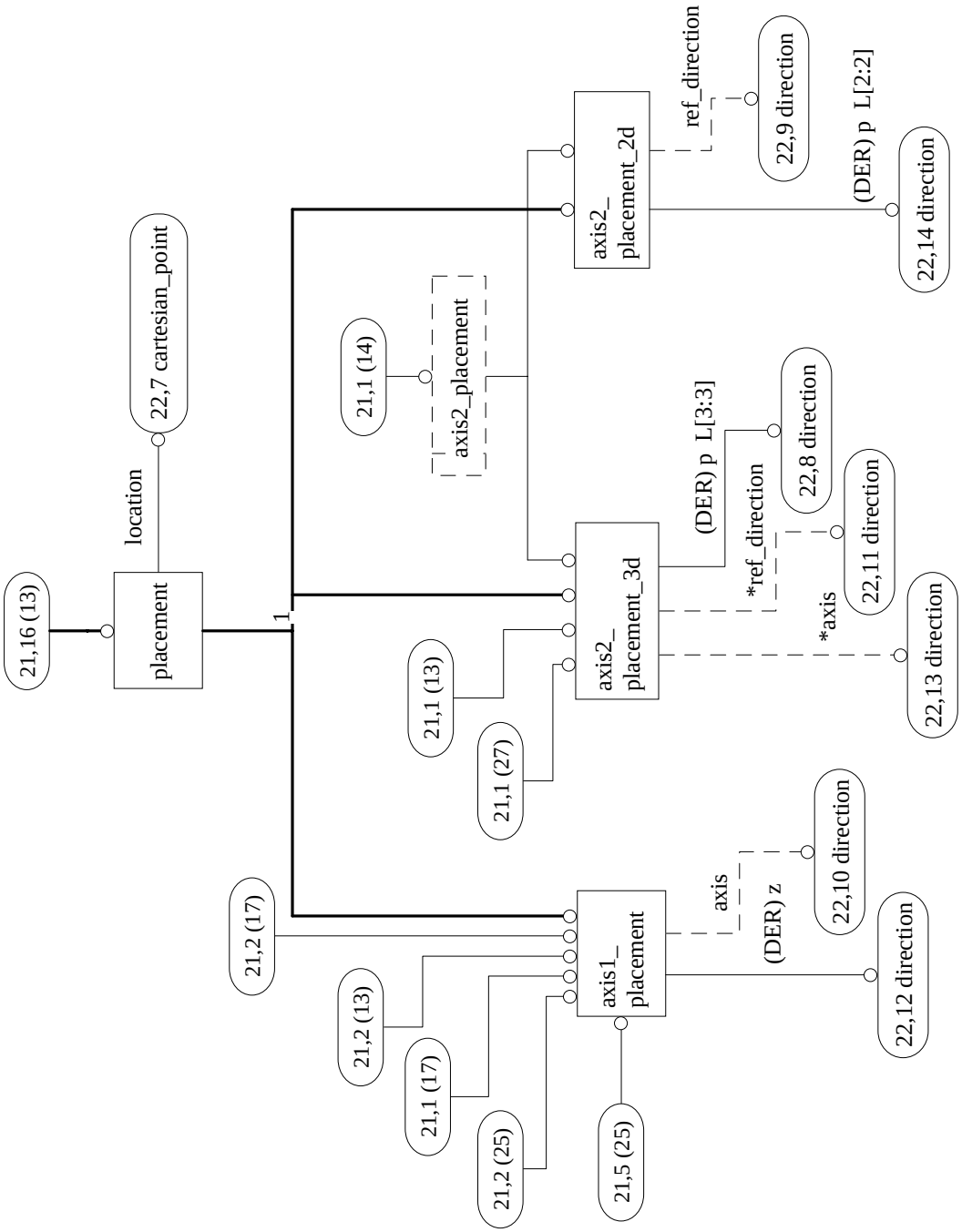


Figure H.21 - AIM diagram 21 of 29 in EXPRESS-G

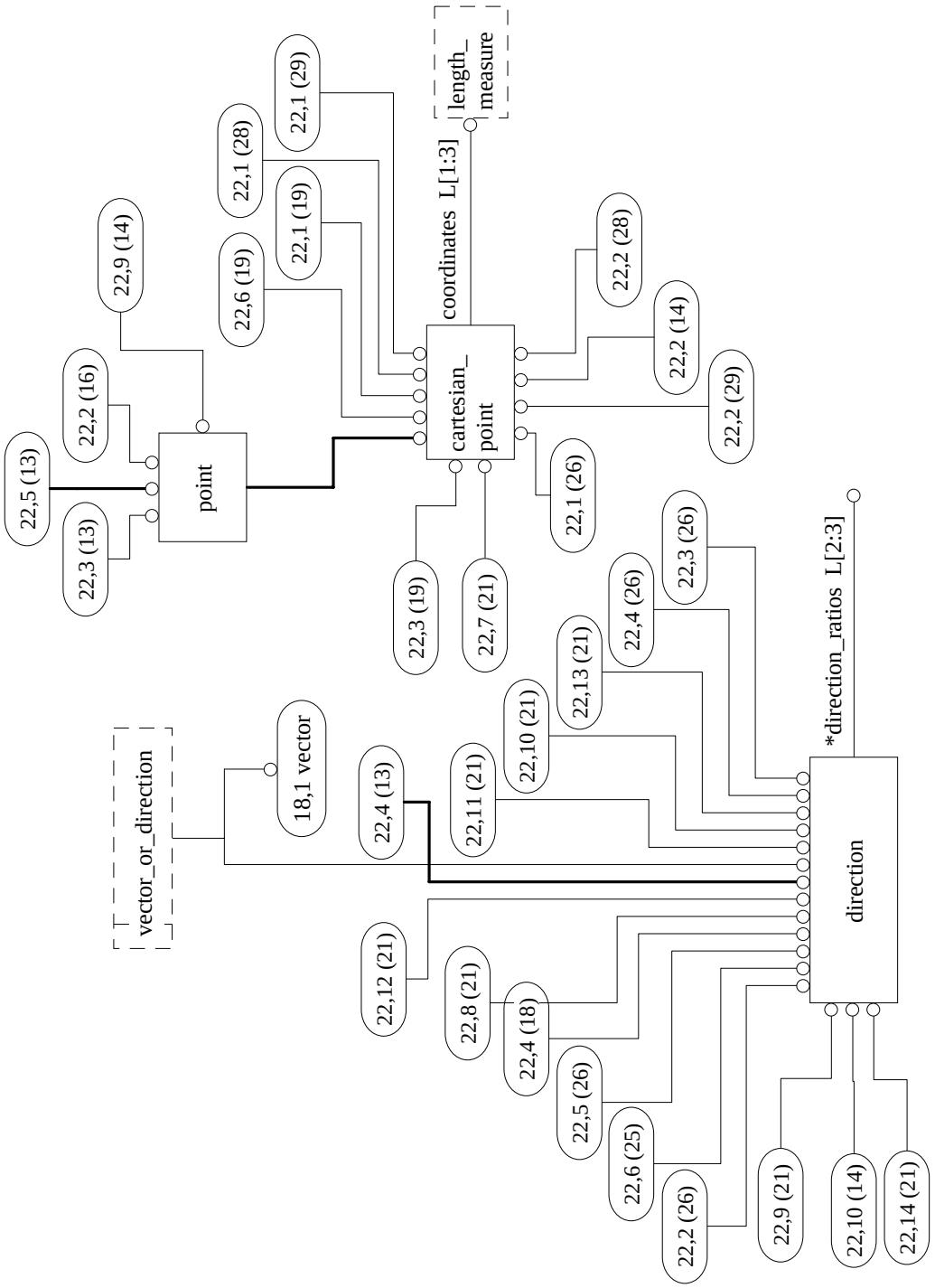


Figure H.22 - AIM diagram 22 of 29 in EXPRESS-G

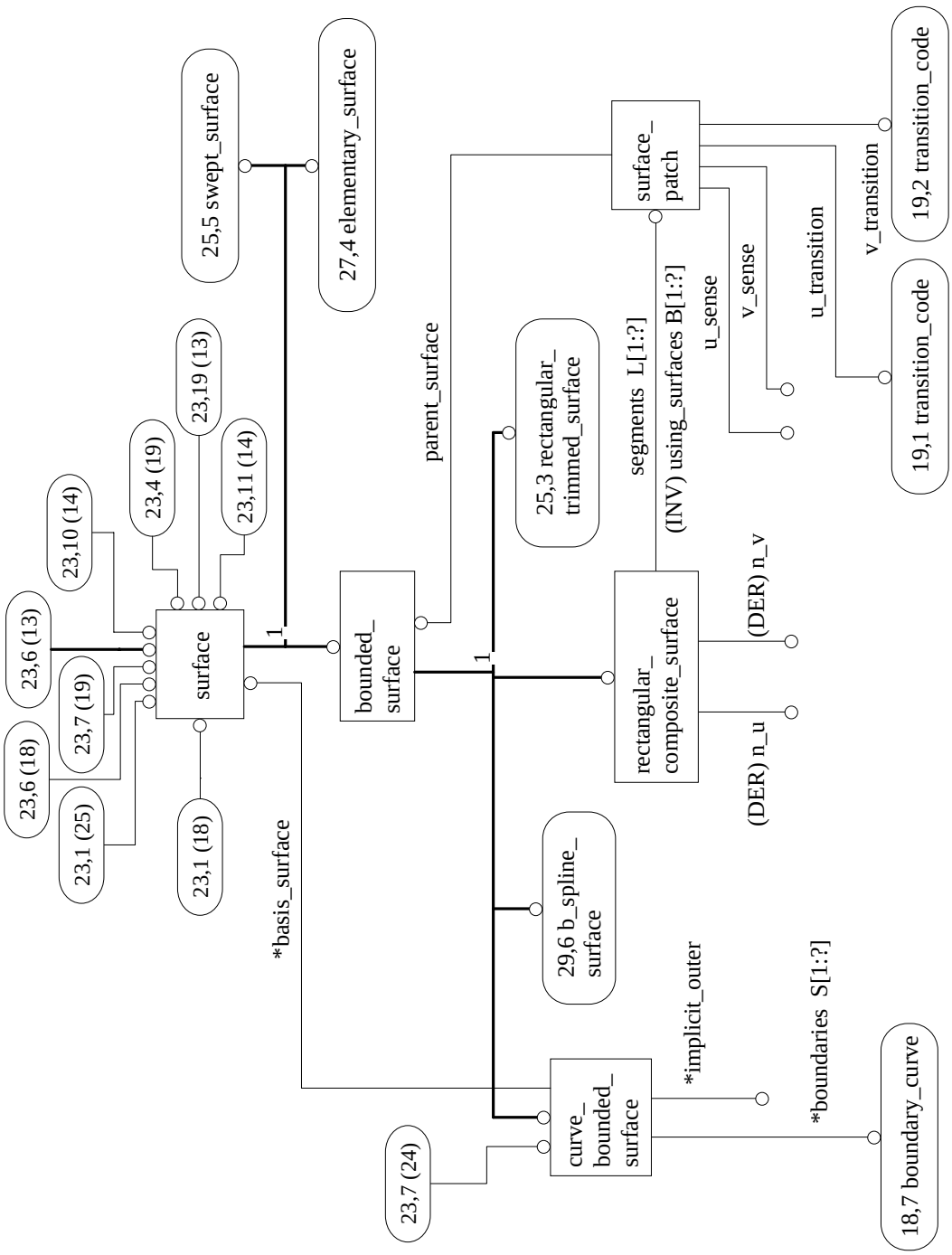


Figure H.23 - AIM diagram 23 of 29 in EXPRESS-G

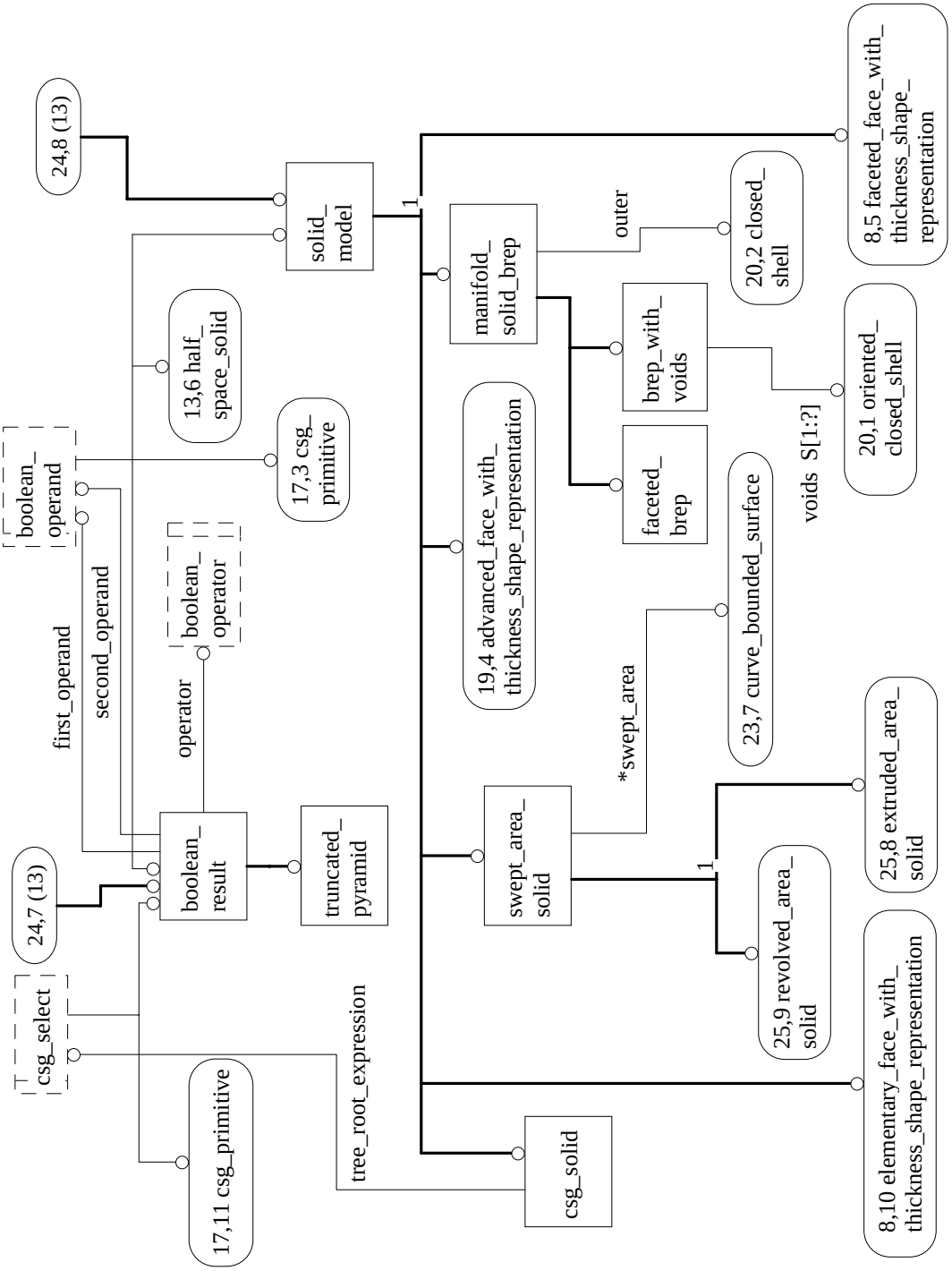


Figure H.24 - AIM diagram 24 of 29 in EXPRESS-G

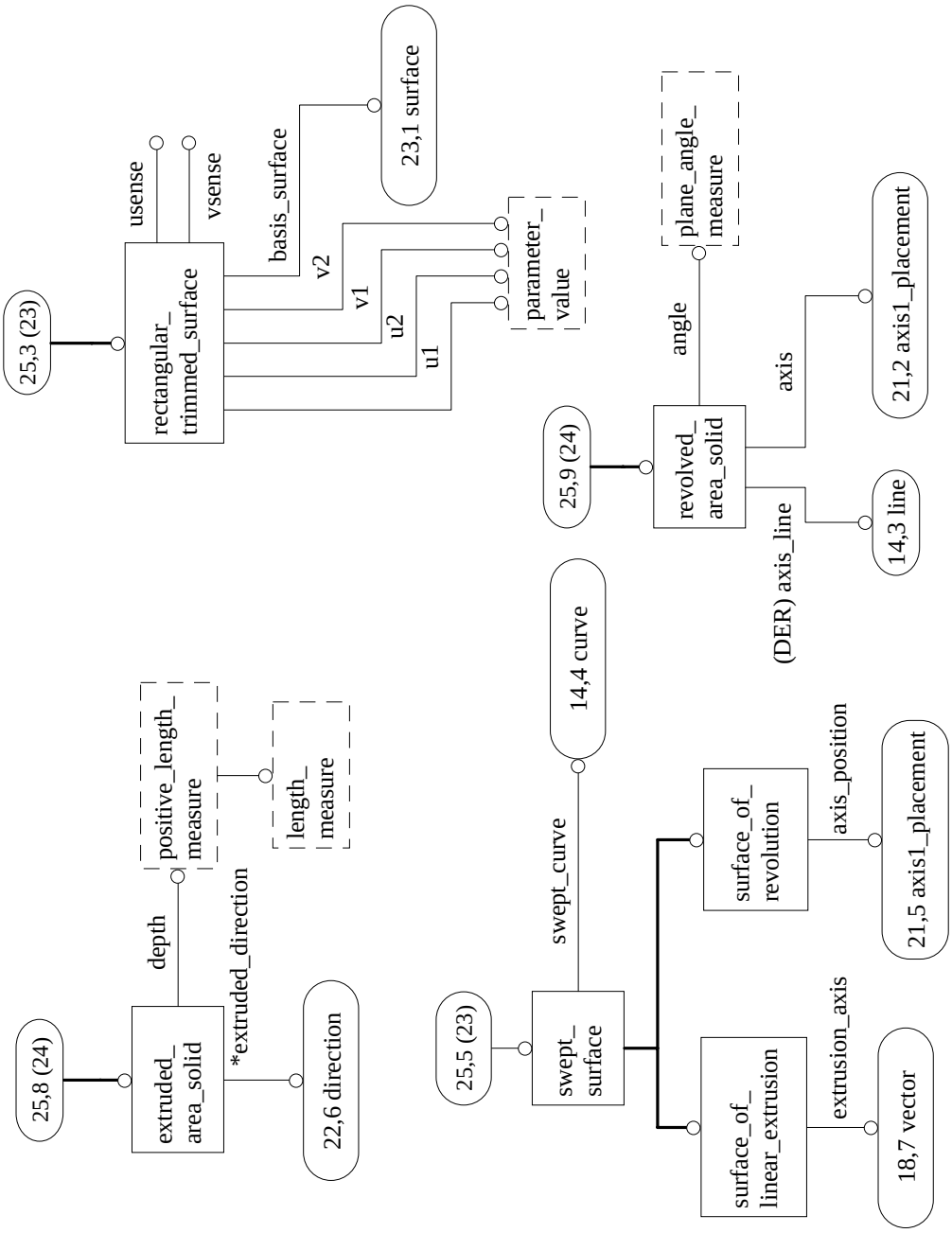


Figure H.25 - AIM diagram 25 of 29 in EXPRESS-G

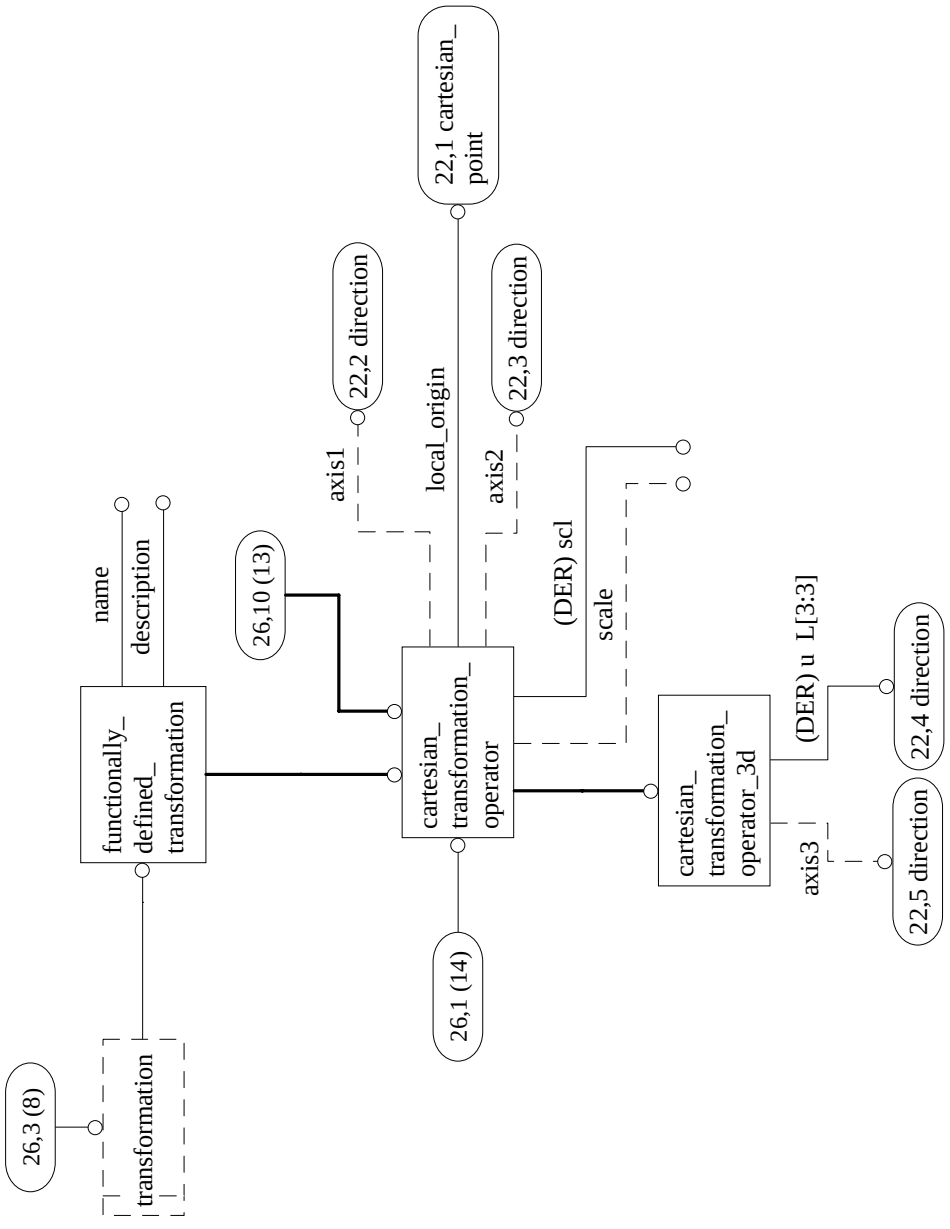


Figure H.26 - AIM diagram 26 of 29 in EXPRESS-G

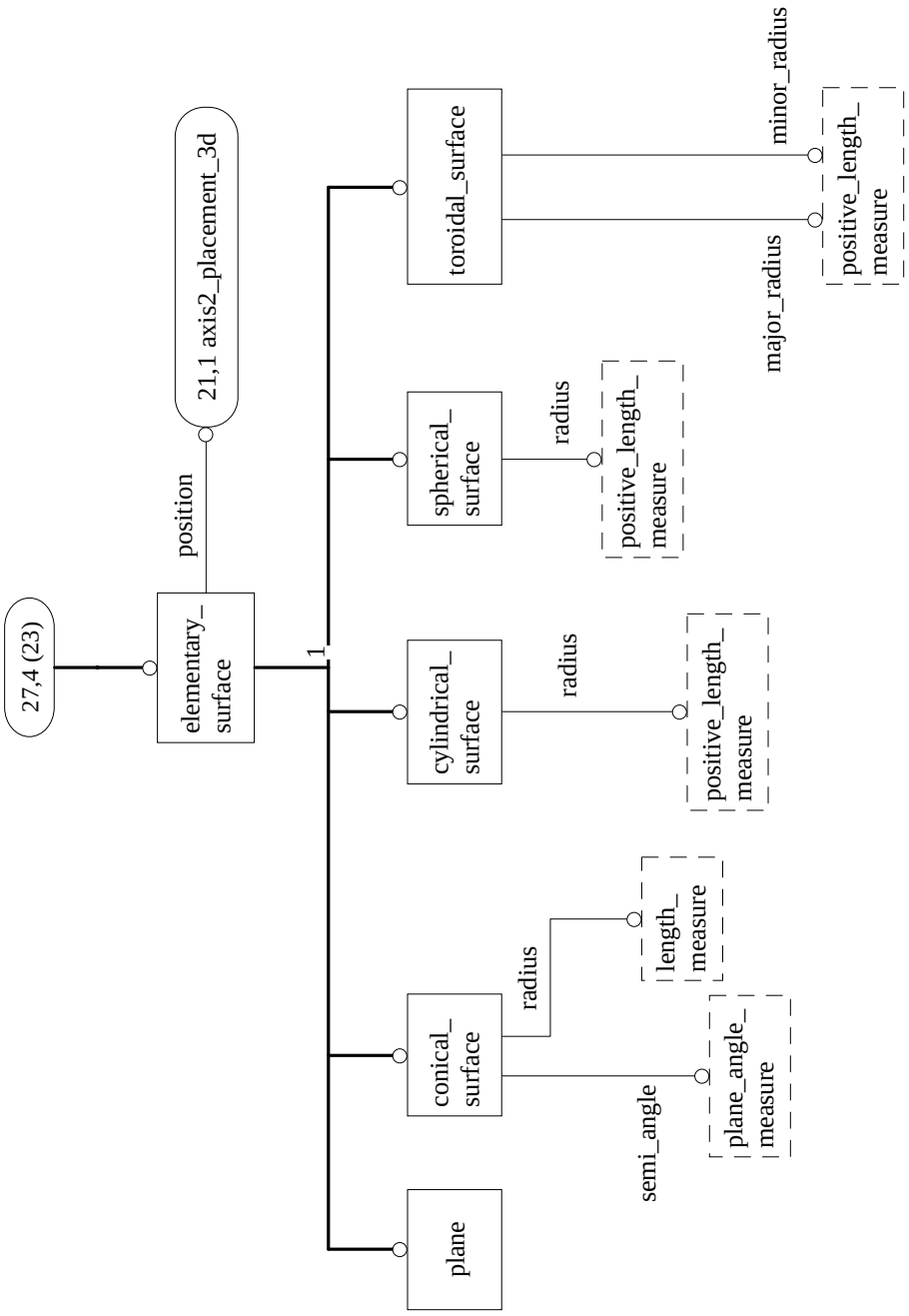


Figure H.27 - AIM diagram 27 of 29 in EXPRESS-G

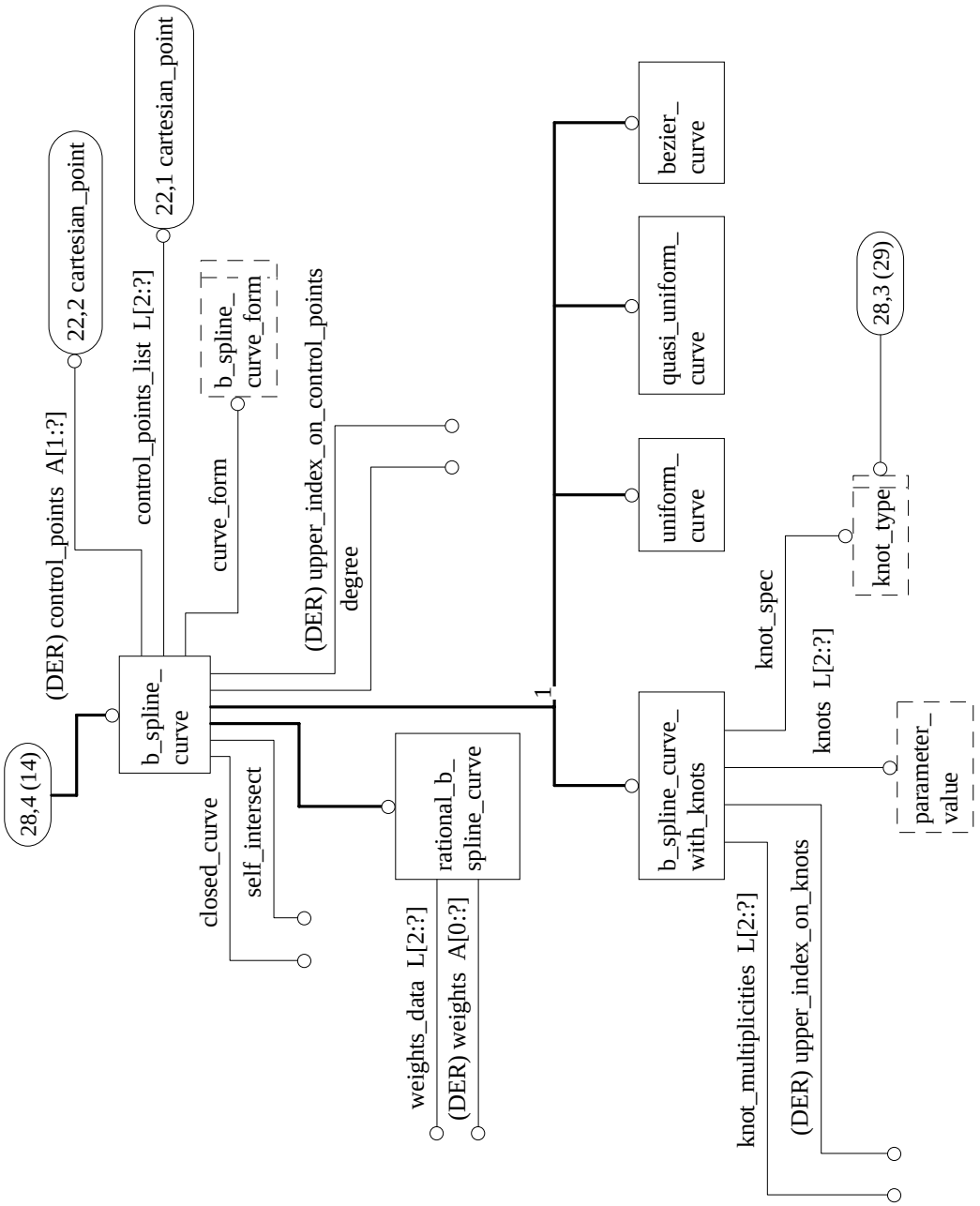


Figure H.28 - AIM diagram 28 of 29 in EXPRESS-G

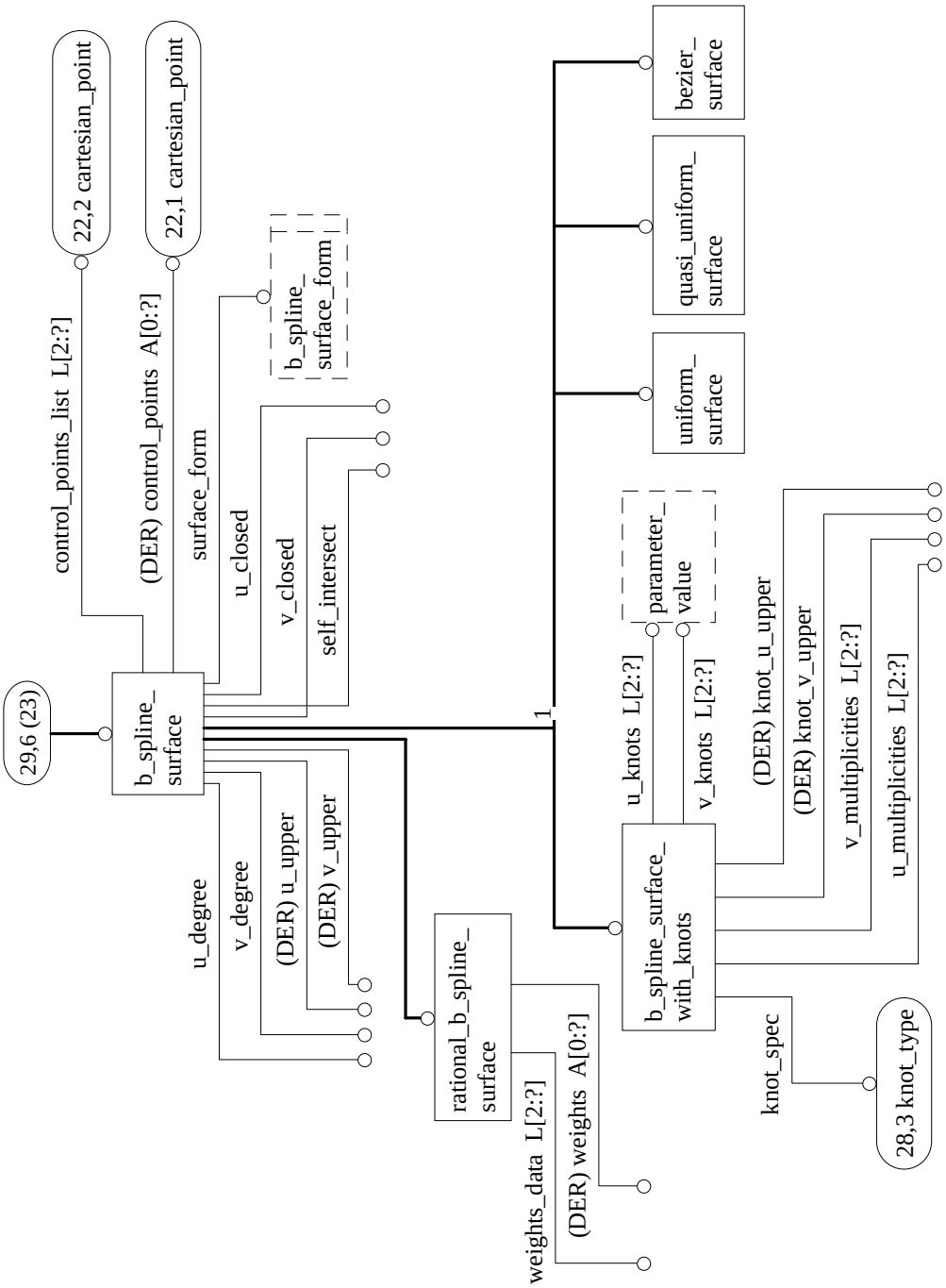


Figure H.29 - AIM diagram 29 of 29 in EXPRESS-G

Annex J
(informative)

AIM EXPRESS

This annex provides a listing of the EXPRESS entity names and corresponding short names as specified in this part of ISO 10303. It also provides a listing of the complete EXPRESS schema specified in this part of ISO 10303 without comments or other explanatory text.

This annex is available in computer-interpretable form and can be found at the following URLs:

Short names: http://www.mel.nist.gov/div826/subject/apde/snr/
EXPRESS: http://www.mel.nist.gov/step/parts/part225/is/

If there is difficulty accessing these sites contact ISO Central Secretariat or contact the ISO TC 184/SC4 Secretariat directly at: sc4sec@cme.nist.gov.

NOTE - The information provided in computer-interpretable form at the above URLs is informative. The information that is contained in the body of this part of ISO 10303 is normative.

Annex K
(informative)

Bibliography

[1] *Federal Information Processing Standards Publication 183, Integration Definition for Function Modeling (IDEF0)*, FIPS PUB 183, National Institute of Standards and Technology, December 1993.

[2] HOFMANN-WELLENHOF, B., LICHTENEGGER, H., AND COLLINS, J. *Global Positioning System - Theory and Practise*. Third Edition, Springer-Verrlag Wien, New York, 1994.

[3] ISO/CD 4157-2 "Construction Drawings - Designation Systems - Part 2 Designation of Rooms and other Spaces". ISO TC10/SC8 N339.

[4] ISO 10303-32[1)], *Industrial automation systems and integration - Product data representation and exchange - Part 32: Conformance testing methodology and framework: Requirements on testing laboratories and clients*.

[5] ISO 10303-325[1)], *Industrial automation systems and integration - Product data representation and exchange - Part 325: Abstract test suite: Building elements using explicit shape representation*.

[1)] To be published

Index

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

12.

13.

14.

15.

16.

17.

18.

19.

20.

21.

22.

23.

24.

25.

26.

27.

28.

29.

30.

31.

32.

33.

34.

35.

36.

37.

38.

ICS 25.040.40

Price based on 396 pages