INTERNATIONAL STANDARD

ISO 10303-502

First edition
2000-03-01

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

Part 502: Application interpreted construct: Shell-based wireframe

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

Partie 502: Construction interprétée d'application: Cadre en fil métallique basé sur une coquille

Reference number
ISO 10303-502:2000(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

Figures

Tables

Foreword

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

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

International Standard ISO 10303-501 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 107, Engineering Analysis Core Application reference model (EA C-ARM);

— Part 108, Integrated application resource: Parameterization and constraints for explicit geometric product models

— 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 mechanical design processes;

— 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 220, Application protocol: Process planning, manufacture, and assembly of layered electronic products

— 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 233, Application Protocol: Systems engineering data representation

— Part 234, Application protocol: Ship Operational logs, records, and messages

— Part 235, Application Protocol: Materials information for the design and verification of products

— 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 processes;

— Part 315, Abstract test suite: Ship arrangement;

— Part 316, Abstract test suite: Ship moulded for

— 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 334, Abstract test suitel: Ship Operational logs, records, and messages

— Part 335, Abstract test suite: Materials information for the design and verification of products

— 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 presentation;

— Part 519, Application interpreted construct: Geometric tolerances;

— Part 520, Application interpreted construct: Associative draughting.

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 108 specify the integrated application resources,

— Parts 201 to 235 specify the application protocols,

— Parts 301 to 335 specify the abstract test suites, and

— Parts 501 to 520 specify the application interpreted constructs.

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

Annexes A and B form an integral part of this part of ISO 10303. Annex C and D are for information only.

Introduction

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

This part of ISO 10303 is a member of the application interpreted construct series. An application interpreted construct (AIC) provides a logical grouping of interpreted constructs that supports a specific functionality for the usage of product data across multiple application contexts. An interpreted construct is a common interpretation of the integrated resources that supports shared information requirements among application protocols.

This document specifies the application interpreted construct for the description of the product shape using three-dimensional wireframe models that are bounded by a set of shells.

Industrial automation systems and integration — Product data representation and exchange — Part 502: Application interpreted construct: Shell-based wireframe

1 Scope

This part of ISO 10303 specifies the interpretation of the integrated resources to satisfy requirements for the representation of the product shape using three-dimensional wireframe models which are bounded by a set of shells.

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

— the representation of wireframe models that are described by a graph of edges and vertices where the edges intersect only at their vertices;

— the representation of a wireframe model by one or more shells that shall not overlap or intersect except at their vertices or edges;

— points defined in a three-dimensional coordinate space;

— curves, including b-splines, defined in a three-dimensional coordinate space;

— the representation of a single wireframe model or an assembly of wireframe models.

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

— geometry defined in a two-dimensional coordinate space;

— surface geometry;

— solid geometry;

— references to eliminated constructs.

2 Normative references

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

ISO/IEC 8824-1:1995, *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 10303-41:1994, *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 10303-202:1996, *Industrial automation systems and integration — Product data representation and exchange — Part 202: Application protocol: Associative draughting.*

3 Terms, definitions, and abbreviations

3.1 Terms defined in ISO 10303-1

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

— application;

— application context;

— application protocol;

— implementation method;

— integrated resource;

— interpretation;

— model;

— product;

— product data.

3.2 Terms defined in ISO 10303-202

For the purpose of this part of ISO 10303, the following terms defined in ISO 10303-202 apply.

— Application interpreted construct (AIC).

3.3 Abbreviations

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

AIC application interpreted construct

AP application protocol

4 EXPRESS short listing

This clause specifies the EXPRESS schema that uses elements from the integrated resources and contains the types, entity specializations, and functions that are specific to this part of ISO 10303.

NOTE 1 - There may be subtypes and items of select lists that appear in the integrated resources that are not imported into the AIC. Constructs are eliminated from the subtype tree or select list through the use of the implicit interface rules of ISO 10303-11.

EXPRESS Specification:

```
*)
SCHEMA aic_shell_based_wireframe;

USE FROM geometric_model_schema                             -- ISO 10303-42
  (shell_based_wireframe_model);

USE FROM geometry_schema                                    -- ISO 10303-42
  (axis2_placement_3d,
   b_spline_curve_with_knots,
   bezier_curve,
   cartesian_transformation_operator_3d,
   circle,
   conic,
   curve,
   curve_replica,
   ellipse,
   geometric_representation_context,
   hyperbola,
   line,
   offset_curve_3d,
   parabola,
   point,
   point_replica,
   polyline,
   quasi_uniform_curve,
   rational_b_spline_curve,
   uniform_curve);
```

```
USE FROM product_property_representation_schema          -- ISO 10303-41
  (shape_representation);

USE FROM representation_schema                           -- ISO 10303-43
  (mapped_item);

USE FROM topology_schema                                 -- ISO 10303-42
  (edge_curve,
   edge_loop,
   path,
   vertex_loop,
   vertex_point,
   vertex_shell,
   wire_shell);
(*
```

NOTE 2 - The schemas referenced above can be found in the following parts of ISO 10303:

geometric_model_schema	ISO 10303-42
geometry_schema	ISO 10303-42
product_property_representation_schema	ISO 10303-41
topology_schema	ISO 10303-42
representation_schema	ISO 10303-43

4.1 Introduction

This part of ISO 10303 provides the geometric and topological structures to represent three dimensional shapes using shells of dimensionality 0 (**vertex_shell**) or 1 (**wire_shell**). This AIC is represented by the entity **shell_based_wireframe_shape_representation**, that is a **shape_representation**

4.2 Fundamental concepts and assumptions

The wireframe representation of shapes is based on the **shell_based_wireframe_model** and the geometric and topological entities required. The shapes represented by the **shell_based_wireframe_shape_-representation** are those whose definition can be satisfied by collections of **vertex_shell**s or **wire_shell**s. Wireframe representations may be assemblies of other wireframe representations that are mapped to the same coordinate space.

4.3 aic_shell_based_wireframe entity definition: shell_based_wireframe_-shape_representation

A **shell_based_wireframe_shape_representation** is a three-dimensional **shape_representation** that represents the shape or a portion of the shape of a product by wireframes that define an implicit volume. This includes all three-dimensional curves and topological entities that define a graph of vertices, edges, and loops.

NOTE - An application protocol that uses this AIC may ensure that the **shape_representation** entity is instantiated as a **shell_based_wireframe_shape_representation**.

EXPRESS Specification:

```
*)

ENTITY shell_based_wireframe_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (it <* SELF.items |
       NOT
(SIZEOF(['AIC_SHELL_BASED_WIREFRAME.SHELL_BASED_WIREFRAME_MODEL',
                    'AIC_SHELL_BASED_WIREFRAME.MAPPED_ITEM',
                    'AIC_SHELL_BASED_WIREFRAME.AXIS2_PLACEMENT_3D'] *
            TYPEOF (it)) = 1
       ))) = 0;
  WR2: SIZEOF (QUERY (it <* SELF.items |
             SIZEOF(['AIC_SHELL_BASED_WIREFRAME.SHELL_BASED_WIREFRAME_MODEL',
             'AIC_SHELL_BASED_WIREFRAME.MAPPED_ITEM'] * TYPEOF (it)) = 1
       )) >= 1;
  WR3: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
         ('AIC_SHELL_BASED_WIREFRAME.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
                             sbwm\shell_based_wireframe_model.sbwm_boundary |
                   ('AIC_SHELL_BASED_WIREFRAME.WIRE_SHELL' IN TYPEOF (sb))) |
           NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                               ws\wire_shell.wire_shell_extent |
                ('AIC_SHELL_BASED_WIREFRAME.EDGE_LOOP' IN TYPEOF (wsb))) |
             NOT (SIZEOF (QUERY (el <* eloop\path.edge_list |
               NOT ('AIC_SHELL_BASED_WIREFRAME.EDGE_CURVE' IN
                       TYPEOF (el.edge_element)))) = 0)
           )) = 0)
         )) = 0)
       )) = 0;
  WR4: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
         ('AIC_SHELL_BASED_WIREFRAME.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
                             sbwm\shell_based_wireframe_model.sbwm_boundary |
                   ('AIC_SHELL_BASED_WIREFRAME.WIRE_SHELL' IN TYPEOF (sb))) |
           NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                               ws\wire_shell.wire_shell_extent |
               ('AIC_SHELL_BASED_WIREFRAME.EDGE_LOOP' IN TYPEOF (wsb))) |
             NOT (SIZEOF (QUERY (pline_el <*
                          QUERY (el <* eloop\path.edge_list |
                    ('AIC_SHELL_BASED_WIREFRAME.POLYLINE' IN
                        TYPEOF (el.edge_element\edge_curve.edge_geometry))) |
               NOT (SIZEOF (pline_el.edge_element\edge_curve.
                            edge_geometry\polyline.points) >2)
             )) = 0)
           )) = 0)
         )) = 0)
       )) = 0;
  WR5: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
         ('AIC_SHELL_BASED_WIREFRAME.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
                             sbwm\shell_based_wireframe_model.sbwm_boundary |
                   ('AIC_SHELL_BASED_WIREFRAME.WIRE_SHELL' IN TYPEOF (sb))) |
           NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                                ws\wire_shell.wire_shell_extent |
                 ('AIC_SHELL_BASED_WIREFRAME.EDGE_LOOP' IN TYPEOF (wsb))) |
```

```
             NOT (SIZEOF (QUERY (el <* eloop\path.edge_list |
                NOT (valid_wireframe_edge_curve
                       (el.edge_element\edge_curve.edge_geometry )))) =0)
        )) = 0)
       )) = 0)
     )) = 0;
  WR6: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
          ('AIC_SHELL_BASED_WIREFRAME.SHELL_BASED_WIREFRAME_MODEL'
               IN TYPEOF(it))) |
         NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
                              sbwm\shell_based_wireframe_model.sbwm_boundary |
                   ('AIC_SHELL_BASED_WIREFRAME.WIRE_SHELL' IN TYPEOF (sb))) |
           NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                                ws\wire_shell.wire_shell_extent |
                   ('AIC_SHELL_BASED_WIREFRAME.EDGE_LOOP' IN TYPEOF (wsb))) |
             NOT (SIZEOF (QUERY (el <* eloop\path.edge_list |
               NOT (('AIC_SHELL_BASED_WIREFRAME.VERTEX_POINT' IN
                        TYPEOF (el.edge_element.edge_start))
                AND
                    ('AIC_SHELL_BASED_WIREFRAME.VERTEX_POINT' IN
                        TYPEOF (el.edge_element.edge_end))) )) = 0)
         )) = 0)
       )) = 0)
     )) = 0;
  WR7: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
         ('AIC_SHELL_BASED_WIREFRAME.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
                              sbwm\shell_based_wireframe_model.sbwm_boundary |
                   ('AIC_SHELL_BASED_WIREFRAME.WIRE_SHELL' IN TYPEOF (sb))) |
           NOT (SIZEOF (QUERY (eloop <* QUERY (wsb <*
                                 ws\wire_shell.wire_shell_extent |
               ('AIC_SHELL_BASED_WIREFRAME.EDGE_LOOP' IN TYPEOF (wsb))) |
             NOT (SIZEOF (QUERY (el <* eloop\path.edge_list |
                    NOT ((valid_wireframe_vertex_point
                          (el.edge_element.
                           edge_start\vertex_point.vertex_geometry))
                 AND
                   (valid_wireframe_vertex_point
                 (el.edge_element.edge_end\vertex_point.vertex_geometry)))
             )) = 0)
           )) = 0)
         )) = 0)
       )) = 0;

  WR8: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
          ('AIC_SHELL_BASED_WIREFRAME.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF(it))) |
          NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
                               sbwm\shell_based_wireframe_model.sbwm_boundary |
                    ('AIC_SHELL_BASED_WIREFRAME.WIRE_SHELL' IN TYPEOF (sb))) |
            NOT (SIZEOF (QUERY (vloop <* QUERY (wsb <*
                                 ws\wire_shell.wire_shell_extent |
                  ('AIC_SHELL_BASED_WIREFRAME.VERTEX_LOOP' IN TYPEOF (wsb))) |
              NOT ('AIC_SHELL_BASED_WIREFRAME.VERTEX_POINT' IN
                      TYPEOF (vloop\vertex_loop.loop_vertex))
            )) = 0)
          )) = 0)
        )) = 0;
  WR9: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
          ('AIC_SHELL_BASED_WIREFRAME.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF(it))) |
```

```
          NOT (SIZEOF (QUERY (ws <* QUERY (sb <*
                              sbwm\shell_based_wireframe_model.sbwm_boundary |
                    ('AIC_SHELL_BASED_WIREFRAME.WIRE_SHELL' IN TYPEOF (sb))) |
            NOT (SIZEOF (QUERY (vloop <* QUERY (wsb <*
                                ws\wire_shell.wire_shell_extent |
                  ('AIC_SHELL_BASED_WIREFRAME.VERTEX_LOOP' IN TYPEOF (wsb))) |
              NOT (valid_wireframe_vertex_point (vloop\vertex_loop.
                    loop_vertex\vertex_point.vertex_geometry))
            )) = 0)
          )) = 0)
        )) = 0;
  WR10: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
          ('AIC_SHELL_BASED_WIREFRAME.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF(it))) |
          NOT (SIZEOF (QUERY (vs <* QUERY (sb <*
                              sbwm\shell_based_wireframe_model.sbwm_boundary |
                  ('AIC_SHELL_BASED_WIREFRAME.VERTEX_SHELL' IN TYPEOF (sb))) |
            NOT ('AIC_SHELL_BASED_WIREFRAME.VERTEX_POINT' IN
                    TYPEOF (vs\vertex_shell.vertex_shell_extent.loop_vertex))
          )) = 0)
        )) = 0;
  WR11: SIZEOF (QUERY (sbwm <* QUERY (it <* SELF.items |
          ('AIC_SHELL_BASED_WIREFRAME.SHELL_BASED_WIREFRAME_MODEL'
              IN TYPEOF(it))) |
          NOT (SIZEOF (QUERY (vs <* QUERY (sb <*
                              sbwm\shell_based_wireframe_model.sbwm_boundary |
                  ('AIC_SHELL_BASED_WIREFRAME.VERTEX_SHELL' IN TYPEOF (sb))) |
            NOT (valid_wireframe_vertex_point (vs\vertex_shell.
                   vertex_shell_extent.loop_vertex\vertex_point.
                   vertex_geometry))
          )) = 0)
        )) = 0;
  WR12: SIZEOF (QUERY (mi <* QUERY (it <* SELF.items |
           ('AIC_SHELL_BASED_WIREFRAME.MAPPED_ITEM' IN TYPEOF (it))) |
         NOT ('AIC_SHELL_BASED_WIREFRAME.' +
                 'SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION' IN
           TYPEOF (mi\mapped_item.mapping_source.mapped_representation)
        ))) = 0;
  WR13: SELF.context_of_items\geometric_representation_context.
             coordinate_space_dimension = 3;
END_ENTITY;
(*
```

Formal propositions:

WR1: The **items** in a **shell_based_wireframe_shape_representation** shall be a **shell_based_wireframe_-model**, **mapped_item**, or **axis2_placement_3d**.

WR2: At least one of the **items** in a **shell_based_wireframe_shape_representation** shall be either a **shell_based_wireframe_model** or a **mapped_item**.

WR3: Every **edge** defined for an **edge_loop** in a **shell_based_wireframe_model** shall be an **edge_curve**.

WR4: Every **polyline** that underlies an **edge** in a **shell_based_wireframe_model** shall contain more than two distinct **point**s.

WR5: The **edge_geometry** that underlies an **edge** for a **shell_based_wireframe_model** shall be a **line**, **conic**, **b_spline_curve**, **offset_curve_3d**, **polyline**, or **curve_replica**, and the curves that have a basis defined by other curves are done so consistently. In the case of an **offset_curve_3d** or **curve_replica** the curve referenced as **basis_curve** shall be one of the above types.

WR6: Every **vertex** defined as the start or end vertex for an **edge** in a **shell_based_wireframe_model** shall be a **vertex_point**.

WR7: The **vertex_geometry** that underlies the vertices that define the boundaries of the **edge**s in an **edge_-loop** for a **shell_based_wireframe_model** shall be a **cartesian_point** or **point_replica**, and the **point_-replica** shall replicate either another **point_replica** or a **cartesian_point**.

WR8: The vertex that defines the **vertex_loop** that is used as a boundary in a **shell_based_wireframe_-model** shall be a **vertex_point**.

WR9: The vertex that defines the **vertex_loop** that is used as a boundary in a **shell_based_wireframe_-model** shall be underlaid by a **cartesian_point** or **point_replica**, and the **point_replica** shall replicate either another **point_replica** or a **cartesian_point**.

WR10: The vertex that defines the **vertex_loop** that is used as the **vertex_shell_extent** for a **vertex_shell** in a **shell_based_wireframe_model** shall be a **vertex_point**.

WR11: The vertex that defines the **vertex_loop** that is used as the **vertex_shell_extent** for a **vertex_shell** in a **shell_based_wireframe_model** shall be underlaid by a **cartesian_point** or **point_replica**, and the **point_replica** shall replicate either another **point_replica** or a **cartesian_point**.

WR12: If there is a **mapped_item** in a **shell_based_wireframe_shape_representation**, the source of the **mapped_item** shall be a **shell_based_wireframe_shape_representation**.

WR13: The **shell_based_wireframe_shape_representation** shall have **coordinate_space_dimension** equal to three.

4.4 aic_shell_based_wireframe function definitions

4.4.1 valid_wireframe_edge_curve

The **valid_wireframe_edge_curve** function determines whether or not an input curve is valid for use in representing a shape defined by a topologically bounded wireframe.

EXPRESS specification:

```
*)
FUNCTION valid_wireframe_edge_curve (crv   : curve) : BOOLEAN;
  -- check for valid basic curve types

  IF SIZEOF (['AIC_SHELL_BASED_WIREFRAME.LINE',
              'AIC_SHELL_BASED_WIREFRAME.CONIC',
              'AIC_SHELL_BASED_WIREFRAME.B_SPLINE_CURVE',
              'AIC_SHELL_BASED_WIREFRAME.POLYLINE'] * TYPEOF (crv)) = 1
    THEN RETURN (TRUE);
  ELSE
  -- recursively check for valid basic curves for curve_replica
```

```
    IF ('AIC_SHELL_BASED_WIREFRAME.CURVE_REPLICA') IN TYPEOF (crv)
      THEN RETURN (valid_wireframe_edge_curve
                          (crv\curve_replica.parent_curve));
    ELSE
      -- recursively check for valid basis curves for offset_curve
      IF ('AIC_SHELL_BASED_WIREFRAME.OFFSET_CURVE_3D') IN TYPEOF (crv)
        THEN RETURN (valid_wireframe_edge_curve
                          (crv\offset_curve_3d.basis_curve));
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;
(*
```

Argument definitions:

crv: the input curve that is to be examined. A valid curve is either a **line**, **conic**, **b_spline_curve**, **polyline**, **offset_curve_3d**, or **curve_replica**. In the case of an **offset_curve_3d** or **curve_replica** the curve referenced as **basis_curve** or **parent_curve** shall be a valid curve.

4.4.2 valid_wireframe_vertex_point

The **valid_wireframe_vertex_point** function determines whether or not an input point is valid for use in representing a shape defined by a topologically bounded wireframe.

EXPRESS specification:

```
*)
FUNCTION valid_wireframe_vertex_point (pnt   : point): BOOLEAN;
  -- check for valid basic point types
  IF ( 'AIC_SHELL_BASED_WIREFRAME.CARTESIAN_POINT' IN TYPEOF (pnt))
    THEN RETURN (TRUE);
  ELSE
    -- recursively check for valid basic point types as parents for a
    -- point_replica

    IF ('AIC_SHELL_BASED_WIREFRAME.POINT_REPLICA') IN TYPEOF (pnt)
      THEN RETURN (valid_wireframe_vertex_point
                          (pnt\point_replica.parent_pt));
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;
(*
```

Argument definitions:

pnt: the input point that is to be examined. A valid point is either a **cartesian_point** or a **point_replica**. In the case of a **point_replica** the **parent_point** shall also be a valid point.

```
*)
END_SCHEMA;
(*
```

Annex A
(normative)

Short names of entities

Table A.1 provides the short names of entities specified in the EXPRESS short listing 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 A.1 - Short names of entities

Entity names	Short names
SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION	SBWSR

Annex B
(normative)

Information object registration

B.1 Document identification

To provide for unambiguous identification of an information object in an open system, the object identifier

{ iso standard 10303 part(502) version(1) }

is assigned to this part of ISO 10303. The meaning of this value is defined in ISO/IEC 8824-1 and is described in ISO 10303-1.

B.2 Schema identification

To provide for unambiguous identification of the aic-shell-based-wireframe-schema in an open system, the object identifier

{ iso standard 10303 part(502) version(1) object(1) aic-shell-based-wireframe-schema(1) }

is assigned to the aic_shell_based_wireframe_schema schema (see clause 4). The meaning of this value is defined in ISO/IEC 8824-1 and is described in ISO 10303-1.

Annex C
(informative)

EXPRESS-G diagrams

Figure C.1 through C.6 correspond to the EXPRESS generated from the short listing given in clause 4 using the interface specifications of ISO 10303-11. The diagrams use the EXPRESS-G graphical notation for the EXPRESS language. EXPRESS-G is defined in annex D of ISO 10303-11.

NOTE - The following select types: geometric_set_select, transformation, trimming_select, reversible_topology, vector_or_direction, and wireframe_select are interfaced into the AIC expanded listing according to the implicit interface rules of ISO 10303-11. These select types are not referenced by other entities in this part of ISO 10303.

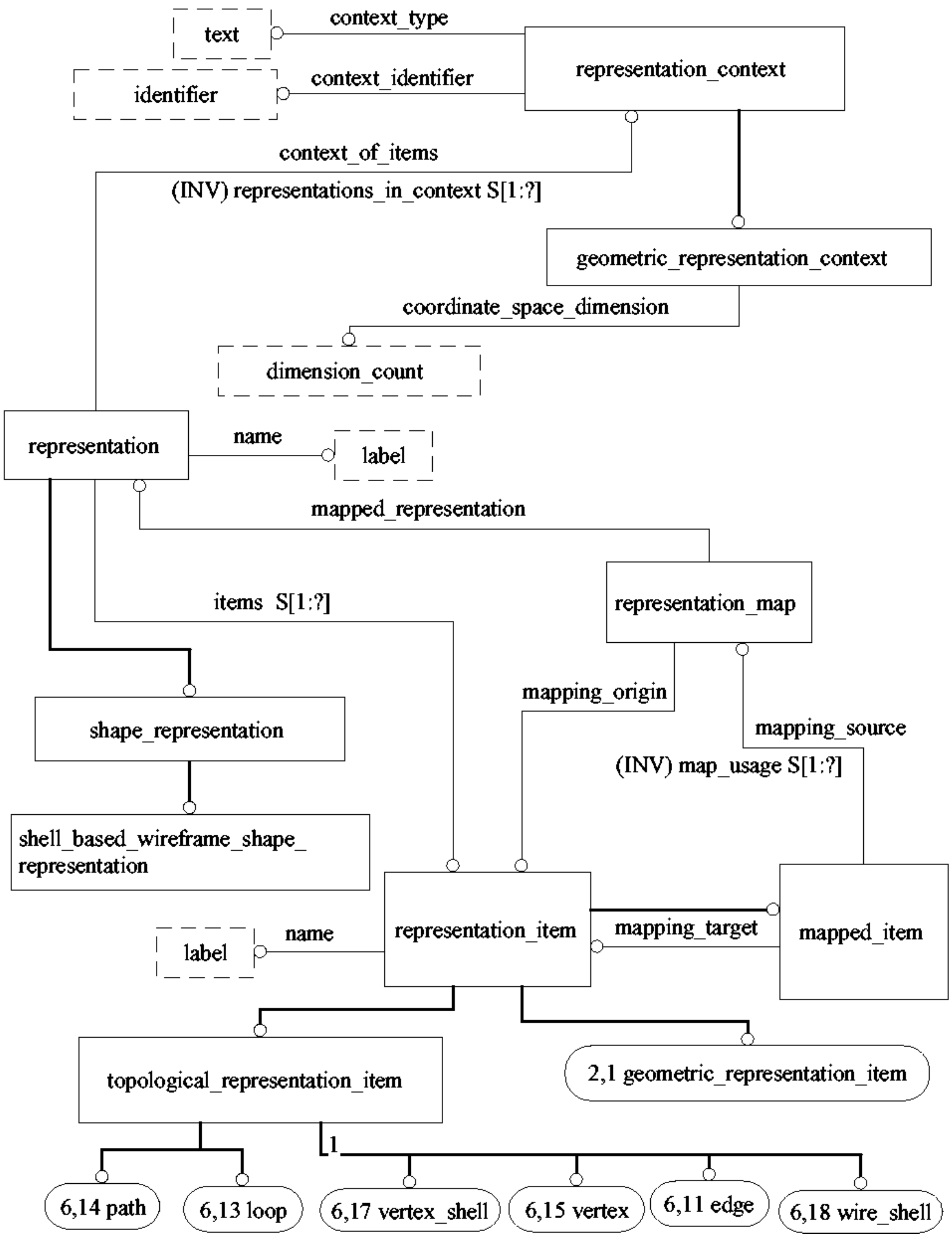

Figure C.1 - aic_shell_based_wireframe - EXPRESS-G diagram 1 of 6

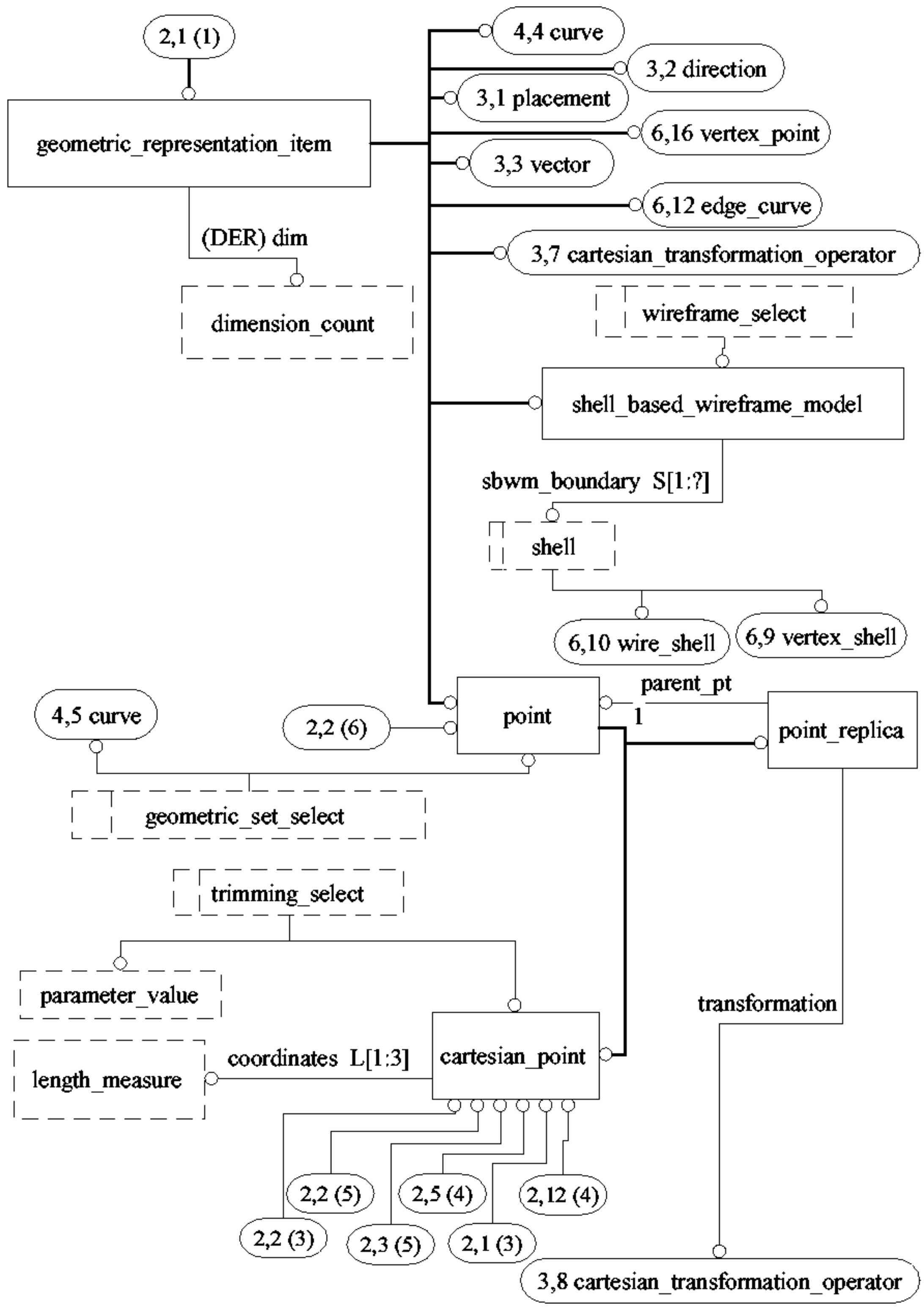

Figure C.2 - aic_shell_based_wireframe - EXPRESS-G diagram 2 of 6

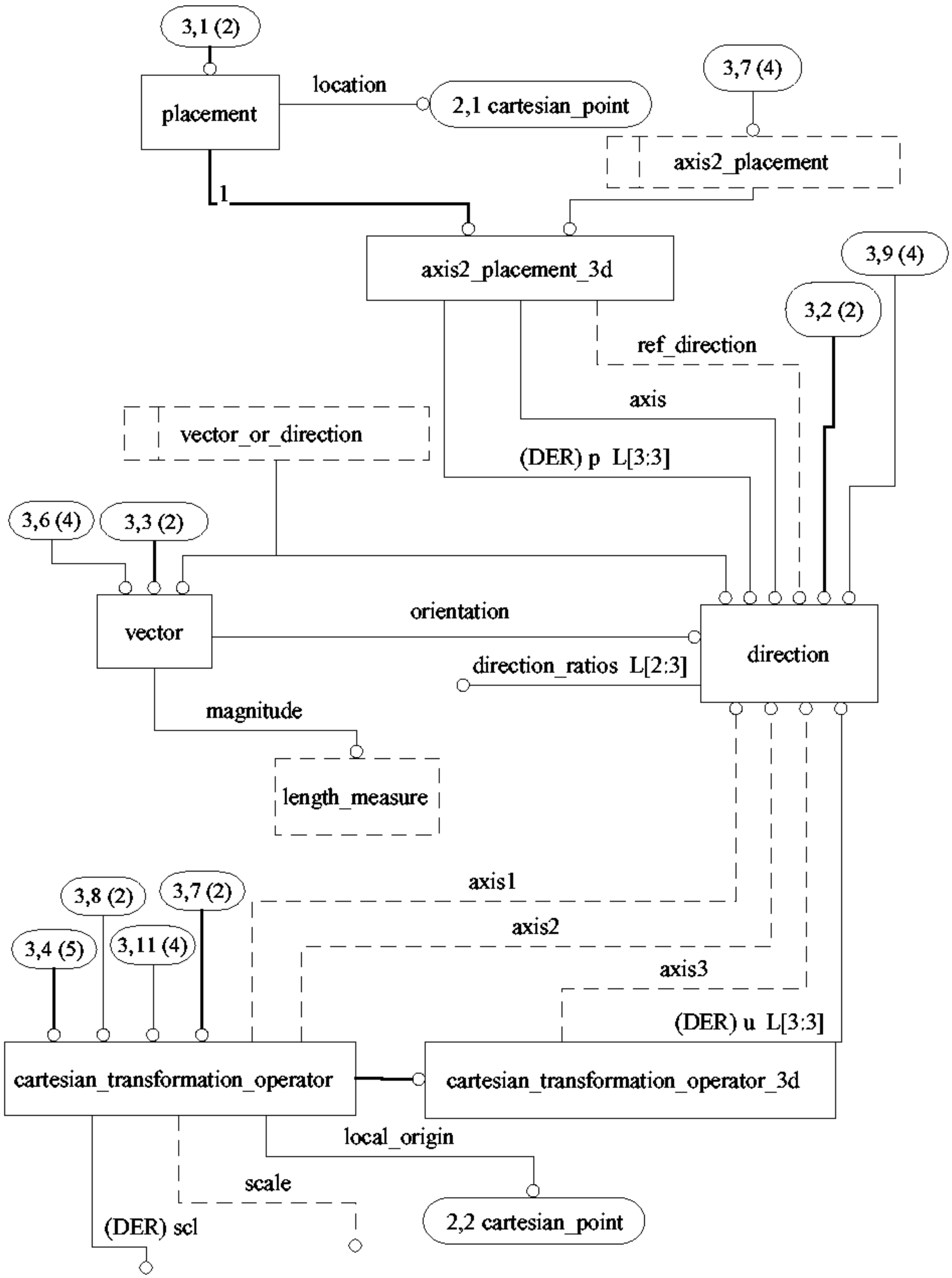

Figure C.3 - aic_shell_based_wireframe - EXPRESS-G diagram 3 of 6

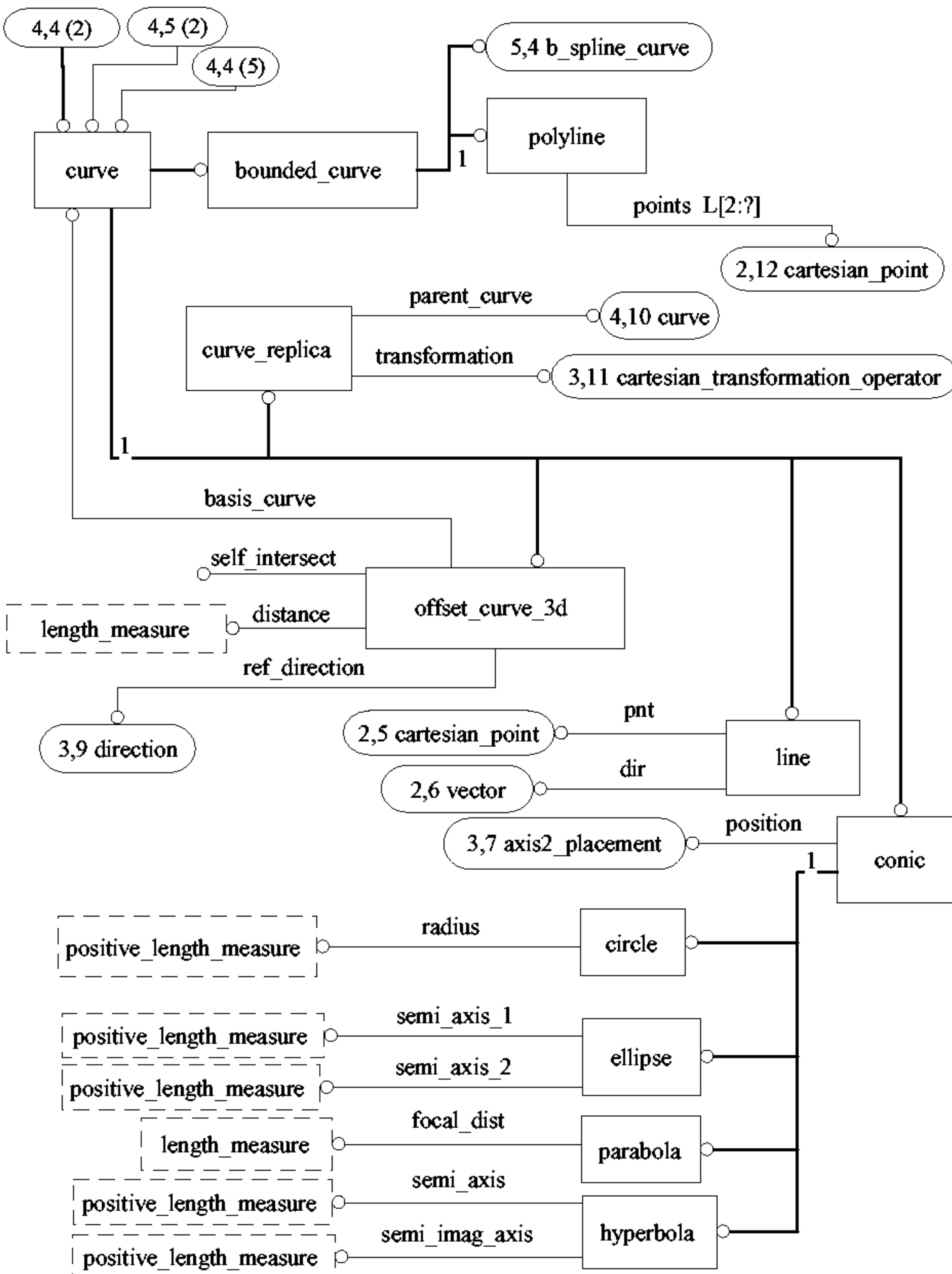

Figure C.4 - aic_shell_based_wireframe - EXPRESS-G diagram 4 of 6

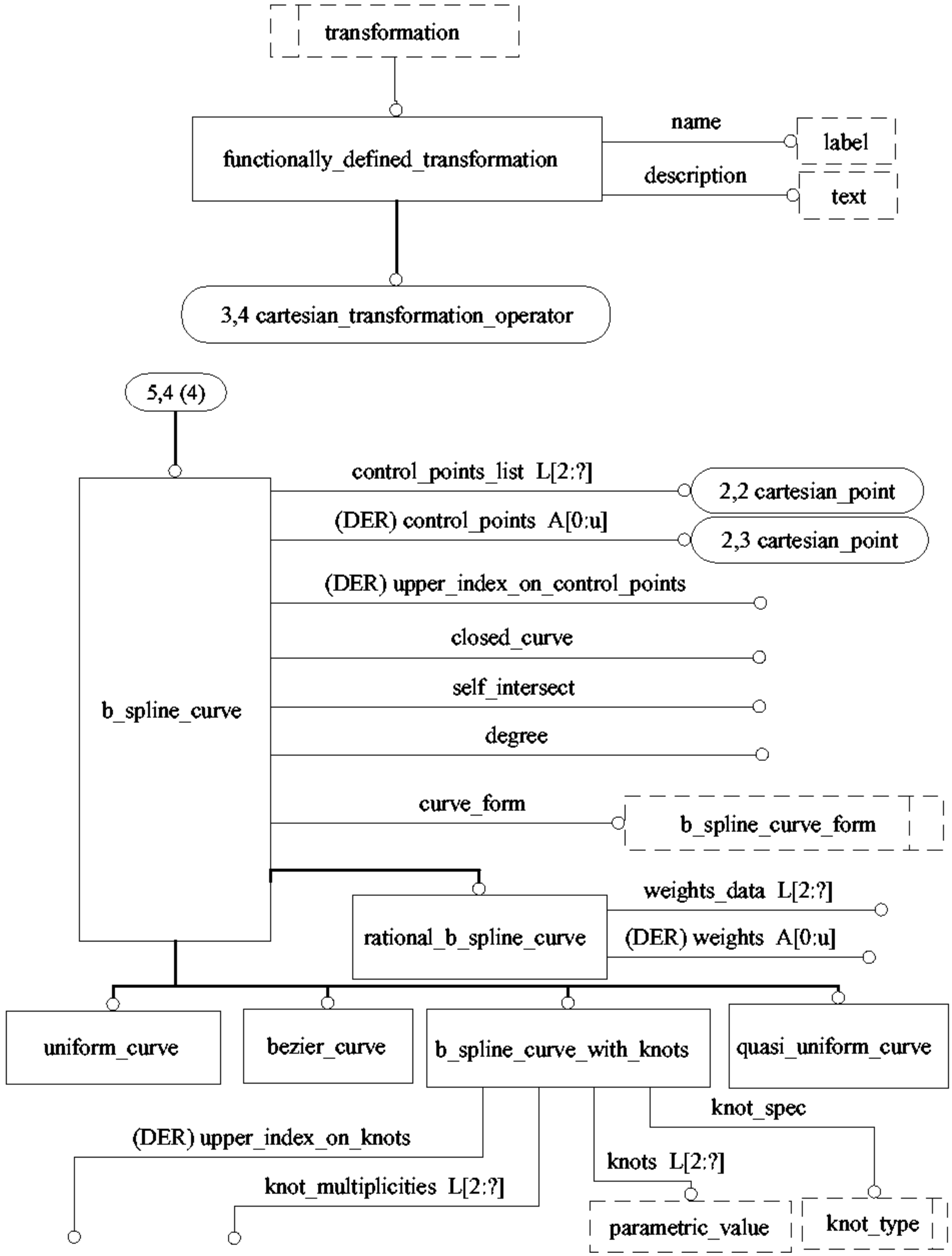

Figure C.5 - aic_shell_based_wireframe - EXPRESS-G diagram 5 of 6

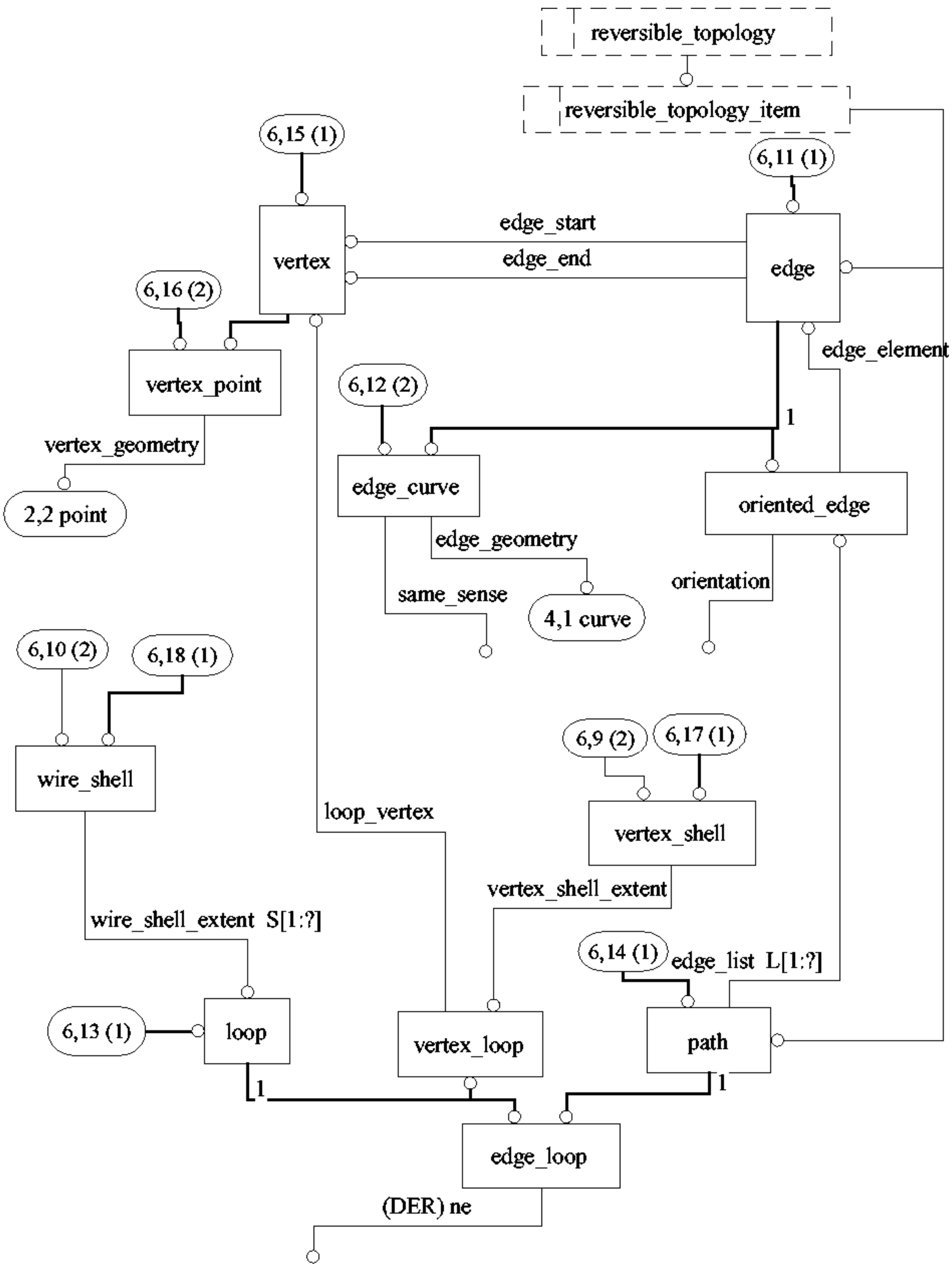

Figure C.6 - aic_shell_based_wireframe - EXPRESS-G diagram 6 of 6

Annex D
(informative)

Computer interpretable listings

This annex references 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 each EXPRESS schema specified in this part of ISO 10303 without comments or other explanatory text. These listings are available in computer-interpretable form and can be found at the following URLs:

Short names: http://www.mel.nist.gov/div826/subject/apde/snr/
EXPRESS: http://www.mel.nist.gov/step/parts/part502/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.

Index

ICS 25.040.40

Price based on 20 pages