INTERNATIONAL STANDARD

ISO 10303-207

First edition
1999-10-01

Corrected
2001-03-01

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

Part 207:
Application protocol: Sheet metal die planning and design

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

Partie 207: Protocole d'application: Prévision et conception des tôles d'acier coulé

Reference number
ISO 10303-207:1999(E)

Contents

Page

International Organization for Standardization
Case postale 56 · CH-1211 Genève 20 · Switzerland
Internet iso@iso.ch

Printed in Switzerland

Annexes

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 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 10303-207 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 33, Conformance testing methodology and framework: Structure and use of abstract test suites;

— 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 mechanical 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 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 228, Application protocol: Building services: Heating, ventilation, and air conditioning;

— 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 320, Abstract test suite: Process planning, manufacture, and assembly of layered electronic products;

— 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 328, Abstract test suite: Building services: Heating, ventilation, and air conditioning;

— 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 this International Standard reflects its structure:

— Parts 11 to 13 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 shall follow the same numbering pattern.

Annexes A, B, C, D, E, and F form an integral part of this part of ISO 10303. Annexes G, H, J, K, L, M, and N of this part of ISO 10303 are for information only.

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

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

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 sheet metal part design, part manufacture, and tool design information between contractors and suppliers. This exchange may be within enterprises, as when one department of an enterprise acts as a contractor, exchanging data with another department that acts as a supplier. This exchange may also be between enterprises, as when one enterprise, the contractor, exchanges information with another enterprise, the supplier. This AP satisfies an industrial need to exchange information to enable a contractor to manufacture a die tool used in the production of a sheet metal part. The information includes any combination of the following:

— Exchange of necessary information, such as sheet metal part geometry and configuration data, part tolerances, and part materials, from a contractor to a supplier to enable the supplier to produce a plan for manufacturing that sheet metal part;

— Exchange of necessary information, such as sheet metal part geometry and configuration data, part tolerances, and part materials, from a contractor to a supplier to enable the supplier to produce the design of a sheet metal die set (see figure 1);

— Exchange from a supplier to a contractor of a plan for the manufacture of a sheet metal part, possibly including press and process data, and plant constraint information;

— Exchange from a supplier to a contractor of the design of a sheet metal die, including die geometry and presentation data, die material data, die component configuration, and inter-relationships among the die component representations.

More information on the sheet metal forming process is included in annex L.

This application protocol defines the context, scope, and information requirements for sheet metal die planning and design, and specifies the integrated resources necessary to satisfy these requirements.

Figure 2 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 contains items, which are the representation of actual physical objects and their classification. Items are defined by item definitions. The item definitions describe whether they are standard or designed, and whether they are a part, die, or material. Item definitions may include features that are areas of interest on an item. A feature on a die may be a depression into which a boss on a press may be inserted; the purpose of describing the depression as a feature would be for the purpose of associating the use of the depression, which is fixturing, with the geometry of the depression.

Shape Definition Representation describes the shape of the item or the shape of a feature of the item. This representation is either a wireframe, surface, or solid. Shape Definition Representation also includes the physical model identification, tolerances, and the part shape relationship to the die shape.

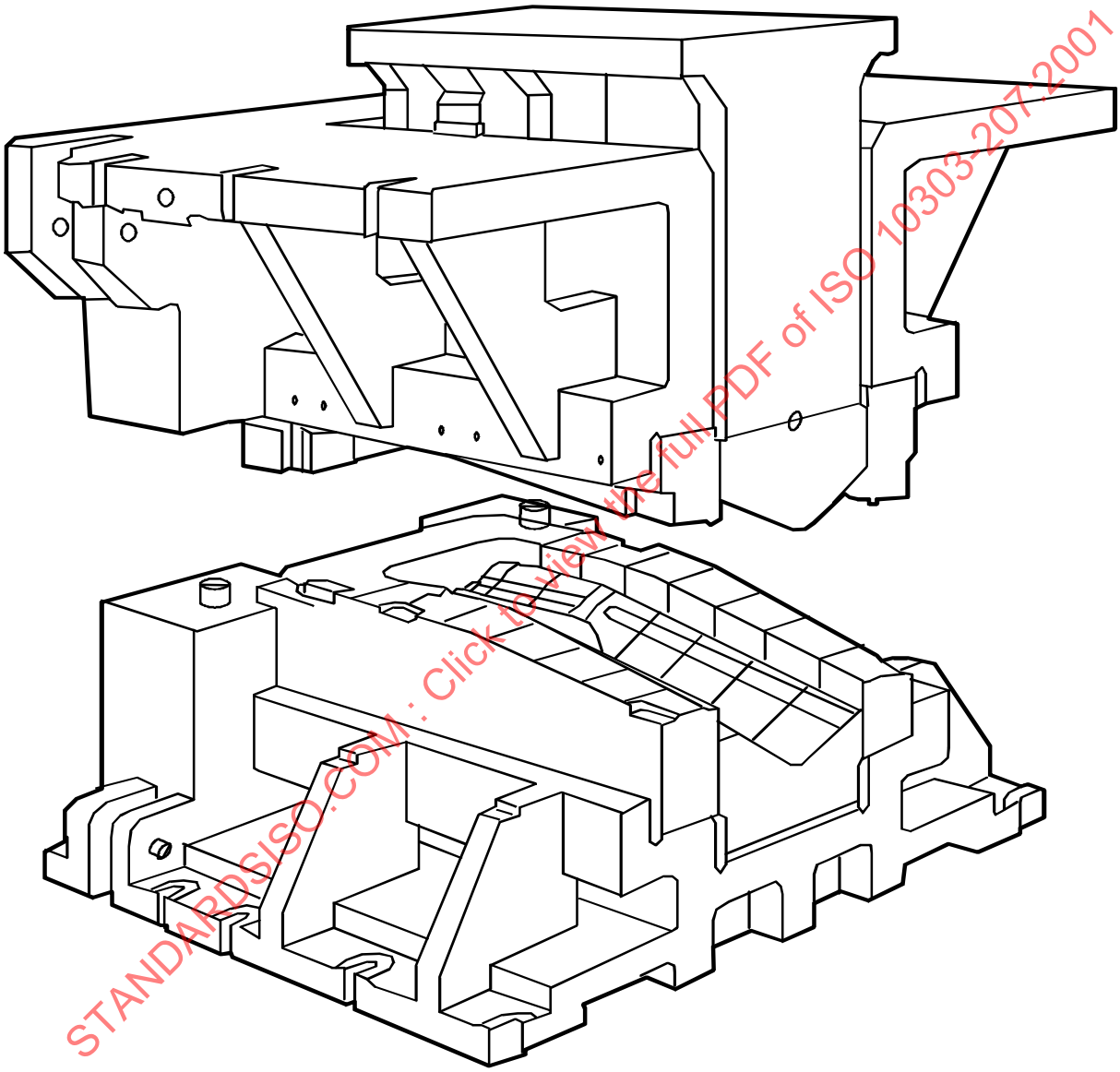

Figure 1 - Typical sheet metal die set

Part Process Plans define the manufacture of the part using dies as tools. The designs of both parts and dies are item definitions. Constraints, versions and component operations are included in the part process plans.

Work leads to the creation or modification of an item definition. A work request results in the development of part process plans.

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

Clause 1 defines the scope of the application protocol and summarizes the functionality and data covered by the AP. Clause 3 lists the words defined in this part of ISO 10303 and gives pointers to words defined elsewhere. An application activity model that is the basis for the definition of the scope is provided in annex G. 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 H.

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 that 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 J. Additional requirements for specific implementation methods are given in annex D.

.

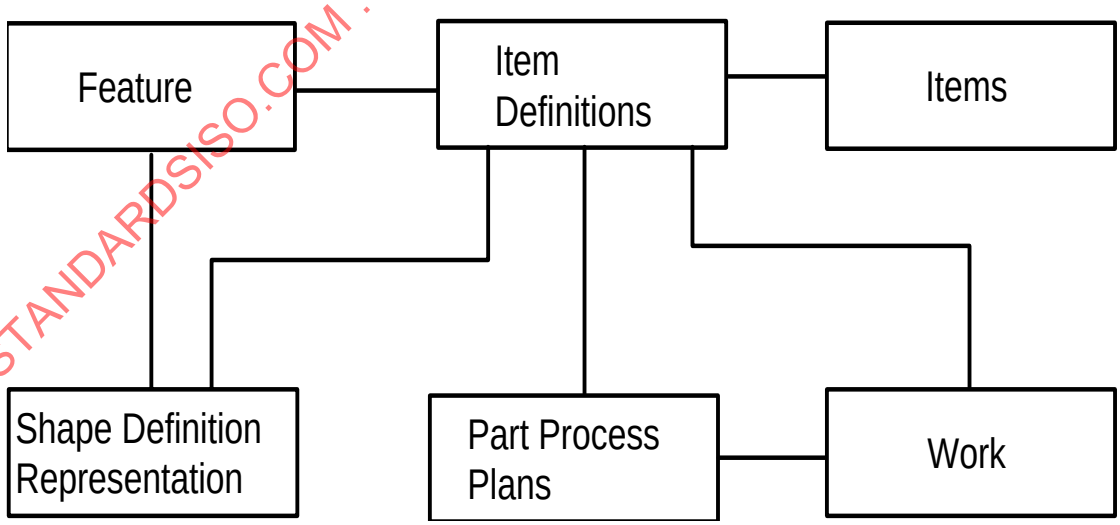

Figure 2 - Data planning model

Industrial automation systems and integration — Product data representation and exchange — Part 207: Application protocol: Sheet metal die planning and design

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 information between contractors and suppliers to enable the eventual manufacture of sheet metal dies used in the production process of sheet metal parts.

NOTE 1 - The application activity model, in annex G, 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:

— Types of products supported:

NOTE 2 - This list describes the types of products for which data are in scope. The list following this one describes types of product data which are in scope.

a) Sheet metal part design data;

EXAMPLE 1 - Sheet metal part designs may be for sheet metal parts intended for the exterior of a product, those intended for the interior of a product, those intended to be in view or not in view on the final product, or those intended to support loads or maintain structure of a product. Sheet metal part designs may also be for sheet metal parts that are products in themselves.

b) Sheet metal die design data, including die face design and die structure design, for an individual die against which sheet metal is formed by processes that do not involve a mating die;

c) Sheet metal die set design data, including die face design and die structure design, for die sets used in a stamping press machine to manufacture sheet metal parts;

d) Sheet metal part manufacture description data.

— Types of product data supported:

a) Design data for materials, sheet metal in-process parts, sheet metal parts, die components, individual dies, and die sets;

b) Process data for sheet metal part manufacture;

c) Change and schedule data for design of product definition data and manufacture description data;

d) Data ownership, generating system information, and exchange history surrounding product definition data and manufacture description data;

e) The identification of externally designed parts and purchased items;

f) Design constraints on dies;

g) Wireframe, surface, and solid geometry;

h) Data describing the relative position of materials and in-process sheet metal parts to the die or dies that will further form them;

i) Composition of materials, sheet metal parts, and die components;

j) Properties associated with materials or with collections of geometric representations, such as hardness, porosity, method of manufacture, and function.

— Stages in the product life cycle supported are data at any stage of completion that describe:

a) Materials;

b) Sheet metal in-process parts and sheet metal parts;

c) Die components, individual dies, and die sets;

d) Sheet metal part manufacture description data;

e) Change and schedule data for design of product definition data and manufacture description data.

— The supported exchange scenarios from contractor to supplier are as follows:

a) requirements to enable the supplier to create a sheet metal part processing plan for the contractor, such as the sheet metal part design, available presses and plants, and plant and press constraints;

b) requirements to enable the supplier to create a die design for the contractor, such as the sheet metal part design and the sheet metal part processing plan. This design may be for the die face, or for the die structure, or for both;

NOTE 3 - Requests for changes by contractors are considered to be the exchange of additional requirements.

c) exchanges wherein the contractor and supplier are divisions of the same company;

d) exchanges wherein the contractor and supplier are different companies.

— The supported exchange scenarios from supplier to contractor are as follows:

a) a part process plan or any portion thereof;

b) a complete die design or any portion thereof;

c) a die face design or any portion thereof;

d) a die structure design or any portion thereof;

e) a change request;

f) exchanges wherein the contractor and supplier are divisions of the same company;

g) exchanges wherein the contractor and supplier are different companies.

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

— Parts that are not made of sheet metal or are not manufactured by a process involving the use of a die or dies;

— Sheet metal parts that are manufactured by explosive forming, forging, or powdered metal processing;

— The design of devices used to stretch sheet metal over single convex dies, or the rubber bladder or sheet used in hydroforming or trap rubber forming to force sheet metal into a single concave die;

— Parametric or variational geometry models of sheet metal parts, dies, or die components;

— Engineering analysis data of any kind;

— Financial data of any kind;

— Manufacturing process data for sheet metal dies;

— Any exchange uses of data in order to:

a) enable the initial design of sheet metal parts;

b) enable the design of checking fixtures;

c) enable the manufacture of the die.

— Data related to production runs of sheet metal parts.

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 edition of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 31:1992, *Quantities and units (all parts).*

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

ISO 8601:1988, *Data elements and interchange formats — Information interchange — Representation of dates and times.*

ISO 8824-1:1995, *Information Technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

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

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

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

ISO 10303-22:—,[1] *Industrial automation systems and integration — Product data representation and exchange — Part 22: Implementation methods: Standard data access interface.*

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

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

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

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

---

[1] To be published.

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:—,[1)] *Industrial automation systems and integration — Product data representation and exchange — Part 45: Integrated generic resources: Materials.*

ISO 10303-47:—,[1)] *Industrial automation systems and integration — Product data representation and exchange — Part 47: Integrated generic resources: Shape variation tolerances.*

ISO 10303-49:—,[1)] *Industrial automation systems and integration — Product data representation and exchange — Part 49: Integrated generic resources: Process structure and properties.*

ISO/TR 10303-307:—,[1)] *Industrial automation systems and integration —Product data representation and exchange — Part 307: Abstract test suite for Sheet metal die planning and design.*

[1)] To be published.

3 Definitions

3.1 Terms defined in ISO 8601

This part of ISO 10303 makes use of the following terms defined in ISO 8601:

— calendar date;

— calendar week;

— day;

— Gregorian calendar;

— hour;

— leap year;

— minute;

— month;

— second;

— week;

— year.

3.2 Terms defined in ISO 10303-1

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

— abstract test suite (ATS);

— application;

— application activity model (AAM);

— application context;

— application interpreted model (AIM);

— application protocol (AP);

— application object;

— application reference model (ARM);

— assembly;

— conformance class;

— conformance requirement;

— context;

— data;

— data exchange;

— implementation method;

— information;

— integrated resource;

— interpretation;

— PICS proforma;

— protocol implementation conformance statement (PICS);

— product;

— product data;

— structure;

— unit of functionality (UoF).

3.3 Terms defined in ISO 10303-31

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

— conformance testing;

— postprocessor;

— preprocessor.

3.4 Terms defined in ISO 10303-42

This part of ISO 10303 makes use of the following terms defined in ISO 10303-42:

— arcwise connected;

— bounds;

— boundary;

— closed surface;

— connected;

— connected component;

— curve;

— cycle;

— dimensionality;

— domain;

— euler equations;

— extent;

— finite;

— graph;

— handle;

— inside;

— interior;

— list;

— *d*-manifold with boundary;

— orientable;

— overlap;

— set;

— space dimensionality;

— surface.

3.5 Terms defined in ISO 10303-43

This part of ISO 10303 makes use of the following terms defined in ISO 10303-43:

— coordinate space;

— geometrically founded.

3.6 Terms defined in ISO 10303-44

This part of ISO 10303 makes use of the following terms defined in ISO 10303-44:

— bill-of-material structure (BOM);

— component;

— constituent;

— effectivity;

— link;

— lot;

— node;

— parts list structure;

— promissory use;

— root node;

— tree.

3.7 Terms defined in ISO 10303-202

This part of ISO 10303 makes use of the following terms defined in ISO 10303-202:

— application interpreted construct (AIC).

3.8 Other definitions

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

3.8.1 cam: a mechanical unit integral to a die set that is used to apply supplementary forming or shearing forces to sheet metal in directions not parallel to the direction of closure of the die set.

3.8.2 contractor: a person or an organization that awards a contract for the production of either a design for something, a plan for the production of something, or a physical object.

3.8.3 die: a single piece of metal that is brought into contact with sheet metal to form, impress, or shear it. A die that is used without a mating die (see 3.8.11) serves as a form against which sheet metal may be stretched, compressed, or hammered by tools other than dies. A die may also have one or more mating dies (see 3.8.11).

3.8.4 die face: those die surfaces that directly contact the sheet metal during the operation of the stamping press containing the die, plus the surfaces that provide contiguity to these contacting surfaces.

NOTES

1 - All die faces that are parts of dies belonging to die sets are enclosed within the die set when it is shut.

2 - Surfaces of the die face are commonly called active surfaces, addendum surfaces, and sculpted surfaces.

3.8.5 die set: a set of two or more mating dies (see 3.8.11) that act as a single tool to form, impress, or shear sheet metal when mounted in a stamping press (see 3.8.16).

NOTE - A die set may include zero or more cams.

3.8.6 die structure: the portions of a die that provide structural rigidity, support the weight and support the in-process stresses of the die.

NOTE - The die structure is essentially all portions of a die that are not part of the die face.

3.8.7 feature: an area of interest on an item that is described as a unit for some purpose. A feature serves as a means of calling attention to or attaching properties to any portion of an item design. The allowable items to which a feature applies are parts, materials or die components.

NOTES

1 - A portion is either a subset of the item design, or is the entire design.

2 - A feature may exist without feature properties attached to it. Such a feature may be noted because it is a subset of an item design that is frequently incorporated in other designs.

3.8.8 feature property: data that are associated with all or a portion of the design of an item. Feature properties are design intent, styling, material composition, method of manufacture, function, or physical properties.

3.8.9 hard aid: a physical model of a shape, part, or product.

NOTE - Hard aids may be made of wood, clay, plaster, or other materials. Hard aids are used when the source of design data are a physical model, as frequently occurs in styling or prototyping work.

3.8.10 in-process: a description of a process that has more than one manufacturing process before the item is completed.

3.8.11 mating die: a die that has a die face (see 3.8.4) that mates with a die face of another die.

3.8.12 part process plan: the detailed plan for the manufacture of a part. It includes a sequence of steps to be executed that call out the selected machines, fabrication, operation steps, setups, routings, and in-process inspection requirements employed to produce a part.

3.8.13 sheet metal: metal in a form in which the thickness is substantially less than the width or length.

NOTE - The maximum metal thickness permitted for metal to be considered sheet metal varies between industries. In the automotive industry, this maximum is 13 millimeters. In aerospace, it is 7 millimeters.

3.8.14 sheet metal part: a part that is not an assembly of other parts but that either is a product or is intended to be assembled into a product, and that is sheet metal or is fabricated from sheet metal (see 3.8.13).

3.8.15 sheet metal part process plan: a part process plan (see 3.8.12) for a sheet metal part (see 3.8.14).

3.8.16 stamping press: a machine that applies force to close and open a die set (see 3.8.5) on a sheet metal work piece in order to form it.

NOTE - Annex L describes stamping presses in more detail.

3.8.17 supplier: a person or organization responsible for producing that design, plan, or physical object under the terms of a contract.

3.9 Abbreviations

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

— AAM Application Activity Model;

— AIC Application Interpreted Construct;

— AIM Application Interpreted Model;

— AP Application Protocol;

— ARM Application Reference Model;

— ATS Abstract Test Suite;

— B-rep Boundry representation;

— CMM Coordinate Measuring Machine;

— ICOM Input, Control, Output, Mechanism;

— IDEF0 Integration Definition for Function Modeling;

— IDEF1X Integration Definition for Information Modeling;

— NC Numeric Control;

— OPT Optional

— PICS Protocol Implementation Conformance Statement;

— UoF Unit of Functionality.

4 Information requirements

This clause specifies the information required for sheet metal die planning and design.

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

NOTES

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

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

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

4.1 Units of functionality

This subclause specifies the units of functionality for the Sheet metal die planning and design application protocol. This part of ISO 10303 specifies the following units of functionality:

— feature;

— item;

— item_definition;

— item_definition_relationship;

— item_version;

— part_process_plan;

— product;

— representation_element;

— shape_definition;

— shape_definition_relationship;

— tolerance;

— work_order;

— work_request.

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 feature

The feature UoF represents areas of interest on an item that are described as a unit for some purpose. These areas of interest may serve as a means of calling attention to, or attaching properties to, all or a portion of the design of an item.

EXAMPLE 2 - When a die is intended to be loaded into a press using a crane, a hole feature may be designed in the die to accommodate the hook of the crane.

The following application objects are used by feature UoF:

— feature;

— feature_property.

4.1.2 item

The item UoF represents actual physical objects and their classifications.

The following application objects are used by item UoF:

— die;

— item;

— item_classification;

— item_classification_relationship;

— material;

— part.

4.1.3 item_definition

The item_definition UoF represents various designs of an item and its classifications.

The following application objects are used by item_definition UoF:

— designed_item;

— die_definition;

— external_item_reference;

— final_part_definition;

— in_process_part_definition;

— item_definition;

— material_definition;

— part_definition;

— scrap_definition.

4.1.4 item_definition_relationship

The item_definition_relationship UoF represents the relationship between the design of one item and the design of another item.

The following application objects are used by item_definition_relationship UoF:

— assembly_component_relationship;

— input_item_die_relationship;

— item_definition_relationship;

— make_similar_to_relationship;

— mating_relationship;

— specified_material_relationship;

— symmetrical_item_relationship.

4.1.5 item_version

The item_version UoF represents versions of items and distinguishes whether they are intended for manufacture internally or externally to the organization that designed them.

The following application objects are used by item_version UoF:

— item_version;

— made_in_house;

— purchased.

4.1.6 part_process_plan

The part_process_plan UoF represents part process plans and their constraints, versions, and component operations.

The following application objects are used by part_process_plan UoF:

— die_definition_constraint;

— part_process_plan;

— part_process_plan_template;

— part_process_plan_version_relationship;

— plant_constraint;

— press_definition;

— process_operation.

4.1.7 product

The product UoF represents the physical product and both where and when given parts are located within it.

The following application objects are used by product UoF:

— marketable_assembly;

— part_on_product;

— part_on_product_location;

— production_effectivity.

4.1.8 representation_element

The representation_element UoF contains representations of elements within overall shapes. These representations include mathematical and physical representations.

The following application objects are used by representation_element UoF:

— angle_distance_dimension;

— curve_distance_dimension;

— curve_length_size_dimension;

— diameter_size_dimension;

— dimensional_representation;

— distance_dimension;

— linear_distance_dimension;

— physical_representation;

— position_orientation_representation;

— radius_size_dimension;

— representation_element;

— size_dimension;

— solid_representation;

— surface_representation;

— wireframe_representation.

4.1.9 shape_definition

The shape_definition UoF represents the shapes of entire items or areas of interest on items that comprise less than entire items.

The following application objects are used by shape_definition UoF:

— die_shape_definition;

— feature_shape_definition;

— material_shape_definition;

— part_shape_definition;

— shape_definition.

4.1.10 shape_definition_relationship

The shape_definition_relationship UoF represents relationships between shapes.

The following application objects are used by shape_definition_relationship UoF:

— envelope_relationship;

— shape_definition_relationship;

— similarity_relationship.

4.1.11 tolerance

The tolerance UoF represents design allowable deviations on measurements and positions of elements within shapes.

The following application object is used by tolerance UoF:

— coincidence_defaults;

— dimension_tolerance_range;

— shape_tolerance;

— tolerance_bound;

— tolerance_value.

4.1.12 work_order

The work_order UoF represents work which has been assigned and all of the classifications of directives for that work to be done.

The following application objects are used by work_order UoF:

— change_order;

— external_order;

— internal_order;

— start_order;

— work_item;

— work_order;

— work_order_relationship;

— work_order_responsibility.

4.1.13 work_request

The work_request UoF represents the classifications of solicitations for work to be done.

The following application objects are used by work_request UoF:

— change_request;

— start_request;

— work_request.

4.2 Application objects

This subclause specifies the application objects for the sheet metal die planning and design application protocol. Each application object is an atomic element that embodies a unique application concept and contains attributes specifying the data elements of the object. The application objects and their definitions are given below.

4.2.1 Angle_distance_dimension

An Angle_distance_dimension is a type of Distance_dimension (see 4.2.16) where the distance is calculated between two elements converging on a common point or line, measured in units of angle measure. The data associated with an Angle_distance_dimension are the following:

— clockwise;

— orientation.

4.2.1.1 clockwise

The clockwise specifies whether or not the dimension is measured in a clockwise direction from the geometric basis element to the geometric target element.

4.2.1.2 orientation

The orientation specifies the vector that defines the right-hand-rule to be used in determining the direction of measurement specified by the clockwise attribute.

4.2.2 Assembly_component_relationship

An Assembly_component_relationship is a type of Item_definition_relationship (see 4.2.31) describing the immediate assembly context of every component and subassembly, and conversely, the component and subassembly composition of every assembly of a die or sheet metal part. The data associated with an Assembly_component_relationship is component_quantity.

The component_quantity specifies the number of components used in the construction of the assembly to which they belong.

NOTE - The item_definitions playing the roles of component and assembly must both be of the same item_-definition type, and must be either a "part_definition" or "die_definition".

4.2.3 Change_order

A Change_order is a type of Work_order (see 4.2.74) that requires modifications to a previous work effort. Items modified include part designs, die designs and part process plans. The data associated with a Change_-order are the following:

— change_design_location;

— implementor.

4.2.3.1 change_design_location

The change_design_location specifies the location of the aspect of a design or a step in a process plan that needs to be changed.

EXAMPLE 3 - When an aspect of a design or a step in a process plan is to be changed, the location of the intended change within the context of the overall design needs to be specified. One possible instance of this specification would be if the shape of a hole in the design of a sheet metal part needed to be changed. A circle might be drawn around the representation of that hole as it appeared on a paper plot of the overall part design. This circle would indicate the location where the Change_order change is intended to occur. Similary in a process plan if a trim die operation step needs to be replaced by draw die operation, the change might be indicated by highlighting the process operation step.

4.2.3.2 implementor

The implementor specifies the person, organization, or both responsible for implementing the change.

4.2.4 Change_request

A Change_request is a type of Work_request (see 4.2.77) that solicits change to some aspect of a part design, die design or process plan. The data associated with a Change_request are the following:

— problem_change_description;

— recommended_solution.

4.2.4.1 problem_change_description

The problem_change_description specifies a problem with a part design, die design or process plan. Problem change descriptions may be defined textually, graphically or a combination of both.

4.2.4.2 recommended_solution

The recommended_solution specifies how a change should be implemented to resolve a problem with a part design, die design or process plan. There may be more than one recommended_solution for a Change_request.

4.2.5 Coincidence_defaults

A Coincidence_defaults is a description of the limits of positional variance between two elements of a CAD model under which the two elements will be considered to be coincident. The data associated with a Coincidence_defaults are the following:

— position;

— angle.

EXAMPLE 4 - A model in a CAD System A consists of two points seperated by 1.0 mm. CAD System A requires a linear seperation between points to equal or exceed 0.1 mm in order to be considered non-coincident. Therefore CAD System A has a coincidence_defaults.postion equal to 0.1 mm. CAD System B has a greater coincidence_defaults.position, equal to 1.0 mm. Therefore if the model from CAD System A is transfered to CAD System B, the two points will be considered to be coincident there.

4.2.5.1 position

The position specifies the maximum linear separation condition between two points under which they will be considered to be coincident.

4.2.5.2 angle

The angle specifies the maximum angular separation condition between two elements under which they will be considered to be coincident.

4.2.6 Curve_distance_dimension

A Curve_distance_dimension is a type of Distance_dimension (see 4.2.16) where the distance is calculated between two elements along a path defined by a third element of geometry.

4.2.7 Curve_length_size_dimension

A Curve_length_size_dimension is a type of Size_dimension (see 4.2.63) where the element to be measured is a curve and the measurement is to be along the entire path of the curve.

4.2.8 Designed_item

A Designed_item is a type of Item_definition (see 4.2.30) that is a part design or die design that is created internal to a business organization rather than created externally. The data associated with a Designed_item are the following:

— creation_date_and_time;

— data_exchange_history;

— data_ownership;

— designer;

— generating_system_information;

— media_requirements.

EXAMPLE 5 - A standard fastener is an example of an externally designed item.

4.2.8.1 creation_date_and_time

The creation_date_and_time specifies the date and optionally the time that a Designed_item was created.

4.2.8.2 data_exchange_history

The data_exchange_history specifies the software, hardware, and sequence of all translations of the designed_-item since that or those translations specified by the generating_system_information. There may be more than one data_exchange_history for a Designed_item.

4.2.8.3 data_ownership

The data_ownership specifies all the organizations and associated personnel that have ownership of part or die design data. Included in this information is ownership history. There may be more than one data_ownership for a Designed_item.

4.2.8.4 designer

The designer specifies the person and organization that designed the item. The information includes names, addresses, and phone numbers of people and organizations who created the design of a part or die Item. There may be more than one designer for a Designed_item.

4.2.8.5 generating_system_information

The generating_system_information specifies a reference to a document, electronic file, or other source that specifies the software, hardware, and file names where the original part or die design information was generated. The original data set location, the name of the user who created it, and the creating organization, with all relevant addresses and phone numbers, are included. There may be more than one generating_system_-information for a Designed_item.

4.2.8.6 media_requirements

The media_requirements specifies the expected format and media to be used for the designed data information. There may be more than one media_requirements for a Designed_item.

EXAMPLE 6 - Media requirements may state whether information is to be plotted on paper or provided in a specified electronic format. A media_requirement might specify that the shape_definition of a die is to be presented in a proprietary corporate format while the bill of material should be presented on a specific paper form.

4.2.9 Diameter_size_dimension

A Diameter_size_dimension is a type of Size_dimension (see 4.2.63) where the element to be measured is circular and the measurement is the diametrical distance between two points on the element.

4.2.10 Die

A Die is a type of Item (see 4.2.27) that is a tool used singly, or in sets in a stamping press, to stretch, compress, or hammer sheet metal against in order to impart a desired shape to an object made of sheet metal. It may or may not include cams.

NOTE - Further explanation may be found in annex L.

4.2.11 Die_definition

A Die_definition is a type of Item_definition (see 4.2.30) that defines references that are specific to dies or their in-process or final-form components. The data associated with a Die_definition are the following:

— die_function_description;

— die_layout_specification_reference;

— die_structure_specification_reference;

— die_weight;

— pattern_casting_specification.

4.2.11.1 die_function_description

The die_function_description specifies the operation of a die assembly, subassembly or component.

4.2.11.2 die_layout_specification_reference

The die_layout_specification_reference specifies the particular manuals and sections within manuals that detail the methods and standards used to design the die face. There may be more than one die_layout_specification_-reference for a Die_definition.

4.2.11.3 die_structure_specification_reference

The die_structure_specification_reference specifies the particular manuals and sections within manuals that detail the methods and standards used to design the structural elements of dies. There may be more than one die_structure_specification_reference for a Die_definition.

4.2.11.4 die_weight

The die_weight specifies the weight of a die component, subassembly or fully assembled die set, and whether it is estimated, calculated, or measured.

4.2.11.5 pattern_casting_specification

The pattern_casting_specification specifies an already manufactured pattern or an existing casting design to be used to obtain the casting from which the die component will be machined. This attribute only applies to die components that are not assemblies.

NOTE - This application element is a reference to either a physical object or to a CAD design.

4.2.12 Die_definition_constraint

A Die_definition_constraint is a description of a restriction on how a die is defined. A Die_definition_-constraint is commonly a Plant_constraint (see 4.2.50), a Press_definition (see 4.2.52), or another kind of constraint as described in the constraint_description attribute of this entity. The data associated with a Die_definition_constraint is constraint_description.
The constraint_description specifies a description of the Die_definition_constraint and also specifies that a Die_definition_constraint is neither a Plant_constraint nor a Press_definition.

NOTES

1 - IDEF1X occasionally is incapable of modelling category relationships when the candidate generic entity and one or more of its candidate category entities participate in circular relationships. The IDEF1X terms generic and category correspond to supertype and subtype in EXPRESS terminology. A Die_definition_constraint is such a candidate generic entity as depicted in the ARM diagrams. Therefore it is modelled as an entity which "is" zero or one of each of the two entities which would have been its category entities. In addition, the note following has been added to the ARM diagram to ensure that the Die_definition_constraint is treated as a generic entity.

2 - Each instance of Die_definition_constraint is either a Press_definition, a Plant_constraint, or neither of them.

EXAMPLES

7 - Items such as plant crane capacity, material handling capabilities and plant layout restrict the manner in which a die may be designed.

8 - Examples of Die_definition_constraints_which are neither examples of Plant_constraints nor of Press_definitions include temperature or humidity variation where a die set will be installed, projected in-process stress on a die, and the coefficient of thermal expansion of the steel from which the die will be cast.

4.2.13 Die_shape_definition

A Die_shape_definition is a type of Shape_definition (see 4.2.59) where the shape being described is that of a die or die component.

4.2.14 Dimension_tolerance_range

A Dimension_tolerance_range is a range of values within which the realization of a Dimensional_representation may fall and still comply with the design specification.

NOTE - Instances of Dimension_tolerance_range either have a non-blank value for their tolerance_fitting_type attribute, or non_blank values for all of their associated Tolerance_value instances; but never do both situations occur simultaneously.

EXAMPLE 9 - The range of values can be specified in many ways. In sheet metal design of the active surfaces of a die or of the shape of the formed sheet metal, tolerances may often be implied by the number of digits given in the Dimensional_representation. For example, if 7.0 mm is specified, it is assumed that the actual measure may vary between 6.9 and 7.1 mm. If 7.00 mm were specified, the actual measure could vary between 6.99 and 7.01 mm. Other sheet metal designers use a similar form of implicit tolerance known as significant digits or rounding tolerance. These represent the range of values that would round to the specified measure. For example, if 7.0 mm is specified, it is assumed that the actual measure may vary between 6.95 and 7.04 mm. If 7.00 mm were specified, the actual measure could vary between 6.995 and 7.004 mm.

The data associated with a Dimension_tolerance_range is tolerance_fitting_type.

The tolerance_fitting_type specifies the class of hole and shaft fit. The tolerance_fitting_type need not be specified for a particular Dimension_tolerance_range.

4.2.15 Dimensional_representation

A Dimensional_representation is a type of Representation_element (see 4.2.57) in which the shape of an object is represented in terms of a series of dimensional units and values.

EXAMPLE 10 - The representation of a piece of sheet metal may be that it is 2 meters long by 1 meter wide by 3 millimeters thick.

4.2.16 Distance_dimension

A Distance_dimension is a type of Dimensional_representation (see 4.2.15) that is a calculation of the distance between two elements of geometry or annotation. Each Distance_dimension is either an Angle_distance_-dimension (see 4.2.1), a Curve_distance_dimension (see 4.2.6), or a Linear_distance_dimension (see 4.2.33).

4.2.17 Envelope_relationship

An Envelope_relationship is a type of Shape_definition_relationship (see 4.2.60) in which one Shape_definition plays the role of envelope for another Shape_definition. It is possible to have multiple Envelope_relationships for the same pair of Shape_definitions.

EXAMPLE 11 - A design may be developed within a constraint requiring it to have a Shape_definition that fits within a boundary. The boundary may also be defined as a Shape_definition. The relationship between the boundary Shape_definition and the design Shape_definition is an Envelope_relationship type of Shape_definition_-relationship

4.2.18 External_item_reference

An External_item_reference is a type of Item_definition (see 4.2.30) in which the definition of the item may be described in terms of or as an item that has been externally originated. An external item is an item that is typically procured rather than designed. The data associated with an External_item_reference are the following:

— manual_reference_description;

— name;

— needed_modifications.

4.2.18.1 manual_reference_description

The manual_reference_description specifies page and paragraph references within books and manuals that contain external item information.

4.2.18.2 name

The name specifies the name by which the external item is known.

4.2.18.3 needed_modifications

The needed_modifications specifies any changes that need to be made to the external item in order to make use

of the item. The needed_modification need not be specified for a particular External_item_reference. There may be more than one needed_modifications for an External_item_reference.

4.2.19 External_order

An External_order is a type of Work_order (see 4.2.74) in which the requested work will be done by an organization outside of the one placing the order. The data associated with an External_order are the following:

— contractual_requirement;

— purchase_order_number;

— purchasing_agent;

— supplier_identification.

4.2.19.1 contractual_requirement

The contractual_requirement specifies the terms of the contract between the contractor and the supplier responsible for carrying out the work outlined in the external_order. There may be more than one contractual_-requirement for an External_order.

4.2.19.2 purchase_order_number

The purchase_order_number specifies the External_order for the supplier.

4.2.19.3 purchasing_agent

The purchasing_agent specifies the people and associated departments or internal organizations responsible for procuring the services of a supplier to service the External_order. There may be more than one purchasing_agent for an External_order.

4.2.19.4 supplier_identification

The supplier_identification specifies the supplier who is servicing the External_order. The identification information includes the name, address, phone number and any other unique identification of a supplier business organization. There may be more than one supplier_identification for an External_order.

4.2.20 Feature

A Feature is an area of interest on an Item (see 4.2.27) that is described as a unit for some purpose. The data associated with a feature are the following:

— designer;

— feature_name.

EXAMPLE 12 - Possible Features include adhesive locations, beads, bends, body lines, breaklines, cutouts, dimples, feature lines, flange lines, holes, joggles, locating points, lock stops, machinable areas, master control surfaces, mold points, overcrowns, overdraws, periphery trim, sausages, slots, tooling holes, trim lines, weld spots, and work lines.

NOTE - The data feature_identification has been removed and will be mapped to the attribute identifier when it is available in ISO 10303-41.

4.2.20.1 designer

The designer specifies the designer of the particular feature being defined as part of an Item_definition.

4.2.20.2 feature_name

The feature_name specifies the class of Feature as a descriptive label.

4.2.21 Feature_property

The Feature_property is the description of a property and associated values that are attached to a feature of a part or die component. The data associated with a Feature_property are the following:

— property_description;

— property_type.

EXAMPLE 13 - Possible properties of features include skin quality, grain direction, surface finish, surface quality, heat treat information, finish coats, springback, coating properties, and key characteristics.

NOTE - The data feature_property_identification has been removed and will be mapped to the attribute identifier when it is available in ISO 10303-41.

4.2.21.1 property_description

The property_description specifies the property measurements associated with a Feature. This property description may be textual or may be a value with a specified unit. This property may or may not be intrinsic to a material.

4.2.21.2 property_type

A property_type specifies the class of property.

4.2.22 Feature_shape_definition

A Feature_shape_definition is a type of Shape_definition (see 4.2.59) in which the shape being described is that of a Feature. The Feature_shape_definition may describe the shape of a Feature on a part whose shape is defined by a Part_shape_definition (see 4.2.48).

4.2.23 Final_part_definition

A Final_part_definition is a type of Part_definition (see 4.2.42) that is the output of the final die operation performed on the part prior to any assembly work.

4.2.24 In_process_part_definition

An In_process_part_definition is a type of Part_definition (see 4.2.42) in which the part being defined is of an in-process sheet metal part that is intended to be operated upon by a manufacturing process. A series of one or more manufacturing processes acting upon one or more in-process part definitions will produce as the final output a part defined by a Final_part_definition (see 4.2.23). As such, an in_process_part_definition is the design of a sheet metal part in terms of the requirements for operation by a subsequent manufacturing process.

4.2.25 Input_item_die_relationship

An Input_item_die_relationship is a type of Item_definition_relationship (see 4.2.31) that describes a relationship between input parts or materials or scrap and the die that modifies them. The input part or material or scrap is the in-process part or raw material or by-product of a previous process that will be stamped by a die in a press operation. This relationship also describes the positioning and orientation of the item within the die. The item_definition playing the role of input_item shall be of item_definition_type "part_definition", "material_definition", or "scrap_definition", and that playing the role of die must be of item_definition_type "die_definition".

4.2.26 Internal_order

An Internal_order is a type of Work_order (see 4.2.74) in which the work described by the Work_order is planned to be done within an organization. The data associated with an Internal_order are the following:

— authorized_hours;

— charge_number;

— source_department.

4.2.26.1 authorized_hours

The authorized_hours specifies the maximum number of man hours that will be allocated for completing the terms of the Internal_order.

4.2.26.2 charge_number

The charge_number specifies a project against which the hours or cost of completing the Work_order may be charged.

4.2.26.3 source_department

The source_department specifies the group that drafted and requested the Internal_order. Information describing the source_department includes any organization identifiers, the name of the department, department address, and department phone number. There may be more than one source_department for an Internal_order.

4.2.27 Item

An Item is an object in the scope of the subject of sheet metal that is, or is intended to be, produced. An Item may be one of the following: a Die (see 4.2.10), a Material (see 4.2.37), or a Part (see 4.2.41). The data associated with an Item are the following:

— item_description;

— item_name;

— item_number.

4.2.27.1 item_description

The item_description specifies the description of an object in the scope of the subject of sheet metal.

4.2.27.2 item_name

The item_name specifies the name of an object in the scope of the subject of sheet metal.

4.2.27.3 item_number

The item_number specifies a unique identification of a manufactured object in the scope of the subject of sheet metal.

4.2.28 Item_classification

An Item_classification specifies a categorization of an Item. This classification is intended to be used so that designers may easily reference similar designs done in the past. The data associated with an Item_classification are the following:

— classification_description;

— classification_identification;

— classification_name.

4.2.28.1 classification_description

The classification_description specifies a human interpretable definition of the Item_classification.

4.2.28.2 classification_identification

The classification_identification specifies a unique identification of the Item_classification.

4.2.28.3 classification_name

The classification_name specifies a meaningful name for a categorization of Items.

EXAMPLE 14 - A classification_name may be deep drawn die assemblies, die shoes, or brackets.

4.2.29 Item_classification_relationship

An Item_classification_relationship is a hierarchical association of two Item_classification objects. The data associated with an Item_classification_relationship is relationship_description.

The relationship_description specifies the nature of the association between the two Item_classification objects.

4.2.30 Item_definition

An Item_definition is a type of Work_item (see 4.2.73) that is the complete description of an assembly, component, scrap, or raw material that is part or all of a sheet metal part, the scrap that results from an operation, or a stamping die. An Item_definition may be one of the following: either a Designed_item (see 4.2.8) or an External_item_reference (see 4.2.18). An Item_definition may also be one of the following: either a Die_definition (see 4.2.11), a Material_definition (see 4.2.38), a Part_definition (see 4.2.42), or a Scrap_-definition (see 4.2.58). This definition includes all relevant properties of the defined item such as item relationships, the physical shape of the Item, and Feature characteristics. The data associated with an Item_-definition are the following:

— approval;

— approval_status;

— definition_description;

— procurement_information;

— proprietary_security_usage_information.

NOTE - The data item_definition_identification has been removed and will be mapped to the attribute identifier when it is available in ISO 10303-41.

4.2.30.1 approval

An approval specifies all relevant signoffs and signoff dates by respective management and technical personnel to indicate that the Item_definition has met certain requirements from their perspectives. There may be more than one approval for an Item_definition.

EXAMPLE 15 - The requirements of a person providing an approval may include conformity to: drafting techniques, contractual obligations, quality, or a standard.

4.2.30.2 approval_status

An approval_status specifies the status of the last approval for an Item_definition.

4.2.30.3 definition_description

The definition_description specifies a description of the physical item being defined.

4.2.30.4 procurement_information

The procurement_information specifies the information required to procure items needed as part of the design, or to procure externally originating items. This information includes the name, address, phone number, and contact personnel of the vendor supplying the item. This vendor may be an in-house department. There may be more than one procurement_information for an Item_definition.

4.2.30.5 proprietary_security_usage_information

The proprietary_security_usage_information specifies a description of the degree to which item definition data may be shared and used within and external to an organization. This information would include any limitations on distribution of data, limitations on who may access the data, and limitations as to what the data may be used for. There may be more than one proprietary_security_usage_information for an Item_definition.

4.2.31 Item_definition_relationship

An Item_definition_relationship is a specific type of relationship between one Part_definition (see 4.2.42), Die_definition (see 4.2.11), Material_definition (see 4.2.38), or Scrap_definition (see 4.2.58) and another Part_definition, Die_definition, Material_definition, or Scrap_definition. An Item_definition_relationship may be one of the following: an Assembly_component_relationship (see 4.2.2), an Input_item_die_relationship (see 4.2.25), a Make_similar_to_relationship (see 4.2.35), a Mating_relationship (see 4.2.40), a Specified_material_relationship (see 4.2.65), or a Symmetrical_item_relationship (see 4.2.69). An Item_definition_relationship is an association of two Item_definitions which define Items which may be assembled together, act upon each other, be similar to each other, mate or interface with each other, be symmetrical to each other, or where one item is used to make the other. It is possible to have multiple relationships between the same pair of part, die, material, or scrap Item_definitions provided that each of the relationships is of a different relationship type. The data associated with an Item_definition_relationship is relationship_description.

The relationship_description specifies a definition of the manner in which two Item_definitions relate to each other. The description shall be defined in a manner that is specific to the type of relationship being described.

EXAMPLE 16 - A symmetrical item relationship_description will be a description of the manner in which two parts are symmetrical to each other with respect to a point, line, or plane.

4.2.32 Item_version

An Item_version is a type of Work_item (see 4.2.73) that is a particular variant of an object in the scope of the subject of sheet metal that is, or is intended to be, produced. An Item_version may be one of the following: either a Purchased (see 4.2.55) or a Made_in_house (see 4.2.34). The data associated with an Item_version are the following:

— approval;

— approval_status;

— description;

— item_version_identification;

— revision_date_and_time.

4.2.32.1 approval

An approval specifies all relevant signoffs and signoff dates by respective management and technical personnel to indicate that the Item_version has met certain requirements from their perspectives. There may be more than one approval for an Item_version.

4.2.32.2 approval_status

An approval_status specifies the status of the last approval for an Item_version.

4.2.32.3 description

The description specifies a description of the version of an object in the scope of the subject of sheet metal that is being identified by the Item_version.

4.2.32.4 item_version_identification

The item_version_identification specifies uniquely the Item_version.

4.2.32.5 revision_date_and_time

A revision_date_and_time specifies the date and optionally the time that the Item_version was revised.

4.2.33 Linear_distance_dimension

A Linear_distance_dimension is a type of Distance_dimension (see 4.2.16) where the distance is calculated between two elements along a linear path.

NOTE - The Linear_distance_dimension can be used for the horizontal measurement between two points by specifying a direction vector parallel to the x-axis of the coordinate system.

4.2.34 Made_in_house

A Made_in_house is a type of Item_version (see 4.2.32) manufactured internal to an organization that designs the Item.

4.2.35 Make_similar_to_relationship

A Make_similar_to relationship is a type of Item_definition_relationship (see 4.2.31) between two Item_definitions where one Item_definition was defined similarly to a second Item_definition.

EXAMPLES

17 - A designer may wish to maintain a record of the association between a current design for a die punch and the design of a die punch used on an otherwise unrelated project several years before, because he wishes to be able to justify the current design's unusual characteristics as being based on the reliability of the previous design.

18 - A Make_similar_to_relationship only models relationships between Item_definitions. Item_definition_-relationships may or may not have their shapes characterized, because purely textual descriptions are allowed, or the level of definition of the Items involved may not have reached the level of geometrical design at a given time. This entity models the fact that designers occasionally base a design on a previous design. Questions about the later design may be answered by referring to records of the earlier design. Therefore the similarity being modelled by this entity may or may not be based on shape, but it is being recorded on the basis of the Item_definition context of that similarity. This distinguishes Make_similar_to_relationship from Similarity_relationship.

4.2.36 Marketable_assembly

A Marketable_assembly is a component or assembly that is, or includes, one or more stamped sheet metal objects. Such components and assemblies include automobiles, airplanes and computer chassis. The data associated with a Marketable_assembly are the following:

— model_identification;

— production_year;

— style_identification.

4.2.36.1 model_identification

The model_identification specifies the specific model of a marketable_assembly within a style categorization of Marketable_assemblies.

EXAMPLE 19 - In the automotive business the model identification might identify a Pontiac Bonneville, that is one of the Marketable_assembly types within the A style category.

4.2.36.2 production_year

The production_year specifies the year in which a Marketable_assembly is or is intended to be manufactured.

4.2.36.3 style_identification

The style_identification specifies a particular classification of Marketable_assemblies based on certain features thereof.

EXAMPLE 20 - In the automotive business the style is often a categorization of Marketable_assemblies based on common shapes of the car body among several different vehicles.

4.2.37 Material

A Material is a type of Item (see 4.2.27) that identifies the raw material of which a part or die Item is made.

4.2.38 Material_definition

A Material_definition is a type of Item_definition (see 4.2.30) defining aspects that are specific to the materials of which sheet metal parts and dies are made. The data associated with a Material_definition are the following:

— material_description;

— material_specification.

4.2.38.1 material_description

The material_description specifies the materials of which sheet metal parts, dies, and die components are made.

4.2.38.2 material_specification

The material_specification specifies the callout or identification of a specific material via references to a book or manual that contains a listing of raw materials.

4.2.39 Material_shape_definition

A Material_shape_definition is a type of Shape_definition (see 4.2.59) where the shape being described is that of a material.

4.2.40 Mating_relationship

A Mating_relationship is a type of Item_definition_relationship (see 4.2.31) used for the description of the relationship between mating components in an assembly or subassembly of a part or die Item. This relationship specifically defines how mating parts interface and any associated shape characteristics of the interface. In the Mating_relationship the types of Items that are related must be of the same type. Thus part Items may not have a mating relationship with die Items and vice versa.

NOTE - The item_definition playing the role of mating and the item_definition playing the role of mated must both be of the same item_definition_type, must be of item_definition_type "part_definition" or "die_definition", and must be at the same level with respect to the next higher level of assembly. This distinguishes this relationship type from assembly_component_relationship.

4.2.41 Part

A Part is a type of Item (see 4.2.27) identifying a sheet metal product or an in-process sheet metal Part Item.

4.2.42 Part_definition

A Part_definition is a type of Item_definition (see 4.2.30) that further defines aspects that are specific to sheet metal parts. A Part_definition is either a Final_part_definition (see 4.2.23) or an In_process_part_definition (see 4.2.24).

4.2.43 Part_on_product

A Part_on_product is the relationship between a product such as an assembly, aircraft, or automobile, and a sheet metal part that is placed on that product.

4.2.44 Part_on_product_location

A Part_on_product_location is the location of a sheet metal part with respect to the Marketable_assembly on which it will be placed. The data associated with the Part_on_product_location is intraproduct_location.

The intraproduct_location specifies where a specific sheet metal part is placed relative to a Marketable_-assembly.

4.2.45 Part_process_plan

A Part_process_plan is a type of Work_item (see 4.2.73) that is the description of the specific process steps needed to manufacture a sheet metal Part. The data associated with a part_process_plan are the following:

— approval;

— approval_status;

— creation_date_and_time;

— data_ownership;

— generating_system_information;

— part_process_version_identification;

— plan_status;

— planner;

— production_rate;

— proprietary_security_usage_information;

— review_date_and_time;

— version_description.

4.2.45.1 approval

An approval specifies all relevant signoffs and signoff dates by respective management and technical personnel to indicate that the plan for the manufacturing process of a sheet metal part has met certain requirements from their perspectives. There may be more than one approval for a Part_process_plan.

4.2.45.2 approval_status

An approval_status specifies the status of the last approval for a Part_process_plan.

4.2.45.3 creation_date_and_time

The creation_date_and_time specifies the date and optionally the time that indicates when the Part_process_-plan was created.

4.2.45.4 data_ownership

The data_ownership specifies a breakdown of all the organizations and associated personnel that have ownership rights in Part_process_plan data. Included in this information is ownership history, all ownership rights and other related proprietary statements. There may be more than one data_ownership for a Part_-process_plan.

4.2.45.5 generating_system_information

The generating_system_information specifies a reference to an external document that specifies the software, hardware and file names where the original process plan information was generated. Included in this information is the user who created the data and the creating organization with all relevant addresses and phone numbers. The location of the original data set is also included in this information. There may be more than one generating_system_information for a Part_process_plan.

4.2.45.6 part_process_version_identification

The part_process_version_identification specifies the unique Part_process_plan.

4.2.45.7 planner

The planner specifies the person and related organization responsible for constructing the Part_process_plan. There may be more than one planner for a Part_process_plan.

4.2.45.8 plan_status

The plan_status specifies the degree to which the plan has been accepted. The statuses that may be specified include released, pre-released, and implemented.

4.2.45.9 production_rate

The production_rate specifies the frequency of sheet metal part production based on the part process plan.

4.2.45.10 proprietary_security_usage_information

The proprietary_security_usage_information specifies the degree to which Part_process_plan data may be shared and used within and external to an organization. This information would include any limitations of distribution of data, limitations on who has access to the data and limitations as to what the data may be used for. There may be more than one proprietary_security_usage_information for a Part_process_plan.

4.2.45.11 review_date_and_time

The review_date_and_time specifies the date and optionally the time information that indicate when the Part_-process_plan was reviewed.

4.2.45.12 version_description

The version_description specifies the general process that is being followed to produce a particular sheet metal Part.

4.2.46 Part_process_plan_template

A Part_process_plan_template is a template for outlining the general steps that are required to manufacture a sheet metal part.

NOTE - The data standard_plan_identification has been removed and will be mapped to the attribute identifier when it is available in ISO 10303-41.

4.2.47 Part_process_plan_version_relationship

A Part_process_plan_version_relationship is a relationship between a Part_process_plan and a subsequent Part_process_plan based upon it. The data associated with a Part_process_plan is revision_reason.

The revision_reason specifies the reason that the subsequent Part_process_plan was created from the original Part_process_plan. There may be more than one revision_reason for a Part_process_plan_version_-relationship.

4.2.48 Part_shape_definition

A Part_shape_definition is a type of Shape_definition (see 4.2.59) where the shape being described is that of a sheet metal part.

4.2.49 Physical_representation

A Physical_representation is a type of Representation_element (see 4.2.57) containing the identification and description of a hard aid that is used to represent the shape of an object. Hard aids include such things as wooden models, plaster models and clay models of the shapes of objects. The data associated with a physical representation are the following:

— hard_aid_description;

— hard_aid_identification;

— hard_aid_type.

4.2.49.1 hard_aid_description

The hard_aid_description specifies the hard aid in terms of its properties and characteristics.

4.2.49.2 hard_aid_identification

The hard_aid_identification specifies uniquely a physical model that represents a shape. The unique identification may indicate a location and physically stamped identification number.

4.2.49.3 hard_aid_type

The hard_aid_type specifies the type of hard aid being used for the Physical_representation.

EXAMPLE 21 - Hard aid types include plaster, wood, and clay.

4.2.50 Plant_constraint

A Plant_constraint is a type of Die_definition_constraint (see 4.2.12) placed on the manufacture of stamped sheet metal parts based on a capability or requirement of the facilities and organization responsible for producing the parts. The data associated with a Plant_constraint are the following:

— constraint_description;

— constraint_type;

— constraint_value;

— plant_identification.

NOTE - The data plant_constraint_identification has been removed and will be mapped to the attribute identifier when it is available in ISO 10303-41.

4.2.50.1 constraint_description

The constraint_description specifies the nature of a constraint.

EXAMPLE 22 - An example of a constraint_description could be "die weight maximum" when referring to a Plant_constraint in which a constraint_value could be "10 tons" and the constraint_type could be "plant crane capacity". Together these attributes define a Plant_constraint in which die weights cannot exceed 10 tons due to crane capacity.

4.2.50.2 constraint_type

The constraint_type specifies the kind of restriction with regard to manufacturing sheet metal parts that a plant has. There may be more than one constraint_type for a Plant_constraint.

EXAMPLE 23 - Constraint_types include crane capacity, automation equipment restrictions, press capabilities, plant layout restrictions, and die material requirements. See also example 22.

4.2.50.3 constraint_value

The constraint_value specifies the measure or value of a restriction with regard to manufacturing sheet metal parts that a plant has. There may be more than one constraint_value for a Plant_constraint.

EXAMPLE 24 - See example 22.

4.2.50.4 plant_identification

A plant_identification specifies the plant in which the constraints regarding the Part_process_plan exist.

4.2.51 Position_orientation_representation

A Position_orientation_representation is a type of Representation_element (see 4.2.57) that describes the orientation of objects relative to each other in a given space, or establishes the context of a single object relative to a space.

NOTE - A Position_orientation_representation may only be used to represent Shape_definitions which are in relationships with subtypes of Item_definition_relationship.

EXAMPLE 25 - A special case example is mirroring, in which an object is oriented as a mirror image across a given plane or line of symmetry relative to another object.

4.2.52 Press_definition

A Press_definition is a type of Die_definition_constraint (see 4.2.12) that describes the device that is used in a process operation to cause a die or dies to form a sheet metal part, and includes any automated devices required to manipulate the sheet metal within that device. This description includes the attributes of this device and its interior handling equipment.

NOTE - Every Press_definition participates in at least one of its non-identifying relationships.

EXAMPLE 26 - Some dies require mounting within a press in order to be used to impart shape to sheet metal; the press is this device in such cases. Other dies may not require a press, but must have the sheet metal mounted in a clamp that is then used to stretch the metal over the die; the clamp is this device in such cases. Some dies need neither a press nor clamps because they are used as a form over which to manually hammer sheet metal.

The data associated with a press definition are the following:

— automation_equipment;

— press_forces;

— press_identification;

— press_line;

— press_location;

— press_size;

— press_specification_reference;

— press_type.

4.2.52.1 automation_equipment

The automation_equipment specifies all automated equipment associated with the press. There may be more than one automation_equipment in a Press_definition.

EXAMPLE 27 - Typical types of automation_equipment that are attached or used with a press may include iron hands, robots, suction grippers, and metal gain devices.

4.2.52.2 press_forces

The press_forces specifies the amount of force that a particular press may apply to a sheet metal part through each of its one or more rams. There may be more than one press_forces for a Press_definition.

4.2.52.3 press_identification

The press_identification specifies uniquely a physical sheet metal stamping press.

4.2.52.4 press_line

The press_line specifies the particular fabrication plant and fabrication line in which the press is or is intended to be located.

4.2.52.5 press_location

The press_location specifies the physical location of the defined press in terms of its position within a press line, or, if there is no press line, in terms of its position within a plant.

4.2.52.6 press_size

The press_size specifies the main dimensional features of a given press, including length, width, and height of the overall press, and the shut height. There may be more than one press_size for a Press_definition.

NOTE - Shut height is the maximum height of die which a press can accomodate.

4.2.52.7 press_specification_reference

The press_specification_reference specifies a book or manual that contains the full specification and description of the defined press.

4.2.52.8 press_type

The press_type specifies the type of press.

EXAMPLE 28 - Press_types could include a single action press, double action press, and several others.

4.2.53 Process_operation

A Process_operation is the description of a single operation that is involved in the manufacture of a sheet metal part. Included in this description is the operation production rate and references to the machines and tools used in the manufacture and design of the sheet metal input for the operation. The data associated with a Process_-operation are the following:

— operation_description;

— operation_sequence_number;

— reference_specifications;

— required_shut_height;

— required_stamping_forces;

— scrap_percentage;

— units_per_operation.

4.2.53.1 operation_description

The operation_description specifies the type of and procedure of a single operation that is part of the manufacturing process of a sheet metal part.

EXAMPLE 29 - Operation parameters such as die temperature may be included.

4.2.53.2 operation_sequence_number

The operation_sequence_number specifies the particular sequence of an operation relative to a series of operations that produce a final sheet metal part.

4.2.53.3 reference_specifications

The reference_specifications specifies processing standards and guidelines that are referenced as part of the description of an individual process operation. There may be more than one reference_specifications for a Process_operation.

4.2.53.4 required_shut_height

The required_shut_height specifies the minimum shut height permissible in a Press_definition capable of fulfilling the requirements of a Process_operation.

4.2.53.5 required_stamping_forces

The required_stamping_forces specifies the minimum required amount of force that a press must apply to a sheet metal part through each of its one or more rams in order to be capable of fulfilling the requirements of a Process_operation. There may be more than one required_stamping_forces for a Process_operation.

4.2.53.6 scrap_percentage

The scrap_percentage specifies the percentage of the material that is input to the Process_operation that will be scrap upon completion of that Process_operation.

4.2.53.7 units_per_operation

The units_per_operation specifies the number of sheet metal items produced within one occurrence of a Process_operation.

4.2.54 Production_effectivity

A Production_effectivity is the description of the production and design conditions under which a particular part item should be used relative to the manufacture of a Marketable_assembly. The data associated with a Production_effectivity are the following:

— date_and_time_effectivity;

— effectivity_identification;

— sequence_effectivity.

NOTE - Either date_and_time_effectivity or sequence_effectivity must be specified, but not both.

4.2.54.1 date_and_time_effectivity

The date_and_time_effectivity specifies the timeframe under which a specific sheet metal part or die or version of a part or die is to be used. The timeframe typically specifies that as of a certain date and time, a certain item should be used in a certain configuration. Sometimes there is a date and time of expiration of the effectivity. There may be more than one date_and_time_effectivity for a Production_effectivity.

4.2.54.2 effectivity_identification

The effectivity_identification specifies a unique production_effectivity.

4.2.54.3 sequence_effectivity

A sequence_effectivity specifies particular sequence numbers of a Marketable_assembly, assembly or subassembly of a part or die for which a specific item should be used. There may be more than one sequence_-effectivity for a Production_effectivity.

4.2.55 Purchased

A Purchased is a type of Item_version (see 4.2.32) procured from an organization external to the organization that designs the Item. The data associated with a Purchased is purchase_requirements.

The purchase_requirements specifies the conditions under which the purchase must be made. There may be more than one purchase_requirements for a Purchased.

4.2.56 Radius_size_dimension

A Radius_size_dimension is a type of Size_dimension (see 4.2.63) where the element is circular and the dimension is from the centre of the element to any point thereon.

4.2.57 Representation_element

The Representation_element is an object that is used to mathematically or physically represent position, orientation or shape. A Representation_element is either a Dimensional_representation (see 4.2.15), a Physical_representation (see 4.2.49), a Position_orientation_representation (see 4.2.51), a Solid_representation (see 4.2.64), a Surface_representation (see 4.2.68), or a Wireframe_representation (see 4.2.72).

NOTE - The data representation_element_identification has been removed and will be mapped to the attribute identifier when it is available in ISO 10303-41.

4.2.58 Scrap_definition

A Scrap_definition is a type of Item_definition (see 4.2.30) that describes the excess pieces of metal that are cut from a part during a Process_operation. Such pieces may be used in another Process_operation.

4.2.59 Shape_definition

The Shape_definition is the physical shape of some object, or of a feature of an object, of interest. A Shape_-definition may be one of the following: a Die_shape_definition (see 4.2.13), a Feature_shape_definition (see

4.2.22), a Material_shape_definition (see 4.2.39), a Part_shape_definition (see 4.2.48), or a Tolerance_bound (see 4.2.70). Objects of interest include both components and assemblies of sheet metal parts and dies. The data associated with a Shape_definition are the following:

— shape_description.

NOTES

1 - Each instance of shape_definition participates in exactly one of Shape_definition's non_identifying relationships.

2 - The data shape_definition_identification has been removed and will be mapped to the attribute identifier when it is available in ISO 10303-41.

4.2.59.1 shape_description

The shape_description specifies a general textual definition of the shape of an object.

4.2.60 Shape_definition_relationship

A Shape_definition_relationship is a specific type of relationship between one Shape_definition and another Shape_definition. A Shape_definition_relationship may be one of the following: an Envelope_relationship (see 4.2.17) or a Similarity_relationship (see 4.2.62). It is possible to have more than one relationship between the same pair of Shape_definitions. The data associated with a Shape_definition_relationship is relationship_-description.

The relationship_description is a precise definition of the manner in which two Shape_definitions relate to each other. The description provided will be defined in a manner that is specific to the type of relationship being described.

EXAMPLE 30 - An Envelope_relationship will contain a relationship_description describing the purpose of the envelope.

4.2.61 Shape_tolerance

A Shape_tolerance is a specification of how precisely a shape feature of a Designed_item shall be maintained when the item is manufactured. The data associated with a Shape_tolerance are the following:

— tolerance_condition;

— tolerance_datum;

— tolerance_type.

4.2.61.1 tolerance_condition

The tolerance_condition specifies whether the Shape_tolerance applies to a maximum material condition, a minimum material condition or other similar property.

4.2.61.2 tolerance_datum

The tolerance_datum specifies a reference position against which the Shape_tolerance is to be compared. There may be more than one tolerance_datum for a Shape_tolerance.

4.2.61.3 tolerance_type

The tolerance_type specifies the kind of Shape_tolerance being applied.

EXAMPLE 31 - The kinds of Shape_tolerance include cylindrical tolerance, perpendicularity tolerance, runout tolerance, angular tolerance, linear tolerance, and circular tolerance.

4.2.62 Similarity_relationship

A Similarity_relationship is a type of Shape_definition_relationship (see 4.2.60) that specifies any similarity between Shape_definitions that is useful to maintain a record of.

EXAMPLE 32 - A Similarity_relationship only models relationships between Shape_definitions. A Similarity_-relationship always is a relationship between shapes. These shapes may or may not be related to Item_definitions. They may, for example, be between shapes of features. Or they may be between shapes that represent the interfaces of two separate Mating_relationships. Or they may be between shapes that represent the shape of a component in an assembly for an Assembly_component_relationship. Or they may be between shapes that describe the position of the input to a die with respect to a die. These uses distinguish Similarity_relationship from Make_similar_to_-relationship.

4.2.63 Size_dimension

A Size_dimension is a type of Dimensional_representation (see 4.2.15) that is a characteristic of size for an individual dimensional_representation. Each Size_dimension is either a Curve_length_size_dimension (see 4.2.7), a Diameter_size_dimension (see 4.2.9), or a Radius_size_dimension (see 4.2.56).

NOTE - A chordal length is a measurement of size, although treated as a distance between two points on the circular arc.

EXAMPLE 33 - Characteristics of size for circular arc geometry are radius, diameter, and arc length.

4.2.64 Solid_representation

A Solid_representation is a type of Representation_element (see 4.2.57) that represents shape in several possible ways. In constructive solid geometry (CSG) solids are represented in terms of boolean trees of operations on three-dimensional primitive polyhedra. In boundary representations (B-reps), solids are represented by the shells that describe the surface of the solid; and these shells consist of one or more surfaces that are topologically connected to each other. The data associated with a Solid_representation are the following:

— B-rep;

— CSG.

4.2.64.1 B-rep

The B-rep specifies a topological construction of base elements to represent the solid shape of an object. Specifically, a B-rep is composed of shells that bound a solid region of space. A shell is composed of faces. Faces are constructed from base surfaces and bounding loops. Loops are composed of edges. Edges are composed of base curves and bounding vertexes. A vertex asserts itself based on a specific point. There may be more than one B-rep for a Solid_representation.

4.2.64.2 CSG

The CSG specifies a solid geometry model composed of a tree structure of boolean operations upon solid primitives.

4.2.65 Specified_material_relationship

A Specified_material_relationship is a type of Item_definition_relationship (see 4.2.31) describing the relationship between a part or component and the raw material from which the part or die component is manufactured. The data associated with a Specified_material_relationship is material_quantity.

The material_quantity specifies the amount of a raw material to be used in creating a part or die component.

4.2.66 Start_order

A Start_order is a type of Work_order (see 4.2.74) that initiates new work on a Work_item.

4.2.67 Start_request

A Start_request is a type of Work_request (see 4.2.77) that solicits initiation of a part design, die design or a process plan. The data associated with a Start_request are the following:

— request_description;

— request_justification.

4.2.67.1 request_description

The request_description specifies text or a graphical description of the new part design, die design, or process plan that is being requested.

4.2.67.2 request_justification

The request_justification specifies the reason a Start_request was initiated.

4.2.68 Surface_representation

A Surface_representation is a type of Representation_element (see 4.2.57) that represents shape in terms of two-dimensional geometry. This two-dimensional geometry is not necessarily planar. The data associated with a Surface_representation are the following:

— surface_direction;

— surface_thickness.

4.2.68.1 surface_direction

The surface_direction specifies the direction from a Surface_representation of a sheet metal part design in which the material of the sheet metal part is designed to lie.

EXAMPLE 34 - Imagine a hemisphere that serves as the Surface_representation of a bowl that is to be made from sheet metal. If one intends to form a bowl so that the inner surface of the bowl is represented by the hemisphere, the surface_direction would be represented by a direction coincident with a ray whose origin is the center of the hemisphere. If one intends to form a bowl so that the outer surface of the bowl is represented by the hemisphere, the surface_direction would be oriented oppositely.

4.2.68.2 surface_thickness

The surface_thickness specifies the thickness of the metal represented by a Surface_representation.

4.2.69 Symmetrical_item_relationship

A Symmetrical_item_relationship is a type of Item_definition_relationship (see 4.2.31) between two Item_definitions where the two related parts, or two related die components, have shape characteristics that are related in a defined symmetrical manner.

NOTE - The item_definitions playing the roles of right and left must both be of item_definition_type "part_definition", or both of item_definition_type "die_definition".

EXAMPLE 35 - This relationship could indicate the correspondence between a right hand and left hand part of a product.

4.2.70 Tolerance_bound

A Tolerance_bound is a type of Shape_definition (see 4.2.59) that specifies all or part of the boundary of an area or volume within which a manufactured shape must exist in order to comply with its design tolerance.

EXAMPLE 36 - If a cylindrical portion of a designed object has a cylindricity tolerance, the cylinder must be manufactured to fit within an imaginary larger cylinder and an imaginary smaller cylinder. Each of these imaginary cylinders is a Tolerance_bound.

4.2.71 Tolerance_value

A tolerance_value specifies a length or angle measurement that is a bound of a Dimension_tolerance_range. The data associated with a Tolerance_value are the following:

— unit_of_measure;

— value.

EXAMPLE 37 - If a dimension of a designed object is 2 meters plus or minus .1 meters, there are two Tolerance_values for this dimension. One Tolerance_value has a value of 2.1; the other has a value of 1.9. Both Tolerance_values have a unit_of_measure of "meter".

4.2.71.1 unit_of_measure

The unit_of_measure specifies the length units with which a Tolerance_value is to be expressed.

4.2.71.2 value

The value specifies the quantity with which a Tolerance_value is to be expressed.

4.2.72 Wireframe_representation

A Wireframe_representation is a type of Rrepresentation_element (see 4.2.57), that represents shape by mathematically depicting a sketch of the edges of an item. This representation gives a general outline of the shape of an object, but does not clarify either the exact outer or inner surface of an object or delineate the solid shape portion of an object. In addition to representing aspects of shapes, the Wireframe_representation includes information on representing the orientation of objects.

4.2.73 Work_item

A Work_item is an object to which work has been assigned. A Work_item may be one of the following: an Item (see 4.2.27), an Item_definition (see 4.2.30), or a Part_process_plan (see 4.2.45). The types of objects to which work may be assigned includes Part_process plans, Item_versions and Item_definitions. The data associated with a Work_item is the following:

— applicable_standards;

— approval;

— completion_date_and_time;

— order_date_and_time;

— preliminary_review_date_and_time;

— priority;

— start_date_and_time;

— work_description;

— work_requirements.

NOTES

1 - IDEF1X occasionally is incapable of modelling category relationships when the candidate generic entity and one or more of its candidate category entities participate in circular relationships. The IDEF1X terms generic and category correspond to supertype and subtype in EXPRESS terminology. A Work_item is such a candidate generic entity as depicted in the ARM diagrams. Therefore it is modelled as an entity which "is" zero or one of each of the three entities which would have been its category entities. In addition, the note following has been added to the ARM diagram to ensure that the Work_item is treated as a generic entity.

2 - An instance of Work_item participates in zero or one of its three non-identifying relationships at a time.

3 - The data work_item_identification has been removed and will be mapped to the attribute identifier when it is

available in ISO 10303-41.

4.2.73.1 applicable_standards

The applicable_standards specifies the callout of standards and specifications that must or should be used in carrying out the work described by the Work_item. There may be more than one applicable_standards for a Work_item.

4.2.73.2 approval

An approval specifies all relevant signoffs and signoff dates by respective management and technical personnel to indicate that the Work_item has met certain requirements from their perspectives. There may be more than one approval for a Work_item.

4.2.73.3 completion_date_and_time

The completion_date_and_time specifies the date and optionally the time on which the Work_item is expected to be finished.

4.2.73.4 order_date_and_time

The order_date_and_time specifies the date and optionally the time the Work_item was created.

4.2.73.5 preliminary_review_date_and_time

The preliminary_review_date_and_time specifies the date and time that a review will be held to check on the status of the work being conducted.

4.2.73.6 priority

The priority specifies how much emphasis should be placed upon completing the Work_item.

EXAMPLE 38 - Typically, priority is simply stated as high, medium or low.

4.2.73.7 start_date_and_time

The start_date_and_time specifies the date that work is expected to commence.

4.2.73.8 work_description

The work_description specifies a complete listing and definition of the work to be done.

4.2.73.9 work_requirements

The work_requirements specifies resources needed to carry out the work as described. The work_requirements need not be specified for a particular Work_item.

EXAMPLE 39 - Work_requirements may describe required materials and labor.

4.2.74 Work_order

A Work_order is a directive for work to be done. Each Work_order is either an External_order (see 4.2.19) or an Internal_order (see 4.2.26). In addition, each Work_order is either a Change_order (see 4.2.3) or a Start_order (see 4.2.66). The work to be done includes part, die design and process planning efforts. The Work_order gives a description of the work to be done and requirements that the resulting deliverables must satisfy. The data associated with a Work_order are the following:

— applicable_standards;

— approval;

— completion_date_and_time;

— order_date_and_time;

— preliminary_review_date_and_time;

— priority;

— production_volume;

— start_date_and_time;

— work_description;

— work_order_number;

— work_requirements.

4.2.74.1 applicable_standards

The applicable_standards specifies the callout of standards and specifications that must or should be used in carrying out the work described by the Work_order. There may be more than one applicable_standards for a Work_order.

4.2.74.2 approval

An approval specifies all relevant signoffs and signoff dates by respective management and technical personnel to indicate that the Work_order has met certain requirements from their perspectives. There may be more than one approval for a Work_order.

4.2.74.3 completion_date_and_time

The completion_date_and_time specifies the date on which the Work_order is expected to be finished.

4.2.74.4 order_date_and_time

The order_date_and_time specifies the date the Work_order was created.

4.2.74.5 preliminary_review_date_and_time

The preliminary_review_date_and_time specifies the date and time that a review will be held to check on the status of the work being conducted.

4.2.74.6 priority

The priority specifies how much emphasis should be placed upon completing the Work_order.

EXAMPLE 40 - Typically, priority is simply stated as high, medium or low.

4.2.74.7 production_volume

The production_volume specifies the intended quantity of sheet metal parts whose production will be associated with the Work_order.

4.2.74.8 start_date_and_time

The start_date_and_time specifies the date and time that work is expected to commence on the Work_order. The start_date_and_time need not be specified for a particular Work_order.

4.2.74.9 work_description

The work_description specifies a complete listing and definition of the work to be done. The work_description need not be specified for a particular Work_order.

4.2.74.10 work_order_number

The work_order_number specifies uniquely the Work_order.

4.2.74.11 work_requirements

The work_requirements specifies resources needed to carry out the work as described. The work_requirements need not be specified for a particular Work_order.

EXAMPLE 41 - Work_requirements may describe required materials and labor.

4.2.75 Work_order_relationship

A Work_order_relationship is a relationship between two Work_orders (see 4.2.74). Work_order_relationships enable reference by a Work_order to other Work_orders. The data associated with a Work_order_relationship is description.

The description specifies the nature of the Work_order_relationship.

EXAMPLE 42 - Work_order_relationships may model the requirement that work orders be able to be issued for other work orders. For example, a project for the production of a sheet metal part may be ordered by the issuance of a master Work_order. Different kinds of work may be required by this master Work_order, such as planning the production process, designing the in-process shapes of the part, and designing the die shapes to correspond with the in-process shapes the part will assume during production. Each of these three kinds of work may have a Work_-order issued for them. Work_order_relationship models the association of this master Work_order with the Work_-orders covering each of these three portions of the overall project.

4.2.76 Work_order_responsibility

A Work_order_responsibility is the specification of the party responsible for carrying out a Work_order. The data associated with a Work_order_responsibility is role.

The role specifies the role played by the Work_order_responsibility. The role need not be specified for a particular Work_order_responsibility.

4.2.77 Work_request

A Work_request is a solicitation for some type of sheet metal work to be done. Each Work_request is either a Change_request (see 4.2.4) or a Start_request (see 4.2.67). The type of sheet metal work that may be done includes part designs, die designs and process planning. The data associated with a Work_request are the following:

— date_and_time_of_request;

— requestor;

— work_request_identification.

4.2.77.1 date_and_time_of_request

The date_and_time_of_request specifies the date and optionally the time, at which the Work_request was created.

4.2.77.2 requestor

The requestor specifies the person and organization making or creating the Work_request.

4.2.77.3 work_request_identification

The work_request_identification specifies uniquely a Work_request.

4.3 Application assertions

This subclause specifies the application assertions for the sheet metal die planning and design application protocol. Application assertions specify the relationships between application objects, the cardinality of the relationships, and the rules required for the integrity and validity of the application objects and UoFs. The application assertions and their definitions are given below.

4.3.1 Assembly_component_relationship to Assembly_component_relationship

Each Assembly_component_relationship substitutes for zero, one, or many Assembly_component_relationship objects. Each Assembly_component_relationship is substituted for by zero or one Assembly_component_-relationship objects.

4.3.2 Assembly_component_relationship to Shape_definition

Each Assembly_component_relationship has as the shape of the component in the assembly zero, one, or many Shape_definition objects. Each Shape_definition describes the shape of zero or one Assembly_component_-relationship objects.

4.3.3 Change_order to Work_item

Each Change_order is based on zero, one, or many Work_item objects. Each Work_item is the subject of zero, one, or many Change_order objects.

Each Change_order results in zero, one, or many Work_item objects. Each Work_item is the resultant of zero, one, or many Change_order objects.

EXAMPLE 43 - When the design of a sheet metal panel is analyzed, it is determined that the model will tear. This results in a change_request which, in turn, results in a change_order which, when carried out, results in a (revised) work_item. The change_order "is based on" is the first work_item and "results in" is the (revised) work_item.

4.3.4 Coincidence_defaults to Shape_definition

Each Coincidence_defaults provides default uncertainties for one or more Shape_definition objects. Each Shape_definition has as default uncertainties exactly one Coincidence_defaults object.

4.3.5 Die_definition to Die_definition_constraint

Each die_definition is constrained by zero, one, or many Die_definition_constraint objects. Each Die_-definition_constraint constrains exactly one Die_definition object.

4.3.6 Die_definition to Process_operation

Each Die_definition defines the tools for zero, one, or many Process_operation objects. Each Process_-operation requires as tool the die defined by zero, one, or many Die_definition objects.

4.3.7 Die_shape_definition to Shape_tolerance

Each Die_shape_definition is associated with zero, one, or many Shape_tolerance objects. Each Shape_tolerance is called out in zero, one, or many Die_shape_definition objects.

4.3.8 Dimensional_representation to Dimension_tolerance_range

Each Dimensional_representation may vary according to zero, one, or many Dimension_tolerance_range objects. Each Dimension_tolerance_range describes permissible variation of one or many Dimensional_representation objects.

4.3.9 Dimension_tolerance_range to Tolerance_value

Each Dimension_tolerance_range has as upper limit zero or one Tolerance_value objects. Each Tolerance_value is upper limit for one or many Dimension_tolerance_range objects.

Each Dimension_tolerance_range has as lower limit zero or one Tolerance_value objects. Each Tolerance_value is lower limit for one or many Dimension_tolerance_range objects.

4.3.10 Feature to Feature_property

Each Feature possesses zero, one, or many Feature_property objects. Each Feature_property belongs to exactly one Feature object.

4.3.11 Feature to Shape_definition

Each Feature has shape defined by zero, one, or many Shape_definition objects. Each Shape_definition defines the shape of zero, one, or many Feature objects.

NOTES

1 - The Shape_definitions which define a Feature are subsets of the Shape_definition which characterizes the Item_definition which includes that Feature.

2 - When the Shape_definition which defines a Feature would be the same as the Shape_definition which characterizes the Item_definition which has the Feature, the Feature is considered to have shape defined by zero Shape_definitions. This is to avoid redundant Shape_definitions.

4.3.12 Feature_shape_definition to Shape_tolerance

Each Feature_shape_definition is associated with zero, one, or more Shape_tolerance objects. Each Shape_tolerance is called out in zero, one, or more Feature_shape_definition objects.

4.3.13 Final_part_definition to Part_on_product

Each Final_part_definition defines the Part on zero, one, or many Part_on_product objects. Each Part_on_-product defines the Marketable_assembly context of exactly one Final_part_definition object.

4.3.14 Final_part_definition to Part_process_plan

Each Final_part_definition has manufacture defined by zero, one, or many Part_process_plan objects. Each Part_process_plan defines the manufacture of one or more Final_part_definition objects.

4.3.15 In_process_part_definition to Process_operation

Each In_process_part_definition is operated upon by one or more Process_operation objects. Each Process_-operation operates upon one or more In_process_part_definition objects.

4.3.16 Input_item_die_relationship to Shape_definition

Each Input_item_die_relationship has input item position within the die described by zero, one, or many Shape_definition objects. Each Shape_definition describes the input item position within the die of zero or one Input_item_die_relationship objects.

4.3.17 Item to Item_classification

Each Item is classified by zero, one, or many Item_classification objects. Each Item_classification classifies one or more Item objects.

4.3.18 Item to Item_version

Each Item is versioned by one or more Item_version objects. Each Item_version versions exactly one Item object.

NOTE - Items which are not considered versionable have one Item_version, whose item_version_identification is zero.

4.3.19 Item_classification to Item_classification_relationship

Each Item_classification is primary in zero, one, or many Item_classification_relationship objects. Each Item_-classification_relationship has as primary exactly one Item_classification object.

Each Item_classification is secondary in zero, one or many Item_classification_relationship objects. Each Item_classification_relationship has as secondary exactly one Item_classification object.

4.3.20 Item_definition to Feature

Each Item_definition includes zero, one, or many Feature objects. Each Feature is included in exactly one Item_definition object.

4.3.21 Item_definition to Item_definition_relationship

Each Item_definition is first in zero, one, or many Item_definition_relationship objects. Each Item_definition_-relationship includes as first exactly one Item_definition object.

Each Item_definition is second in zero, one, or many Item_definition_relationship objects. Each Item_-definition_relationship includes as second exactly one Item_definition object.

4.3.22 Item_definition to Shape_definition

Each Item_definition has shape characterized by zero, one, or many Shape_definition objects. Each Shape_-definition characterizes the shape of zero or one Item_definition objects.

4.3.23 Item_version to Item_definition

Each Item_version is defined by one or more Item_definition objects. Each Item_definition is the definition of exactly one Item_version object.

4.3.24 Marketable_assembly to Part_on_product

Each Marketable_assembly is the product for zero, one, or many Part_on_product objects. Each Part_on_-product is founded on exactly one Marketable_assembly object.

4.3.25 Material_shape_definition to Shape_tolerance

Each Material_shape_definition is associated with zero, one, or many Shape_tolerance objects. Each Shape_tolerance is called out in zero, one, or many Material_shape_definition objects.

4.3.26 Mating_relationship to Shape_definition

Each Mating_relationship has as the shape of the interface zero, one, or many Shape_definition objects. Each Shape_definition describes the shape of zero or one Mating_relationship objects.

4.3.27 Part_definition to Material_definition

Each Part_definition has material specified by zero, one, or many Material_definition objects. Each Material_definition is the material specification for zero or one Part_definition objects.

NOTE - Only Part_definitions which are also Designed_items are required to participate in this relationship with Material_definition. For Part_definitions which are External_item_references, this relationship is optional.

4.3.28 Part_on_product to Part_on_product_location

Each Part_on_product is located at zero, one, or many Part_on_product_location objects. Each Part_on_-product_location is the location of exactly one Part_on_product object.

4.3.29 Part_on_product to Production_effectivity

Each Part_on_product has zero, one, or many Production_effectivity objects. Each Production_effectivity applies to one or more Part_on_product objects.

4.3.30 Part_process_plan to Part_process_plan

Each Part_process_plan substitutes for zero, one, or many Part_process_plan objects. Each Part_process_plan is substituted for by zero or one Part_process_plan objects.

4.3.31 Part_process_plan to Part_process_plan_version_relationship

Each Part_process_plan is the previous part process plan in zero, one, or more Part_process_plan_version_-relationship objects. Each Part_process_plan_version_relationship includes as previous exactly one Part_-process_plan object.

Each Part_process_plan is the next part process plan in zero, one, or more Part_process_plan_version_-relationship objects. Each Part_process_plan_version_relationship includes as next exactly one Part_process_-plan object.

4.3.32 Part_process_plan to Plant_constraint

Each Part_process_plan is constrained by zero, one, or many Plant_constraint objects. Each Plant_constraint constrains one or more Part_process_plan objects.

4.3.33 Part_process_plan to Process_operation

Each Part_process_plan defines a series of one or more Process_operation objects. Each Process_operation is part of exactly one Part_process_plan object.

4.3.34 Part_process_plan_template to Part_process_plan

Each Part_process_plan_template is the template for one or many Part_process_plan objects. Each Part_-process_plan has as a template zero or one Part_process_plan_template objects.

4.3.35 Part_shape_definition to Shape_tolerance

Each Part_shape_definition is associated with zero, one, or many Shape_tolerance objects. Each Shape_tolerance is called out in zero, one, or many Part_shape_definition objects.

4.3.36 Press_definition to Press_definition

Each Press_definition is alternate for for zero, one or many Press_definition objects. Each Press_definition serves alternately for zero or one Press_definition objects.

4.3.37 Press_definition to Process_operation

Each Press_definition defines a press used for one or more Process_operation objects. Each Process_operation uses a press defined by zero or one Press_definition objects.

4.3.38 Process_operation to Process_operation

Each Process_operation is alternate for zero, one, or more Process_operation objects. Each Process_operation serves alternately for zero or one Process_operation objects.

4.3.39 Process_operation to Scrap_definition

Each Process_operation produces zero, one, or many Scrap_definition objects. Each Scrap_definition is produced by exactly one Process_operation object.

Each Process_operation uses zero, one, or many Scrap_definition objects. Each Scrap_definition is used by exactly one Process_operation object.

4.3.40 Representation_element to Shape_definition

Each Representation_element represents zero, one, or many Shape_definition objects. Each Shape_definition is represented by zero, one, or many Representation_element objects.

4.3.41 Shape_definition to Shape_definition_relationship

Each Shape_definition is primary in zero, one, or many Shape_definition_relationship objects. Each Shape_-definition_relationship includes as primary exactly one Shape_definition object.

Each Shape_definition is secondary in zero, one, or many Shape_definition_relationship objects. Each Shape_-definition_relationship includes as secondary exactly one Shape_definition object.

4.3.42 Specified_material_relationship to Specified_material_relationship

Each Specified_material_relationship substitutes for zero, one, or many Specified_material_relationship objects. Each Specified_material_relationship is substituted for by zero or one Specified_material_relationship objects.

4.3.43 Tolerance_bound to Shape_tolerance

Each Tolerance_bound bounds one or many Shape_tolerance objects. Each Shape_tolerance is bounded by zero, one, or many Tolerance_bound objects.

4.3.44 Work_item to Start_order

Each Work_item is modified or initiated by zero, one, or many Start_order objects. Each Start_order modifies or initiates one or more Work_item objects.

4.3.45 Work_order to Work_order_relationship

Each Work_order is primary in zero, one, or many Work_order_relationship objects. Each Work_order_-relationship includes as primary exactly one Work_order object.

Each Work_order is secondary in zero, one, or many Work_order_relationship objects. Each Work_order_-relationship includes as secondary exactly one Work_order object.

4.3.46 Work_order_responsibility to Work_order

Each Work_order_responsibility is responsible for one or more Work_order objects. Each Work_order is carried out by zero, one, or many Work_order_responsibility objects.

4.3.47 Work_request to Work_order

Each Work_request results in zero, one or many Work_order objects. Each Work_order results from zero or one Work_request 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 resource constructs (see annex A).

The mapping table is organized in five columns. The contents of these five columns are:

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

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

Column 3) Source: For those AIM elements that are interpreted from the integrated resources, this is the number of the corresponding part of ISO 10303. For those AIM elements that are created for the purpose of this part of ISO 10303, this is the number of this part.

NOTE 1- 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.

The following AIC references, as defined within annex F, are used within this part of ISO 10303:

a) 501 : aic_edge_based_wireframe

b) 502 : aic_shell_based_wireframe

c) 507 : aic_geometrically_bounded_surface

d) 509 : aic_manifold_surface

e) 510 : aic_geometrically_bounded_wireframe

f) 512 : aic_faceted_brep

g) 514 : aic_advanced_brep

h) 515 : aic_csg

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

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

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

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

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

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

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

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

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

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

k) \ : the line continuations for strings that wrap.

Table 1 - Mapping table for feature UoF

Application element	AIM element	Source	Rules	Reference path
FEATURE	shape_aspect	41		
designer	sheet_metal_person_-assignment	207	25	shape_aspect person_assigned_item = shape_aspect person_assigned_item <- sheet_metal_person_assignment.items[i] sheet_metal_person_assignment {<= person_assignment person_assignment.role -> person_role person_role.name = 'designer'}
feature_name	shape_aspect.name	41		
feature to feature_-property	PATH			shape_aspect shape_definition = shape_aspect shape_definition characterized_definition = shape_definition characterized_definition <- property_definition.definition property_definition => material_property
feature to shape_definition	IDENTICAL MAPPING			

Table 1 - Mapping table for feature UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
FEATURE_PROPERTY #1: For feature_properties that are intrinsic to the material #2: For feature_properties that are not intrinsic to the material	#1(material_property) #2(descriptive_-representation_item)	45 45		
property_description #1: If the property is intrinsic to the material and the description is purely textual #2: If the property is intrinsic to the material and the description is a measurement #3: If the property is not intrinsic to the material	#1(property_definition.-description) #2 (measure_-representation_item) #3(descriptive_-representation_item.description)	41 41 45		#1 (material_property <= property_definition property_definition.description) #2 (material_property <= property_definition <- property_definition_representation.definition property_definition_representation property_definition_representation.used_representation -> representation representation.items[i]-> representation_item => measure_representation_item) #3 (descriptive_representation_item <= representation_item representation_item.name)

Table 1 - Mapping table for feature UoF (continued)

Application element	AIM element	Source	Rules	Reference path
property_type #1: If the property is intrinsic to the material #2: If the property is not intrinsic to the material	#1 (property_definition.-name) #2 (representation_item.-name)	41 43		#1 (material_property <= property_definition property_definition.name) #2 (descriptive_representation_item <= representation_item representation_item.name)

Table 2 - Mapping table for item UoF (continued)

Table 2 - Mapping table for item UoF

Application element	AIM element	Source	Rules	Reference path
DIE	product	41	18, 19	{product <- product_related_product_category.products[i] [product_related_product_category => product_type] [product_related_product_category <= product_category product_category.name = 'die']}
ITEM	product	41	18, 19	
item_description	product.description	41		
item_name	product.name	41		
item_number	product.id	41		
item to item_classification	PATH			product <- product_related_product_category.products[i] product_related_product_category
item to item_version	PATH		18	product <- product_definition_formation.of_product product_definition_formation
ITEM_-CLASSIFICATION	product_related_-product_category	41		
classification_description	product_category .description	41		product_related_product_category <= product_category product_category.description

Table 2 - Mapping table for item UoF (continued)

Application element	AIM element	Source	Rules	Reference path
classification_- identification	product_category.name	41		product_related_product_category <= product category product_category.name
classification_name	product_category.name	41		product_related_product_category <= product category product_category.name
item_classification to item_classification_- relationship (is primary in)	PATH			product_related_product_category <= product_category <- product_category_relationship.category product_category_relationship
item_classification to item_classification_- relationship (is secondary in)	PATH			product_related_product_category <= product_category <- product_category_relationship.subcategory product_category_relationship
ITEM_- CLASSIFICATION_- RELATIONSHIP	product_category_- relationship	41		
relationship_description	product_category_- relationship.description	41		
MATERIAL	product	41	18 , 19	{product <- product_related_product_category.products[i] [product_related_product_category => product_type] [product_related_product_category <= product_category product_category.name = 'material'']}

Table 2 - Mapping table for item UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
PART	product	41	18, 19	{product <- product_related_product_category.products[i] [product_related_product_category => product_type] [product_related_product_category <= product_category product_category.name = 'part']}

Table 3 - Mapping table for item_definition UoF

Application element	AIM element	Source	Rules	Reference path
DESIGNED_ITEM	product_definition	41	18	{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 = 'internal'}

Table 3 - Mapping table for item_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
creation_date_and_time #1: If date is the number of the day in the week, the number of the week in the year, and the number of the year. #2: If date is the number of the day in the month, the number of the month in the year, and the number of the year. #3: If date the number of the day in the year and the number of the year. #4: Same as case 1, but with the time included. #5: Same as case 2, but with the time included. #6: Same as case 3, but with the time included.	#1, #2, #3 (sheet_metal_-date_assignment) #4, #5, #6 (sheet_metal_date_and_-time_assignment)	207 207	21, 22	product_definition #1-#3 (date_assigned_item = product_definition date_assigned_item <- sheet_metal_date_assignment.items[i] sheet_metal_date_assignment {sheet_metal_date_assignment <= date_assignment [date_assignment.assigned_date -> date => #1 (week_of_year_and_day_date) #2 (calendar_date) #3 (ordinal_date)] [date_assignment.role -> date_role date_role.name = 'creation date']}) #4-#6 (date_and_time_assigned_item = product_definition date_and_time_assigned_item <- sheet_metal_date_and_time_assignment.items[i] sheet_metal_date_and_time_assignment {sheet_metal_date_and_time_assignment <= date_and_time_assignment [date_and_time_assignment.assigned_date_and_time -> date_and_time date_and_time.date_component -> date => #4 (week_of_year_and_day_date) #5 (calendar_date) #6 (ordinal_date)] date_and_time_assignment.role ->] date_time_role date_time_role.name = 'creation date and time']})

Table 3 - Mapping table for item_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
data_exchange_history	sequenced_product_-definition_relationship	207		product_definition <- [product_definition_relationship.relating_product_definition] [product_definition_relationship.related_product_definition] product_definition_relationship => sequenced_product_definition_relationship
data_ownership	sheet_metal_person_and_-organization_assignment	207	24	product_definition person_and_organization_assigned_item = product_definition person_and_organization_assigned_item <- sheet_metal_person_and_organization_assignment.items[i] sheet_metal_person_and_organization_assignment {sheet_metal_person_and_organization_assignment <= person_and_organization_assignment person_and_organization_assignment.role -> person_and_organization_role person_and_organization_role.name =’data owner’}
designer	sheet_metal_person_-assignment	207	25	product_definition person_assigned_item = product_definition person_assigned_item <- sheet_metal_person_assignment.items[i] sheet_metal_person_assignment {sheet_metal_person_assignment <= person_assignment person_assignment.role -> person_role person_role.name = 'designer'}

Table 3 - Mapping table for item_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
generating_system_-information	document.description	41		product_definition => product_definition_with_associated_documents product_definition_with_associated_documents.documentation_ids[i] ->document document.description {document document.kind ->document_type document_type.product_data_type = 'electronic'}
media_requirements	document_type	41		product_definition => product_definition_with_associated_documents product_definition_with_associated_documents.documentation_ids[i] ->document document.kind ->document_type
DIE_DEFINITION	product_definition_with_-associated_documents	41	18	{product_definition_with_associated_documents <=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 = 'die definition'] [product_related_product_category.products[i] [product_related_product_category =>product_type] [product_related_product_category <=product_category product_category.name = 'die']]}

Table 3 - Mapping table for item_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
die_function_-description	product_definition.-description	41		product_definition_with_associated_documents <= product_definition product_definition.description
die_layout_-specification_reference	sheet_metal_document_-reference	207		{product_definition_with_associated_documents product_definition_with_associated_documents.documentation_ids[i] -> document document.kind -> document_type document_type.product_data_type = 'die layout'} product_definition_with_associated_documents <= product_definition document_referenced_item = product_definition document_referenced_item <- sheet_metal_document_reference.items[i] sheet_metal_document_reference
die_structure_-specification_reference	sheet_metal_document_-reference	207		{product_definition_with_associated_documents product_definition_with_associated_documents.documentation_ids[i] -> document document.kind -> document_type document_type.product_data_type = 'die structure'} product_definition_with_associated_documents <= product_definition document_referenced_item = product_definition document_referenced_item <- sheet_metal_document_reference.items[i] sheet_metal_document_reference
die_weight	product_definition .description	41		product_definition_with_associated_documents <= product_definition product_definition.description

Table 3 - Mapping table for item_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
pattern_casting _- specification	sheet_metal_document_- reference	207		{product_definition_with_associated_documents product_definition_with_associated_documents.documentation_ids[i] ->document document.kind ->document_type document_type.product_data_type = 'pattern casting' } product_definition_with_associated_documents <=product_definition document_referenced_item = product_definition document_referenced_item <- sheet_metal_document_reference.items[i] sheet_metal_document_reference
die_definition to die_- definition_constraint #1: For die_definition_- constraints that are press_definitions or plant_constraints #2: For die_definition_- constraints that are not for process planning	PATH			product_definition_with_associated_documents <= #1 (product_definition characterized_product_definition = product_definition characterized_product_definition <- process_product_association.defined_product process_product_association process_product_association.process ->product_definition_process <= action supported_item = action supported_item <- action_resource.usage action_resource <- requirement_for_action_resource.resources[i] requirement_for_action_resource) #2 (product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship => die_definition_constraint_relationship)

Table 3 - Mapping table for item_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
die_definition to process_operation	PATH			product_definition_with_associated_documents <= product_definition action_assigned_item = product_definition action_assigned_item <- sheet_metal_action_assignment.items[i] sheet_metal_action_assignment <= action_assignment action_assignment.assigned_action => action => product_definition_process
EXTERNAL_ITEM_-REFERENCE	product_definition_with_-associated_documents	41	18	{product_definition_with_associated_documents <= 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 = 'external'] [product_related_product_category.products[i] [product_related_product_category => product_type] [product_related_product_category <= product_category product_category.name = 'part']]}
manual_reference_-description	product_definition_with_-associated_documents.-documentation_ids[i]	41		

Table 3 - Mapping table for item_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
name	product.name	41	18, 19	product_definition_with_associated_documents <= product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product product.name
needed_modifications	product_definition.-description	41		product_definition_with_associated_documents <= product_definition product_definition.description
FINAL_PART_-DEFINITION	product_definition	41	26	{product_definition [product_definition.frame_of_reference -> product_definition_context product_definition_context.life_cycle_stage = 'design'] [product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] [product_related_product_category => product_type] [product_related_product_category <= product_category product_category.name = 'part']}
final_part_definition to part_on_product	PATH			product_definition <- product_definition_relationship.related_product_definition product_definition_relationship => product_definition_usage => assembly_component_usage

Table 3 - Mapping table for item_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
final_part_definition to part_process_plan	PATH		15	product_definition action_assigned_item = product_definition action_assigned_item <- sheet_metal_action_assignment.items[i] sheet_metal_action_assignment <= action_assignment action_assignment.assigned_action ->action => process_plan
IN_PROCESS_PART_-DEFINITION	product_definition	41	26	{product_definition [product_definition.frame_of_reference -> product_definition_context product_definition_context.life_cycle_stage = 'in process'] [product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] [product_related_product_category => product_type] [product_related_product_category <= product_category product_category.name = 'part']}
in_process_part_-definition to process_-operation	PATH			product_definition characterized_product_definition = product_definition characterized_product_definition <- process_product_association.defined_product process_product_association process_product_association.process -> product_definition_process
ITEM_DEFINITION	product_definition	41	26	

Table 3 - Mapping table for item_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
approval	[approval] [approval_date_time] [approval_person_- organization]	41 41 41	1, 2,11	product_definition approval_assigned_item = product_definition approval_assigned_item <- sheet_metal_approval_assignment.items[i] sheet_metal_approval_assignment <= approval_assignment approval_assignment.assigned_approval -> [approval] [approval <- approval_date_time.dated_approval approval_date_time] [approval <- approval_person_organization.authorized_approval approval_person_organization]
approval_status	approval_status	41		product_definition approval_assigned_item = product_definition approval_assigned_item <- sheet_metal_approval_assignment.items[i] sheet_metal_approval_assignment <= approval_assignment approval_assignment.assigned_approval -> approval approval.status -> approval_status
definition_description	product_definition- .description	41		

Table 3 - Mapping table for item_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
procurement_information	sheet_metal_organization_-assignment	207	8, 23	product_definition organization_assigned_item = product_definition organization_assigned_item <- sheet_metal_organization_assignment.items[i] sheet_metal_organization_assignment {sheet_metal_organization_assignment <= organization_assignment organization_assignment.role ->> organization_role organization_role.name = 'design supplier'}
proprietary_security_-usage_information	sheet_metal_security_-classification_assignment	207		product_definition security_classification_assigned_item = product_definition security_classification_assigned_item <- sheet_metal_security_classification_assignment.items[i] sheet_metal_security_classification_assignment
item_definition to feature	PATH			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
item_definition to item_-definition_relationship (is first in)	PATH			product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship

Table 3 - Mapping table for item_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
item_definition to item_-definition_relationship (is second in)	PATH			product_definition <- product_definition_relationship.related_product_definition product_definition_relationship
Application element	**AIM element**	**Source**	**Rules**	**Reference path**
item_definition to shape_definition	PATH			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
MATERIAL_-DEFINITION	product_definition_with_-associated_documents	41		{product_definition_with_associated_documents <= 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 = 'material definition'] [product_related_product_category.products[i] [product_related_product_category => product_type] [product_related_product_category <= product_category product_category.name = 'material']]}
material_description	product_definition.-description	41		product_definition_with_associated_documents <= product_definition product_definition.description

Table 3 - Mapping table for item_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
material_specification	document.id	41		product_definition_with_associated_documents product_definition_with_associated_documents.documentation_ids[i] ->document document.id

Table 3 - Mapping table for item_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
PART_DEFINITION	product_definition	41	26	{product_definition [product_definition.frame_of_reference -> product_definition_context {product_definition_context.life_cycle_stage = 'in process'} (product_definition_context.life_cycle_stage = 'design')] [product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product <- product_related_product_category.products[i] [product_related_product_category.products[i] product_related_product_category <= product_category product_category.name = 'part definition'] [product_related_product_category.products[i] [product_related_product_category => product_type] [product_related_product_category <= product_category product_category.name = 'part']]}
part_definition to material_definition	PATH			product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship {product_definition_relationship => product_definition_usage => assembly_component_usage => promissory_usage_occurrence} product_definition_relationship.related_product_definition-> product_definition => product_definition_with_associated_documents

Table 3 - Mapping table for item_definition UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
SCRAP_DEFINITION	product_definition_with_-associated_documents	41		{product_definition_with_associated_documents <= 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 = 'scrap definition'] [product_related_product_category.products[i] [product_related_product_category => product_type] [product_related_product_category <= product_category (product_category.name = 'material') (product_category.name = 'unspecified')]]}

Table 4 - Mapping table for item_definition_relationship UoF

Application element	AIM element	Source	Rules	Reference path
ASSEMBLY_- COMPONENT_- RELATIONSHIP	quantified_assembly_- component_usage	44	27, 28	{quantified_assembly_component_usage <= assembly_component_usage <= product_definition_usage <= product_definition_relationship product_definition_relationship.name = 'assembly component'}
component_quantity	measure_with_unit	41	27	quantified_assembly_component_usage quantified_assembly_component_usage.quantity ->> measure_with_unit {measure_with_unit.value_component ->> measure_value measure_value = count_measure}
assembly_component_- relationship to assembly_component_- relationship	PATH		20	quantified_assembly_component_usage <= assembly_component_usage <- assembly_component_usage_substitute.base assembly_component_usage_substitute assembly_component_usage_substitute.substitute ->> assembly_component_usage => quantified_assembly_component_usage
assembly_component_- relationship to shape_- definition	PATH			quantified_assembly_component_usage <= assembly_component_usage <= product_definition_usage <= product_definition_relationship characterized_product_definition = product_definition_relationship characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape

Table 4 - Mapping table for item_definition_relationship UoF (continued)

Application element	AIM element	Source	Rules	Reference path
INPUT_ITEM_DIE_-RELATIONSHIP	input_item_die_-relationship	207		input_item_die_relationship <= product_definition_relationship
input_item_die_-relationship to shape_-definition	PATH			input_item_die_relationship <= product_definition_relationship characterized_product_definition = product_definition_relationship characterized_product_definition characterized_definition = characterized_product_definition characterized_definition <- property_definition.definition property_definition => product_definition_shape
ITEM_DEFINITION_-RELATIONSHIP	product_definition_-relationship	41	28	
relationship_-description	product_definition_-relationship.description	41		
MAKE_SIMILAR_TO_-RELATIONSHIP	product_definition_-relationship	41		{product_definition_relationship product_definition_relationship.name = 'similar'}
MATING_-RELATIONSHIP	product_definition_-relationship	41	28	{product_definition_relationship product_definition_relationship.name = 'mating'}

Table 4 - Mapping table for item_definition_relationship UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
mating_relationship to shape_definition	PATH			{product_definition_relationship product_definition_relationship.name = 'mating'} characterized_product_definition = product_definition_relationship 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.name = 'mating shape'}
SPECIFIED_- MATERIAL_- RELATIONSHIP	make_from_usage_option	44		
material_quantity	measure_with_unit	41		make_from_usage_option make_from_usage_option.quantity ->measure_with_unit {measure _with_unit.value_component ->measure_value measure_value = mass_measure}
specified_material_- relationship to specified_material_- relationship	PATH			make_from_usage_option <- make_from_usage_option_group.members[i] make_from_usage_option_group make_from_usage_option_group.members[i] ->make_from_usage_option
SYMMETRICAL_- ITEM_RELATIONSHIP	product_definition_- relationship	41	28	{product_definition_relationship product_definition_relationship.name = 'symmetrical'}

Table 5 - Mapping table for item_version UoF

Application element	AIM element	Source	Rules	Reference path
ITEM_VERSION	product_- definition_formation	41	16, 17	
approval	[approval] [approval_date_time] [approval_person_- organization]	41 41 41	1, 2,11	product_definition_formation approval_assigned_item = product_definition_formation approval_assigned_item <- sheet_metal_approval_assignment.items[i] sheet_metal_approval_assignment <= approval_assignment approval_assignment.assigned_approval -> [approval] [approval <- approval_date_time.dated_approval approval_date_time] [approval <- approval_person_organization.authorized_approval approval_person_organization]
approval_status	approval_status	41		product_definition_formation approval_assigned_item = product_definition_formation approval_assigned_item <- sheet_metal_approval_assignment.items[i] sheet_metal_approval_assignment <= approval_assignment approval_assignment.assigned_approval -> approval approval.status -> approval_status
description	product_- definition_formation- .description	41		

Table 5 - Mapping table for item_version UoF (continued)

Application element	AIM element	Source	Rules	Reference path
item_version_-identification	product_-definition_formation.id	41		
revision_date_and_time #1: If date is expressed as a combination of the number of the day in the week, the number of the week in the year, and the number of the year. #2: If date is expressed as a combination of the number of the day in the month, the number of the month in the year, and the number of the year. #3: If date is expressed as a combination of the number of the day in the year and the number of the year.	#1, #2, #3 (sheet_metal_-date_assignment)	207	16, 21, 22	product_definition_formation <- product_definition.formation product_definition #1-#3 (date_assigned_item = product_definition date_assigned_item <- sheet_metal_date_assignment.items[i] sheet_metal_date_assignment {sheet_metal_date_assignment <= date_assignment [date_assignment.assigned_date -> date => #1 (week_of_year_and_day_date) #2 (calendar_date) #3 (ordinal_date)] [date_assignment.role -> date_role date_role.name = 'revision date']})
#4: Same as case 1, but with the time included. #5: Same as case 2, but with the time included. #6: Same as case 3, but with the time included.	#4, #5, #6 (sheet_metal_-date_and_time_-assignment)	207		#4-#6 (date_and_time_assigned_item = product_definition date_and_time_assigned_item <- sheet_metal_date_and_time_assignment.items[i] sheet_metal_date_and_time_assignment {sheet_metal_date_and_time_assignment <= date_and_time_assignment [date_and_time_assignment.assigned_date_and_time -> date_and_time date_and_time.date_component -> date => #4 (week_of_year_and_day_date) #5 (calendar_date) #6 (ordinal_date)] [date_and_time_assignment.role -> date_time_role date_time_role.name= 'revision date and time']})

Table 5 - Mapping table for item_version UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
item_version to item_-definition	PATH		17	product_definition_formation <- product_definition.formation product_definition
MADE_IN_HOUSE	product_-definition_formation_-with_specified_source	41	16, 17	{product_definition_formation_with_specified_source product_definition_formation_with_specified_source.make_or_buy -> source source = 'make'}
PURCHASED	product_-definition_formation_-with_specified_source	41	16, 17	{product_definition_formation_with_specified_source product_definition_formation_with_specified_source.make_or_buy -> source source = 'buy'}
purchase_requirements	sheet_metal_contract_-assignment	207		product_definition_formation_with_specified_source <- contract_assigned_item = product_definition_formation_with_specified_source contract_assigned_item <- sheet_metal_contract_assignment.items[i] sheet_metal_contract_assignment

Table 6 - Mapping Table for part_process_plan UoF

Application element	AIM element	Source	Rules	Reference path
DIE_DEFINITION_-CONSTRAINT #1: For die_definition_-constraints that are press_definitions or plant_constraints	#1 :(requirement_for_-action_resource)	49	26	{#1 requirement_for_action_resource action_resource_requirement (requirement_for_action_resource requirement_for_action_resource.resources[i] ->) action_resource --this is the press action_resource.usage ->) supported_item supported_item=action_method action_method NOTE 2 - the action_method above is the first process_operation in a chain. NOTE 3 - The following block is repeated zero or more times, depending on how many action_methods deep into the action (in ARM terms, how many process_operations deep into the process_plan) it is necessary to traverse until one encounters the action_method (in ARM terms, process_operation) that is constrained by this requirement_for_action_resource (in ARM terms, die_definition_constraint). (action_method_relationship.related_method action_method_relationship action_method_relationship.relating_method ->) action_method) NOTE 4 - the action_method above is the (n + 1)th (ARM) process_opera-tion in a chain. action_method <- action.chosen_method action NOTE 5 - the action above is the (ARM) process_plan. product_definition_process <- process_product_association.process process_product_association

Table 6 - Mapping table for part_process_plan (continued)

Application element	AIM element	Source	Rules	Reference path
DIE_DEFINITION_- CONSTRAINT #1: For die_definition_- constraints that are press_definitions or plant_constraints (continued)				product_definition_process <- process_product_association.process process_product_association process_product_association.defined_product -> characterized_product_definition characterized_product_definition = product_definition [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_related_product_category => product_type] [product_related_product.category <= product_category product_category.name = 'die']] [product_definition <- product_definition_relationship.relating_product_definition product_definition_relationship product_definition_relationship.related_product_definition -> product_definition product_definition.frame_of_reference -> product_definition_context product_definition_context.life_cycle_stage = 'as design constrained']})

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
DIE_DEFINITION_-CONSTRAINT (concluded) #2: If not for process planning	#2: (die_definition_-constraint_relationship)	207	26	#2 (die_definition_constraint_relationship <= product_definition_relationship {product_definition_relationship product_definition_relationship.related_product_definition ->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_related_product_category =>product_type] [product_related_product_category <=product_category product_category.name = 'die']] [product_definition.frame_of_reference ->product_definition_context product_definition_context.life_cycle_stage = 'as design constrained']})
constraint_description NOTE 6 - This application element has no mapping when Die_definition_-constraint is a press_definition or a plant_constraint, as in case 1 above, because it is not populated in that case.	product_definition_-relationship.description	41		die_definition_constraint_relationship <= product_definition_relationship product_definition_relationship.description
PART_PROCESS_-PLAN	process_plan	207	15	process_plan <= action

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
approval	[approval] [approval_date_time] [approval_person_- organization]	41 41 41	1, 2, 11	process_plan approval_assigned_item = process_plan approval_assigned_item <- sheet_metal_approval_assignment.items[i] sheet_metal_approval_assignment <= approval_assignment approval_assignment.assigned_approval ->[approval] [approval <- approval_date_time.dated_approval approval_date_time] [approval <- approval_person_organization.authorized_approval approval_person_organization]
approval_status	approval_status	41		process_plan approval_assigned_item = process_plan approval_assigned_item <- sheet_metal_approval_assignment.items[i] sheet_metal_approval_assignment <= approval_assignment approval_assignment.assigned_approval ->approval approval.status ->approval_status

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
creation_date_and_time #1: If date is the number of the day in the week, the number of the week in the year, and the number of the year. #2: If date is the number of the day in the month, the number of the month in the year, and the number of the year. #3: If date the number of the day in the year and the number of the year. #4: Same as case 1, but with the time included. #5: Same as case 2, but with the time included. #6: Same as case 3, but with the time included.	#1, #2, #3 (sheet_metal_-date_assignment) #4, #5, #6 (sheet_metal_-date_and_time_-assignment)	207 207	21, 22	process_plan #1-#3 (date_assigned_item = process_plan date_assigned_item <- sheet_metal_date_assignment.items[i] sheet_metal_date_assignment {sheet_metal_date_assignment <= date_assignment [date_assignment.assigned_date -> date => #1 (week_of_year_and_day_date) #2 (calendar_date) #3 (ordinal_date)] [date_assignment.role -> date_role date_role.name = 'creation date']}) #4-#6 (date_and_time_assigned_item = process_plan date_and_time_assigned_item <- sheet_metal_date_and_time_assignment.items[i] sheet_metal_date_and_time_assignment {sheet_metal_date_and_time_assignment <= date_and_time_assignment [date_and_time_assignment.assigned_date_and_time -> date_and_time date_and_time.date_component -> date => #4 (week_of_year_and_day_date) #5 (calendar_date) #6 (ordinal_date)] date_and_time_assignment.role -> date_time_role date_time_role.name = 'creation date and time'})

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
data_ownership	sheet_metal_person_-and_organization_-assignment	207	24	process_plan person_and_organization_assigned_item = process_plan person_and_organization_assigned_item <- sheet_metal_person_and_organization_assignment.items[i] sheet_metal_person_and_organization_assignment {sheet_metal_person_and_organization_assignmen t<= person_and_organization_assignment person_and_organization_assignment.role -> person_and_organization_role person_and_organization_role.name = 'data owner'}
generating_system_-information	sheet_metal_document_-reference	207		process_plan document_referenced_item = process_plan document_referenced_item <- sheet_metal_document_reference.items[i] sheet_metal_document_reference {sheet_metal_document_reference <= document_reference document_reference.assigned_document -> document document.kind -> document_type document_type.product_data_type = 'generating system information'}
part_process_version_-identification	action_relationship.name	41		process_plan <= action <- [action_relationship.related_action] [action_relationship.relating_action] action_relationship action_relationship.name

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
plan_status	approval.level	41	1, 2, 3	process_plan approval_assigned_item = process_plan approval_assigned_item <- sheet_metal_approval_assignment.items[i] sheet_metal_approval_assignment <= approval_assignment approval_assignment.assigned_approval ->> approval approval.level
planner	sheet_metal_person_and_organization_assignment	207	24	process_plan person_and_organization_assigned_item = process_plan person_and_organization_assigned_item <- sheet_metal_person_and_organization_assignment.items[i] sheet_metal_person_and_organization_assignment <= person_and_organization_assignment {person_and_organization_assignment person_and_organization_assignment.role ->> person_and_organization_role person_and_organization_role.name = 'planner'}
production_rate	measure_representation_item	45		process_plan <= action characterized_action_definition = action characterized_action_definition <- action_property.definition action_property <- action_property_representation.property action_property_representation action_property_representation.representation ->> representation representation.items[i] ->> representation_item => measure_representation_item

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
proprietary_security_usage_information	sheet_metal_security_classification_assignment	207		process_plan security_classification_assigned_item = process_plan security_classification_assigned_item <- sheet_metal_security_classification_assignment.items[i] sheet_metal_security_classification_assignment

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
review_date_and_time #1: If date is the number of the day in the week, the number of the week in the year, and the number of the year. #2: If date is expressed as a combination of the number of the day in the month, the number of the month in the year, and the number of the year. #3: If date is the number of the day in the year and the number of the year. #4: Same as case 1, but with the time included. #5: Same as case 2, but with the time included. #6: Same as case 3, but with the time included.	#1, #2, #3 (sheet_metal_-date_assignment) #4, #5, #6 (sheet_metal_-date_and_time_-assignment)	207 207	21, 22	process_plan #1-#3 (date_assigned_item = process_plan date_assigned_item <- sheet_metal_date_assignment.items[i] sheet_metal_date_assignment {sheet_metal_date_assignment <= date_assignment [date_assignment.assigned_date -> date => #1 (week_of_year_and_day_date) #2 (calendar_date) #3 (ordinal_date)] [date_assignment.role -> date_role date_role.name = 'review date']}) #4-#6 (date_and_time_assigned_item = process_plan date_and_time_assigned_item <- sheet_metal_date_and_time_assignment.items[i] sheet_metal_date_and_time_assignment {sheet_metal_date_and_time_assignment <= date_and_time_assignment [date_and_time_assignment.assigned_date_and_time -> date_and_time date_and_time.date_component -> date => #4 (week_of_year_and_day_date) #5 (calendar_date) #6 (ordinal_date)] [date_and_time_assignment.role -> date_time_role date_time_role.name= 'review date and time']})

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
version_description	action_- relationship.description	41		process_plan <= action <- action_relationship.related_action action_relationship action_relationship.description
part_process_plan to part_process_plan	PATH			process_plan <= action <- action_relationship.related_action {action_relationship => replacement_relationship} action_relationship action_relationship.relating_action ->> action => process_plan
part_process_plan to part_process_plan_- version_relationship (is the previous)	PATH			process_plan <= action <- action_relationship.related_action action_relationship
part_process_plan to part_process_plan_- version_relationship (is the next)	PATH			process_plan <= action <- action_relationship.relating_action action_relationship
part_process_plan to plant_constraint	PATH			process_plan <= action characterized_action_definition = action characterized_action_definition <- action_resource_requirement.operations[i] action_resource_requirement

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
part_process_plan to process_operation	PATH		3	process_plan <= action action.chosen_method -> action_method (action_method <- action_method_relationship.relating_method action_method_relationship action_method_relationship.related_method -> action_method) <- action.chosen_method action => product_definition_process NOTE 7 - There may be many instances of action_method_-relationship related to the first ocurrance of action_method, either directly or through chains.
PART_PROCESS_-PLAN_TEMPLATE	action_method	41	3	{action_method action_method.name = 'part process plan template'}
part_process_plan_-template to part_process_plan	PATH		3	action_method <- action.chosen_method action => process_plan
PART_PROCESS_-PLAN_VERSION_-RELATIONSHIP	action_relationship	41		
revision_reason	action_-relationship.description	41		
PLANT_CONSTRAINT	action_resource_-requirement	49		

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
constraint_description	action_resource_- requirement.description	49		
constraint_type	resource_property.name	49		action_resource_requirement characterized_resource_definition = action_resource_requirement characterized_resource_definition <- resource_property.resource resource_property resource_property.name
constraint_value	measure_representation_- item	45		action_resource_requirement characterized_resource_definition = action_resource_requirement characterized_resource_definition <- resource_property.resource resource_property <- resource_property_representation.property resource_property_representation resource_property_representation.representation ->representation representation.items[i] ->representation_item =>measure_representation_item
plant_identification	action_resource.name	41	23	action_resource_requirement => requirement_for_action_resource requirement_for_action_resource.resources[i] ->{action_resource action_resource.kind ->action_resource_type action_resource_type.name = 'plant'}action_resource action_resource.name
PRESS_DEFINITION	action_resource_- requirement	49	4	{action_resource_requirement action_resource_requirement.name = ‘press definition’}

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
automation_equipment	action_resource	41		action_resource_requirement => requirement_for_action_resource requirement_for_action_resource.resources[i] -> action_resource <- action_resource_relationship.related_resource action_resource_relationship action_resource_relationship.relating_resource -> action_resource
press_forces	measure_representation_-item	45		action_resource_requirement characterized_resource_definition = action_resource_requirement characterized_resource_definition <- resource_property.resource {resource_property resource_property.name = 'press force'} resource_property <- resource_property_representation.property resource_property_representation resource_property_representation.representation -> representation representation.items[i] -> representation_item => measure_representation_item
press_identification	action_resource.name	41		action_resource_requirement => requirement_for_action_resource requirement_for_action_resource.resources[i] -> action_resource action_resource.name
press_line	resource_property	49		action_resource_requirement characterized_resource_definition = action_resource_requirement characterized_resource_definition <- resource_property.resource resource_property {resource_property.name = 'press line'}

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
press_location	resource_property	49		action_resource_requirement characterized_resource_definition = action_resource_requirement characterized_resource_definition <- resource_property.resource resource_property {resource_property.name = 'press location'}
press_size #1: If the press size refers to its length. #2: If the press size refers to its width. #3: If the press size refers to its height. #4: If the press size refers to its shut height.	measure_representation_-item	45		action_resource_requirement characterized_resource_definition = action_resource_requirement characterized_resource_definition <- resource_property.resource {resource_property #1 (resource_property.name = 'press length = capacity') #2 (resource_property.name = 'press width = capacity') #3 (resource_property.name = 'press height = capacity') #4 (resource_property.name = 'shut height')} resource_property <- resource_property_representation.property resource_property_representation resource_property_representation.representation ->representation representation.items[i] ->representation_item => measure_representation_item

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
press_specification_-reference	sheet_metal_document_-reference	207		action_resource_requirement {action_resource_requirement => requirement_for_action_resource requirement_for_action_resource.resources[i] -> action_resource action_resource.kind -> action_resource_type action_resource_type.name = 'press'} document_referenced_item = action_resource_requirement document_referenced_item <- sheet_metal_document_reference.items[i] sheet_metal_document_reference {sheet_metal_document_reference <= document_reference document_reference.assigned_document -> document document.kind -> document_type document_type.product_data_type = 'press'}
press_type	action_resource.-description	41		action_resource_requirement => requirement_for_action_resource requirement_for_action_resource.resources[i] -> action_resource action_resource.description
press_definition to press_definition	PATH			action_resource_requirement <- action_resource_requirement_relationship. / related_action_resource_requirement action_resource_requirement_relationship action_resource_requirement_relationship. / relating_action_resource_requirement -> action_resource_requirement

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
press_definition to process_operation	PATH		4	action_resource_requirement action_resource_requirement.operations[i] ->> characterized_action_definition characterized_action_definition = action action => product_definition_process
PROCESS_-OPERATION	product_definition_-process	49	4	
operation_description	action.description	41		product_definition_process <= action action.description
operation_sequence_-number	sequential_-method.sequence_-position	49		product_definition_process <= action action.chosen_method -> action_method <- action_method_relationship.related_method action_method_relationship => serial_action_method => sequential_method sequential_method.sequence_position
reference_specifications	action_method_with_-associated_documents	49		product_definition_process <= action action.chosen_method -> action_method => action_method_with_associated_documents

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
required_shut_height	action_resource_-requirement	49	4	product_definition_process <= action action.chosen_method -> action_method characterized_action_definition = action_method characterized_action_definition <- action_resource_requirement.operations[i] action_resource_requirement {action_resource_requirement action_resource_requirement.name = 'required shut height'}
required_stamping_-forces	action_resource_-requirement	49	4	product_definition_process <= action action.chosen_method -> action_method characterized_action_definition = action_method characterized_action_definition <- action_resource_requirement.operations[i] action_resource_requirement {action_resource_requirement action_resource_requirement.name = 'required stamping force'}

Table 6 - Mapping table for part_process_plan UoF (continued)

Application element	AIM element	Source	Rules	Reference path
scrap_percentage	measure_representation_-item	45		product_definition_process <= action action.chosen_method -> action_method characterized_action_definition = action_method characterized_action_definition <- action_property.definition {action_property action_property.name = 'scrap percentage'} <- action_property <- action_property_representation.property action_property_representation action_property_representation.representation -> representation representation.items[i] -> representation_item => measure_representation_item
units_per_operation	measure_representation_-item	45		product_definition_process <= action action.chosen_method -> action_method characterized_action_definition = action_method characterized_action_definition <- action_property.definition {action_property action_property.name = 'units per operation'} action_property <- action_property_representation.property action_property_representation action_property_representation.representation -> representation representation.items[i] -> representation_item => measure_representation_item

Table 6 - Mapping table for part_process_plan UoF(concluded)

Application element	AIM element	Source	Rules	Reference path
process_operation to process_operation	action_relationship	41		product_definition_process <= action <- action_relationship.relating_action action_relationship {action_relationship.name = 'alternate'} action_relationship.related_action -> action => product_definition_process
process_operation to scrap_definition (produces)	PATH			{product_definition_process <= action action.name = 'produce scrap'} product_definition_process <- process_product_association.process process_product_association process_product_association.defined_product -> characterized_product_definition characterized_product_definition = product_definition product_definition => product_definition_with_associated_documents
process_operation to scrap_definition (uses)	PATH			{product_definition_process <= action action.name = 'use scrap'} product_definition_process <- process_product_association.process process_product_association process_product_association.defined_product -> characterized_product_definition characterized_product_definition = product_definition product_definition => product_definition_with_associated_documents

Table 7 - Mapping table for product UoF

Application element	AIM element	Source	Rules	Reference path
MARKETABLE_-ASSEMBLY	product_concept	44		
model_identification	product_concept.id	44		
production_year	date.year_component	41	21	product_concept date_assigned_item = product_concept date_assigned_item <- sheet_metal_date_assignment.items[i] sheet_metal_date_assignment <= date_assignment {date_assignment.role -> date_role date_role.name = 'production year date'} date_assignment.assigned_date -> date date.year_component
style_identification	product_concept.name	44		
marketable_assembly to part_on_product	PATH			product_concept <- configuration_item.item_concept configuration_item <- configuration_design.configuration configuration_design configuration_design.design -> product_definition_formation <- product_definition.formation product_definition <- product_definition_relationship.related_product_definition product_definition_relationship => product_definition_usage => assembly_component_usage

Table 7 - Mapping table for product (concluded)

Application element	AIM element	Source	Rules	Reference path
PART_ON_PRODUCT	assembly_component_-usage	44		
part_on_product to part_on_product_-location	PATH			assembly_component_usage assembly_component_usage.reference_designator
part_on_product to production_effectivity	PATH			assembly_component_usage <= product_definition_usage <= product_definition_relationship <- product_definition_effectivity.usage product_definition_effectivity <= effectivity
PART_ON_-PRODUCT_LOCATION	assembly_component_-usage.reference_-designator	44		
intraproduct_location	assembly_component_-usage.reference_-designator	44		
PRODUCTION_-EFFECTIVITY	effectivity	41		
date_and_time_-effectivity	dated_effectivity	41		effectivity => dated_effectivity
effectivity_-identification	effectivity.id	41		
sequence_effectivity	serial_numbered_-effectivity	41		effectivity => serial_numbered_effectivity

Table 8 - Mapping table for representation_element UoF

Application element	AIM element	Source	Rules	Reference path
ANGLE_DISTANCE_-DIMENSION	angular_location	47		
clockwise	angle_relator	47		angular_location angular_location.angle_selection -> angle_relator
orientation	angle_relator	47		angular_location angular_location.angle_selection -> angle_relator
CURVE_DISTANCE_-DIMENSION	dimensional_location_-with_path	47		
CURVE_LENGTH_-SIZE_DIMENSION	dimensional_size	47		{dimensional_size.name = ‘curve length’}
DIAMETER_SIZE_-DIMENSION	dimensional_size	47		{dimensional_size.name = ‘diameter’}
DIMENSIONAL_-REPRESENTATION	shape_dimension_-representation	47		

Table 8 - Mapping table for representation_element UoF(continued)

Application element	AIM element	Source	Rules	Reference path
dimensional_- representation to dimension_tolerance_- range	PATH			shape_dimension_representation <- dimensional_characteristic_representation.representation dimensional_characteristic_representation dimensional_characteristic_representation.dimension ->dimensional_characteristic <- plus_minus_tolerance.toleranced_dimension pluse_minus_tolerance pluse_minus_tolerance.range->tolerance_method_definition
DISTANCE_- DIMENSION	dimensional_location	47		
LINEAR_DISTANCE_- DIMENSION #1: If a direction_vector is specified #2: If a direction_vector is not specified	#1: (dimensional_location_- with_path) #2: (dimensional_location)	47 47		
PHYSICAL_- REPRESENTATION	physically_modelled_- product_definition	207	26	physically_modelled_product_definition <= product_definition {product_definition product_definition.frame_of_reference ->product_definition_context product_definition_context.life_cycle_stage = 'as physically modelled'}
hard_aid_description	product_- definition.description	41		physically_modelled_product_definition <= product_definition product_definition.description

Table 8 - Mapping table for representation_element UoF (continued)

Application element	AIM element	Source	Rules	Reference path
hard_aid_identification	product.id	41	18, 19	physically_modelled_product_definition <= product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product product.id
hard_aid_type	product.name	41	18, 19	physically_modelled_product_definition <= product_definition product_definition.formation -> product_definition_formation product_definition_formation.of_product -> product product.name
POSITION_-ORIENTATION_-REPRESENTATION #1: If representing a relationship between two representation_-elements #2: If representing a single representation_element solely in order to impart context	#1 (representation_-relationship_with_-transformation) #2 (representation)	43 43		
RADIUS_SIZE_-DIMENSION	dimensional_size	47		{dimensional_size.name = ‘radius’}
REPRESENTATION_-ELEMENT	shape_representation	41	29	

Table 8 - Mapping table for representation_element UoF (continued)

Application element	AIM element	Source	Rules	Reference path
representation_element to shape_definition #1: If shape_definition is of an entire item #2: If shape_definition is of a portion of an 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 = shape_definition shape_definition #1 (shape_definition-product_definition_shape product_definition_shape) #2 (shape_definition =shape_aspect shape_aspect)
SIZE_DIMENSION	dimensional_size	47		
SOLID_-REPRESENTATION	(advanced_brep_shape_-representation) (faceted_brep_shape_-representation) (csg_shape_-representation)	514 512 515		(advanced_brep_shape_representation <=) (faceted_brep_shape_representaiton <=) (csg_shape_representation <=) shape_representation
B-rep	(advanced_brep_shape_-representation) (faceted_brep_shape_-representation)	514 512		(advanced_brep_shape_representation <=) (faceted_brep_shape_representaiton <=) shape_representation
CSG	csg_shape_representation	515		(csg_shape_representation <=) shape_representation

Table 8 - Mapping table for representation_element UoF (continued)

Application element	AIM element	Source	Rules	Reference path
SURFACE_- REPRESENTATION	(geometrically_- bounded_surface_shape_- representation) (manifold_surface_- shape_representation)	507 509		(geometrically_bounded_surface_shape_representation <-) (manifold_surface_shape_representation <=) shape_representation
surface_direction	direction	42		geometrically_bounded_surface_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 = shape_definition shape_definition = shape_aspect shape_aspect <- shape_aspect_relationship.relating_shape_aspect shape_aspect_relationship => dimensional_location dimensional_characteristic = dimensional_location dimensional_characteristic <- dimensional_characteristic_representation.dimension dimensional_characteristic_representation dimensional_characteristic_representation.representation ->> shape_dimension_representation <= shape_representation <= representation representation.items[i] ->> representation_item => geometric_representation_item => direction

Table 8 - Mapping table for representation element (concluded)

Application element	AIM element	Source	Rules	Reference path
surface_thickness	measure_representation_-item	45		geometrically_bounded_surface_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 = shape_definition shape_definition = shape_aspect shape_aspect <- shape_aspect_relationship.relating_shape_aspect shape_aspect_relationship => dimensional_location dimensional_characteristic = dimensional_location dimensional_characteristic <- dimensional_characteristic_representation.dimension dimensional_characteristic_representation dimensional_characteristic_representation.representation -> shape_dimension_representation <= shape_representation <= representation representation.items[i] -> representation_item => measure_representation_item
WIREFRAME_-REPRESENTATION	(edge_based_-wireframe_shape_-representation)	501		(edge_based_wireframe_shape_representation <=) (shell_based_wireframe_shape_representation <=) (geometrically_bounded_wireframe_shape_representation <=) shape_representation
	(shell_based_-wireframe_shape_-representation)	502		
	(geometrically_-bounded_wireframe_-shape_representation)	510		

Table 9 - Mapping table for shape_definition UoF

Application element	AIM element	Source	Rules	Reference path
DIE_SHAPE_-DEFINITION	product_definition_shape	41		{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 product_definition.formation -> product_definition_formation product_definition_formation.of_product -> {product <- product_related_product_category.products[i] product_related_product_category {product_related_product_category => product_type} {product_type <= product_related_product.category <= product_category product_category.name = 'die'}}}
die_shape_definition to shape_tolerance	PATH			product_definition_shape <- shape_aspect.of_shape shape_aspect <- geometric_tolerance.toleranced_shape_aspect geometric_tolerance

Table 9 - Mapping table for shape_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
FEATURE_SHAPE_-DEFINITION	shape_aspect	41		{shape_aspect shape_aspect.of_shape -> 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 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 = 'feature'}
feature_shape_definition to shape_tolerance	PATH			shape_aspect <- geometric_tolerance.toleranced_shape_aspect geometric_tolerance

Table 9 - Mapping table for shape_definition UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
MATERIAL_SHAPE_-DEFINITION	product_definition_shape	41		{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 product_definition.formation -> product_definition_formation product_definition_formation.of_product -> {product <- product_related_product_category.products[i] product_related_product_category {product_related_product_category => product_type} {product_type <= product_related_product.category <= product_category product_category.name = 'material'}}}
material_shape_-definition to shape_tolerance	PATH			product_definition_shape <- shape_aspect.of_shape shape_aspect <- geometric_tolerance.toleranced_shape_aspect geometric_tolerance

Table 9 - Mapping table for shape_definition UoF (continued)

Application element	AIM element	Source	Rules	Reference path
PART_SHAPE_-DEFINITION	product_definition_shape	41		{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 product_definition.formation -> product_definition_formation product_definition_formation.of_product -> {product <- product_related_product_category.products[i] product_related_product_category {product_related_product_category => product_type} {product_type <= product_related_product.category <= product_category product_category.name = 'part'}}}
part_shape_definition to shape_tolerance	PATH			product_definition_shape <- shape_aspect.of_shape shape_aspect <- geometric_tolerance.toleranced_shape_aspect geometric_tolerance
SHAPE_DEFINITION #1: If the shape being defined is of an entire item #2: If the shape being defined is of a portion of an item	#1 (product_definition_-shape) #2 (shape_aspect)	41 41		

Table 9 - Mapping table for shape_definition UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
shape_description	#1 (property_definition.-description) #2 (shape_aspect.-description)			#1(product_definition_shape <= property_definition property_definition.description) #2(shape_aspect shape_aspect.description)
shape_definition to shape_definition_-relationship (is primary in)	PATH			#1 (product_definition_shape <- shape_aspect.of_shape shape_aspect <- shape_aspect_relationship.related_shape_aspect shape_aspect_relationship) #2 (shape_aspect <- shape_aspect_relationship.related_shape_aspect shape_aspect_relationship shape_aspect_relationship)
shape_definition to shape_definition_-relationship (is secondary in)	PATH			#1 (product_definition_shape <- shape_aspect.of_shape shape_aspect <- shape_aspect_relationship.relating_shape_aspect shape_aspect_relationship) #2 (shape_aspect <- shape_aspect_relationship.relating_shape_aspect shape_aspect_relationship shape_aspect_relationship)

Table 10 - Mapping table for shape_definition_relationship UoF

Application element	AIM element	Source	Rules	Reference path
ENVELOPE_-RELATIONSHIP	shape_aspect_-relationship	41		{shape_aspect_relationship shape_aspect_relationship.name = 'envelope'}
SHAPE_DEFINITION_-RELATIONSHIP	shape_aspect_-relationship	41		
relationship_-description	shape_aspect_-relationship.description	41		
SIMILARITY_-RELATIONSHIP	shape_aspect_-relationship	41		{shape_aspect_relationship shape_aspect_relationship.name = 'similar'}

Table 11 - Mapping table for tolerance UoF

Application element	AIM element	Source	Rules	Reference path
COINCIDENCE_-DEFAULTS	global_uncertainty_-assigned_context	43	14	
position	uncertainty_measure_-with_unit	43	14	global_uncertainty_assigned_context global_uncertainty_assigned_context.uncertainty[i] ->uncertainty_measure_with_unit <=measure_with_unit{measure_with_unit[measure_with_unit.unit_component ->unitunit = named_unitnamed_unitnamed_unit = length_unit][measure_with_unit.value_component ->measure_valuemeasure_value = positive_length_measure]}
angle	uncertainty_measure_-with_unit	43	14	global_uncertainty_assigned_context global_uncertainty_assigned_context.uncertainty[i] ->uncertainty_measure_with_unit <=measure_with_unit{measure_with_unit[measure_with_unit.unit_component ->unitunit = named_unitnamed_unitnamed_unit = plane_angle_unit[measure _with_unit.value_component ->measure_valuemeasure_value = positive_plane_angle_measure]}

Table 11 - Mapping table for tolerance UoF (continued)

Application element	AIM element	Source	Rules	Reference path
coincidence_defaults to shape_definition #1: If the shape being defined is of an entire item #2: If the shape being defined is of a portion of an item	PATH			#1 (global_uncertainty_assigned_context <= representation_context <- representation.context_of_items representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition -> property_definition property_definition.definition -> characterized_definition shape_definition = characterized_definition shape_definition product_definition_shape = shape_definition product_definition_shape) #2 (global_uncertainty_assigned_context <= representation_context <- representation.context_of_items representation <- property_definition_representation.used_representation property_definition_representation property_definition_representation.definition -> property_definition property_definition.definition -> characterized_definition shape_definition = characterized_definition shape_definition shape_aspect = shape_definition shape_aspect)
DIMENSION_- TOLERANCE_RANGE	tolerance_method_- definition	47		
tolerance_fitting_type	limits_and_fits	47		tolerance_method_definition tolerance_method_definition = limits_and_fits limits_and_fits

Table 11 - Mapping table for tolerance UoF (continued)

Application element	AIM element	Source	Rules	Reference path
dimension_tolerance_-range to tolerance_value (has as upper limit)	PATH			tolerance_method_definition tolerance_method_definition = tolerance_value tolerance_value tolerance_value.upper_bound ->measure_with_unit
dimension_tolerance_-range to tolerance_value (has as lower limit)	PATH			tolerance_method_definition tolerance_method_definition = tolerance_value tolerance_value tolerance_value.lower_bound ->measure_with_unit
SHAPE_TOLERANCE	geometric_tolerance	47		
tolerance_condition	limit_condition	47		geometric_tolerance => modified_geometric_tolerance modified_geometric_tolerance.modifier ->limit_condition
tolerance_datum	geometric_tolerance_-with_datum_reference.-datum_system[i]	47		geometric_tolerance => geometric_tolerance_with_datum_reference geometric_tolerance_with_datum_reference.datum_system[i]
tolerance_type	geometric_tolerance-.name	47		
TOLERANCE_BOUND	tolerance_zone_definition	47		
tolerance_bound to shape_tolerance	PATH			tolerance_zone_definition tolerance_zone_definition.zone ->tolerance_zone tolerance_zone.defining_tolerance[i] ->geometric_tolerance

Table 11 - Mapping table for tolerance UoF (concluded)

Application element	AIM element	Source	Rules	Reference path
TOLERANCE_VALUE #1: If tolerance_value is expressed in terms of length. #2: If tolerance_value is expressed in terms of an angle.	measure_with_unit	41		measure_with_unit {#1 (measure_with_unit <= length_measure_with_unit) #2 (measure_with_unit <= plane_angle_measure_with_unit)}
unit_of_measure .	measure_with_unit.- unit_component	41		
value	measure_with_unit.- value_component	41		

Table 12 - Mapping table for work_order UoF

Application element	AIM element	Source	Rules	Reference path
CHANGE_ORDER	change_order	207	10, 11, 12, 13, 30	change_order <= work_order <= executed_action
change_design_location	action.description	41		change_order <= work_order <= executed_action <= action action.description

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
implementor #1: If the implementor is an individual person	#1 (sheet_metal_person_-assignment)	207	25	change_order #1 (person_assigned_item = change_order person_assigned_item <- sheet_metal_person_assignment.items[i] sheet_metal_person_assignment <= person_assignment {person_assignment person_assignment.role -> person_role person_role.name = 'implementor'})
#2: If the implementor is an organization represented by an individual person	#2 (sheet_metal_person_-and_organization_-assignment)	207	24	#2 (person_and_organization_assigned_item = change_order person_and_organization_assigned_item <- sheet_metal_person_and_organization_assignment.items[i] sheet_metal_person_and_organization_assignment <= person_and_organization_assignment {person_and_organization_assignment person_and_organization_assignment.role -> person_and_organization_role person_and_organization_role.name = 'implementor'})
#3: If the implementor is an organization without an individual contact person	#3 (sheet_metal_-organization_-assignment)	207	8, 10, 23	#3 (organization_assigned_item = change_order organization_assigned_item <- sheet_metal_organization_assignment.items[i] sheet_metal_organization_assignment <= organization_assignment {organization_assignment organization_assignment.role -> organization_role organization_role.name = 'implementor' })

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
NOTE 8 - The following two mappings are provided informatively, not normatively. There are no application assertions in the ARM as shown in the left column in this note.				
change_order to external_order	PATH	PATH	26	change_order contract_assigned_item = change_order contract_assigned_item <- sheet_metal_contract_assignment.items[i] sheet_metal_contract_assignment
change_order to internal_order	PATH	PATH		change_order organization_assigned_item = change_order organization_assigned_item <- sheet_metal_organization_assignment.items[i]
change_order to work_item (is based on)	PATH			change_order <= work_order <= executed_action <= action <- action_assignment.assigned_action action_assignment => sheet_metal_action_assignment
change_order to work_item (results in)	PATH			change_order <= work_order <= executed_action <= action <- action_assignment.assigned_action action_assignment => sheet_metal_action_assignment
EXTERNAL_ORDER	sheet_metal_contract_-assignment	207	10	sheet_metal_contract_assignment <= contract_assignment

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
contractual_-requirement	contract.purpose	41		sheet_metal_contract_assignment <= contract_assignment contract_assignment.assigned_contract -> contract contract.purpose
purchase_order_-number	contract.name	41		sheet_metal_contract_assignment <= contract_assignment contract_assignment.assigned_contract -> contract contract.name
purchasing_agent	sheet_metal_person_-assignment	207	25	sheet_metal_contract_assignment <= contract_assignment contract_assignment.assigned_contract -> contract person_assigned_item = contract person_assigned_item <- sheet_metal_person_assignment.items[i] sheet_metal_person_assignment <= {person_assignment person_assignment.role -> person_role person_role.name = 'purchaser'}

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
supplier_identification	sheet_metal_- organization_assignment	207	8, 23	sheet_metal_contract_assignment <= contract_assignment contract_assignment.assigned_contract ->
contract organization_assigned_item = contract organization_assigned_item <- sheet_metal_organization_assignment.items[i] sheet_metal_organization_assignment <= organization_assignment {organization_assignment organization_assignment.role ->
organization_role organization_role.name = 'supplier'}
external_order to start_order	PATH		10	sheet_metal_contract_assignment sheet_metal_contract_assignment.items[i] ->
contract_assigned_item contract_assigned_item = start_order start_order
INTERNAL_ORDER	sheet_metal_- organization_assignment	207	8, 10, 11, 12, 13, 23	sheet_metal_organization_assignment <= organization_assignment {organization_assignment.role ->
organization_role organization_role.name = 'supplier'} {sheet_metal_organization_assignment.items[i] ->
organization_assigned_item organization_assigned_item = executed_action}

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
authorized_hours	action_directive.comment	41	8	sheet_metal_organization_assignment sheet_metal_organization_assignment.items[i] ->organization_assigned_item organization_assigned_item = executed_action executed_action =>directed_action directed_action.directive ->action_directive action_directive.comment
charge_number	action_directive.comment	41	8	sheet_metal_organization_assignment sheet_metal_organization_assignment.items[i] ->organization_assigned_item organization_assigned_item = executed_action executed_action =>directed_action directed_action.directive ->action_directive action_directive.comment
source_department	organization.name	41	23	sheet_metal_organization_assignment <=organization_assignment organization_assignment.assigned_organization ->organization organization.name
NOTE 9 - The following mapping is provided informatively, not normatively. There is no application assertion in the ARM as shown in the left column in this note.				
internal_order to start_order	PATH	PATH	8,10	sheet_metal_organization_assignment sheet_metal_organization_assignment.items[i] ->organization_assigned_item organization_assigned_item = start_order start_order

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
START_ORDER	start_order	207	10, 11, 12, 13, 30	start_order <= work_order <= executed_action
WORK_ITEM	sheet_metal_action_-assignment	207	5, 6, 7	sheet_metal_action_assignment <= action_assignment
applicable_standards	sheet_metal_document_-reference	207		sheet_metal_action_assignment document_referenced_item = sheet_metal_action_assignment document_referenced_item <- sheet_metal_document_reference.items[i] sheet_metal_document_reference <= document_reference {document_reference document_reference.assigned_document -> document document.kind -> document_type document_type.product_data_type = 'applicable standards'}
approval	[approval] [approval_date_time] [approval_person_-organization]	41 41 41	1, 2, 11	sheet_metal_action_assignment approval_assigned_item = sheet_metal_action_assignment approval_assigned_item <- sheet_metal_approval_assignment.items[i] sheet_metal_approval_assignment <= approval_assignment approval_assignment.assigned_approval -> [approval] [approval <- approval_date_time.dated_approval approval_date_time] [approval <- approval_person_organization.authorized_approval approval_person_organization]

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
completion_date_and_time #1: If date is the number of the day in the week, the number of the week in the year, and the number of the year. #2: If date is the number of the day in the month, the number of the month in the year, and the number of the year. #3: If date is the number of the day in the year and the number of the year. #4: Same as case 1, but with the time included. #5: Same as case 2, but with the time included. #6: Same as case 3, but with the time included.	#1, #2, #3 (sheet_metal_-date_assignment) #4, #5, #6 (sheet_metal_-date_and_time_-assignment)	207 207	12, 21, 22	sheet_metal_action_assignment #1-#3 (date_assigned_item = sheet_metal_action_assignment date_assigned_item <- sheet_metal_date_assignment.items[i] sheet_metal_date_assignment <= date_assignment {date_assignment date_assignment.assigned_date -> date => #1 (week_of_year_and_day_date) #2 (calendar_date) #3 (ordinal_date)] date_assignment.role -> date_role date_role.name = 'completion date']}) #4-#6 (date_and_time_assigned_item = sheet_metal_action_assignment date_and_time_assigned_item <- sheet_metal_date_and_time_assignment.items[i] sheet_metal_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment [date_and_time_assignment.assigned_date_and_time -> date_and_time date_and_time.date_component -> date => #4 (week_of_year_and_day_date) #5 (calendar_date) #6 (ordinal_date)] [date_and_time_assignment.role -> date_time_role date_time_role.name= 'completion date and time']})

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
order_date_and_time #1: If date is the number of the day in the week, the number of the week in the year, and the number of the year. #2: If date is the number of the day in the month, the number of the month in the year, and the number of the year. #3: If date is the number of the day in the year and the number of the year. #4: Same as case 1, but with the time included. #5: Same as case 2, but with the time included. #6: Same as case 3, but with the time included.	#1, #2, #3 (sheet_metal_-date_assignment) #4, #5, #6 (sheet_metal_-date_and_time_-assignment)	207 207	12, 21, 22	sheet_metal_action_assignment #1-#3 (date_assigned_item = sheet_metal_action_assignment date_assigned_item <- sheet_metal_date_assignment.items[i] sheet_metal_date_assignment <= date_assignment {date_assignment date_assignment.assigned_date -> date => #1 (week_of_year_and_day_date) #2 (calendar_date) #3 (ordinal_date)] date_assignment.role -> date_role date_role.name = 'order date']}) #4-#6 (date_and_time_assigned_item = sheet_metal_action_assignment date_and_time_assigned_item <- sheet_metal_date_and_time_assignment.items[i] sheet_metal_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment [date_and_time_assignment.assigned_date_and_time -> date_and_time date_and_time.date_component -> date => #4 (week_of_year_and_day_date) #5 (calendar_date) #6 (ordinal_date)] [date_and_time_assignment.role -> date_time_role date_time_role.name= 'order date and time']})

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
preliminary_review_- date_and_time #1: If date is the number of the day in the week, the number of the week in the year, and the number of the year. #2: If date is the number of the day in the month, the number of the month in the year, and the number of the year. #3: If date is the number of the day in the year and the number of the year. #4: Same as case 1, but with the time included. #5: Same as case 2, but with the time included. #6: Same as case 3, but with the time included.	#1, #2, #3 (sheet_metal_- date_assignment) #4, #5, #6 (sheet_metal_- date_and_time_- assignment)	207 207	12, 21, 22	sheet_metal_action_assignment #1-#3 (date_assigned_item = sheet_metal_action_assignment date_assigned_item <- sheet_metal_date_assignment.items[i] sheet_metal_date_assignment <= date_assignment {date_assignment date_assignment.assigned_date -> date => #1 (week_of_year_and_day_date) #2 (calendar_date) #3 (ordinal_date)] date_assignment.role -> date_role date_role.name = 'preliminary review date']}) #4-#6 (date_and_time_assigned_item = sheet_metal_action_assignment date_and_time_assigned_item <- sheet_metal_date_and_time_assignment.items[i] sheet_metal_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment [date_and_time_assignment.assigned_date_and_time -> date_and_time date_and_time.date_component -> date => #4 (week_of_year_and_day_date) #5 (calendar_date) #6 (ordinal_date)] [date_and_time_assignment.role -> date_time_role date_time_role.name= ‘preliminary review date and time']})

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
priority	action_directive.comment	41		sheet_metal_action_assignment sheet_metal_action_assignment.items[i] -> action_assigned_item action_assigned_item = executed_action executed_action {executed_action => work_order} executed_action => directed_action directed_action.directive -> action_directive action_directive.comment

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
start_date_and_time #1: If date is the number of the day in the week, the number of the week in the year, and the number of the year. #2: If date is the number of the day in the month, the number of the month in the year, and the number of the year. #3: If date is the number of the day in the year and the number of the year. #4: Same as case 1, but with the time included. #5: Same as case 2, but with the time included. #6: Same as case 3, but with the time included.	#1, #2, #3 (sheet_metal_-date_assignment) #4, #5, #6 (sheet_metal_-date_and_time_-assignment)	207 207	12, 21, 22	sheet_metal_action_assignment #1-#3 (date_assigned_item = sheet_metal_action_assignment date_assigned_item <- sheet_metal_date_assignment.items[i] sheet_metal_date_assignment <= date_assignment {date_assignment date_assignment.assigned_date -> date => #1 (week_of_year_and_day_date) #2 (calendar_date) #3 (ordinal_date)] date_assignment.role -> date_role date_role.name = 'start date']}) #4-#6 (date_and_time_assigned_item = sheet_metal_action_assignment date_and_time_assigned_item <- sheet_metal_date_and_time_assignment.items[i] sheet_metal_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment [date_and_time_assignment.assigned_date_and_time -> date_and_time date_and_time.date_component -> date => #4 (week_of_year_and_day_date) #5 (calendar_date) #6 (ordinal_date)] [date_and_time_assignment.role -> date_time_role date_time_role.name= ‘start date and time']})

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
work_description	action_method.-description	41		sheet_metal_action_assignment sheet_metal_action_assignment.items[i] ->action_assigned_item action_assigned_item = executed_action {executed_action => work_order} executed_action <= action action.method ->action_method action_method.description
work_requirements	action_resource.-description	41		sheet_metal_action_assignment sheet_metal_action_assignment.items[i] ->action_assigned_item action_assigned_item = executed_action {executed_action => work_order} executed_action <= action supported_item = action supported_item <-action_resource.usage action_resource action_resource.description
work_item to start_order	PATH		9	sheet_metal_action_assignment <=action_assignment action_assignment.assigned_action ->action => executed_action => work_order => start_order

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
WORK_ORDER #1: If work_order is a change_order #2: If work_order is a start_order	#1 (change_order) #2 (start_order)	207 207	9, 10, 11, 12, 13, 30	#1 (change_order <=) #2 (start_order <=) work_order <= executed_action
applicable_standards	sheet_metal_document_-reference	207		#1 (change_order <=) #2 (start_order <=) {work_order <= executed_action document_referenced_item = executed_action document_referenced_item <- sheet_metal_document_reference.items[i] sheet_metal_document_reference <= document_reference {document_reference document_reference.assigned_document -> document document.kind -> document_type document_type.product_data_type = 'applicable standards'}

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
approval	[approval] [approval_date_time] [approval_person_-organization]	41 41 41	1, 2, 11	#1 (change_order <=) #2 (start_order <=) work_order <= executed_action approval_assigned_item = executed_action approval_assigned_item <- sheet_metal_approval_assignment.items[i] sheet_metal_approval_assignment <= approval_assignment approval_assignment.assigned_approval ->[approval] [approval <- approval_date_time.dated_approval approval_date_time] [approval <- approval_person_organization.authorized_approval approval_person_organization]

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
completion_- date_and_time #3: If date is the number of the day in the week,the number of the week in the year, and the number of the year. #4: If date is the number of the day in the month, the number of the month in the year, and the number of the year. #5: If date is the number of the day in the year and the number of the year.	#3, #4, #5 (sheet_metal_- date_assignment)	207	12, 21, 22	#1 (change_order <=) #2 (start_order <=) work_order <= executed_action #3-#5 (date_assigned_item = executed_action date_assigned_item <- sheet_metal_date_assignment.items[i] sheet_metal_date_assignment <= date_assignment {date_assignment [date_assignment.assigned_date -> date => #3 (week_of_year_and_day_date) #4 (calendar_date) #5 (ordinal_date)] [date_assignment.role -> date_role date_role.name = 'completion date']})
#6: Same as case 3, but with the time included. #7: Same as case 4, but with the time included. #8: Same as case 5 but with the time included.	#6, #7, #8 (sheet_metal_- date_and_time_- assignment)	207		#6-#8 (date_and_time_assigned_item = executed_action date_and_time_assigned_item <- sheet_metal_date_and_time_assignment.items[i] sheet_metal_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment [date_and_time_assignment.assigned_date_and_time -> date_and_time date_and_time.date_component -> date => #6 (week_of_year_and_day_date) #7 (calendar_date) #8 (ordinal_date)] date_and_time_assignment.role -> date_time_role date_time_role.name= 'completion date and time']})

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
order_date_and_time #3: If date is the number of the day in the week,the number of the week in the year, and the number of the year. #4: If date is the number of the day in the month, the number of the month in the year, and the number of the year. #5: If date is the number of the day in the year and the number of the year.	#3, #4, #5 (sheet_metal_-date_assignment)	207	12, 21, 22	#1 (change_order) #2 (start_order) work_order <= executed_action #3-#5 (date_assigned_item = executed_action date_assigned_item <- sheet_metal_date_assignment.items[i] sheet_metal_date_assignment <= date_assignment {date_assignment [date_assignment.assigned_date -> date => #3 (week_of_year_and_day_date) #4 (calendar_date) #5 (ordinal_date)] [date_assignment.role -> date_role date_role.name = 'order date']})
#6: Same as case 3, but with the time included. #7: Same as case 4, but with the time included. #8: Same as case 5, but with the time included.	#6, #7, #8 (sheet_metal_-date_and_time_-assignment)	207		#6-#8 (date_and_time_assigned_item = executed_action date_and_time_assigned_item <- sheet_metal_date_and_time_assignment.items[i] sheet_metal_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment [date_and_time_assignment.assigned_date_and_time -> date_and_time date_and_time.date_component -> date => #6 (week_of_year_and_day_date) #7 (calendar_date) #8 (ordinal_date)] date_and_time_assignment.role -> date_time_role date_time_role.name= 'order date and time']})

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
preliminary_review_-date_and_time #3: If date is the number of the day in the week,the number of the week in the year, and the number of the year. #4: If date is the number of the day in the month, the number of the month in the year, and the number of the year. #5: If date is the number of the day in the year and the number of the year.	#3, #4, #5 (sheet_metal_-date_assignment)	207	12, 21, 22	#1 (change_order) #2 (start_order) work_order <= executed_action #3-#5 (date_assigned_item = executed_action date_assigned_item <- sheet_metal_date_assignment.items[i] sheet_metal_date_assignment <= date_assignment {date_assignment [date_assignment.assigned_date -> date => #3 (week_of_year_and_day_date) #4 (calendar_date) #5 (ordinal_date)] [date_assignment.role -> date_role date_role.name = 'preliminary review date']})
#6: Same as case 3, but with the time included. #7: Same as case 4, but with the time included. #8: Same as case 5, but with the time included.	#6, #7, #8 (sheet_metal_-date_and_time_-assignment)	207		#6-#8 (date_and_time_assigned_item = executed_action date_and_time_assigned_item <- sheet_metal_date_and_time_assignment.items[i] sheet_metal_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment [date_and_time_assignment.assigned_date_and_time -> date_and_time date_and_time.date_component -> date => #6 (week_of_year_and_day_date) #7 (calendar_date) #8 (ordinal_date)] date_and_time_assignment.role -> date_time_role date_time_role.name='preliminary review date and time']})

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
priority	action.description	41		#1 (change_order <=) #2 (start_order <=) work_order <= executed_action <= action action.description
production_volume	measure_representation_-item	45		#1 (change_order <=) #2 (start_order <=) work_order <= executed_action <= action characterized_action_definition = action characterized_action_definition <- action_property.definition action_property <- action_property_representation.property action_property_representation action_property_representation.representation ->representation representation.items[i] ->representation_item => measure_representation_item

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
start_date_and_time #3: If date is the number of the day in the week,the number of the week in the year, and the number of the year. #4: If date is the number of the day in the month, the number of the month in the year, and the number of the year. #5: If date is the number of the day in the year and the number of the year.	#3, #4, #5 (sheet_metal_-date_assignment)	207	12, 21, 22	#1 (change_order) #2 (start_order) work_order <= executed_action #3-#5 (date_assigned_item = executed_action date_assigned_item <- sheet_metal_date_assignment.items[i] sheet_metal_date_assignment <= date_assignment {date_assignment [date_assignment.assigned_date -> date => #3 (week_of_year_and_day_date) #4 (calendar_date) #5 (ordinal_date)] [date_assignment.role -> date_role date_role.name = 'start date']})
#6: Same as case 1, but with the time included. #7: Same as case 2, but with the time included. #8: Same as case 3, but with the time included.	#6, #7, #8 (sheet_metal_-date_and_time_-assignment)	207		#6-#8 (date_and_time_assigned_item = executed_action date_and_time_assigned_item <- sheet_metal_date_and_time_assignment.items[i] sheet_metal_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment [date_and_time_assignment.assigned_date_and_time -> date_and_time date_and_time.date_component -> date => #6 (week_of_year_and_day_date) #7 (calendar_date) #8 (ordinal_date)] date_and_time_assignment.role -> date_time_role date_time_role.name= 'start date and time']})

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
work_description	action_method.-description	41		#1 (change_order <=) #2 (start_order <=) work_order <= executed_action <= action action.method ->action_method action_method.description
work_order_number	action.name	207		#1 (change_order <=) #2 (start_order <=) work_order <= executed_action <= action action.name
work_requirements	action_resource.-description	41		#1 (change_order <=) #2 (start_order <=) work_order <= executed_action <= action supported_item = action supported_item <-action_resource.usage action_resource action_resource.description
work_order to work_-order_relationship (is primary in) #1: If work_order is a change_order #2: If work_order is a start_order	PATH			#1 (change_order <=) #2 (start_order <=) work_order <= executed_action <= action <-action_relationship.relating_action action_relationship => work_order_relationship

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
work_order to work_-order_relationship (is secondary in)	PATH			#1 (change_order <=) #2 (start_order <=) work_order <= executed_action <= action <- action_relationship.related_action action_relationship => work_order_relationship
WORK_ORDER_-RELATIONSHIP	work_order_relationship	207		work_order_relationship <= action_relationship
description	action_relationship.-.description	41		work_order_relationship <= action_relationship action_relationship.description
WORK_ORDER_-RESPONSIBILITY	organization	41	8, 13, 23	{organization <- organization_assignment.assigned_organization organization_assignment organization_assignment.role ->organization_role organization_role.name = 'effector'}
role	organization_role	41	23	organization <- organization_assignment.assigned_organization organization_assignment organization_assignment.role ->organization_role

Table 12 - Mapping table for work_order UoF (continued)

Application element	AIM element	Source	Rules	Reference path
work_order_ - responsibility to work_ - order	PATH		8, 13, 23	organization <- organization_assignment.assigned_organization organization_assignment => sheet_metal_organization_assignment sheet_metal_organization_assignment.items[i] organization_assigned_item organization_assigned_item = executed_action executed_action => work_order => #1 (change_order) #2 (start_order)

Table 13 - Mapping table for work_order_request UoF

Application element	AIM element	Source	Rules	Reference path
CHANGE_REQUEST	versioned_action_request	41	31, 32	{versioned_action_request versioned_action_request.purpose = 'change'}
problem_change_-description	versioned_action_-request.description	41		
recommended_solution	action_method	41	3	versioned_action_request <- action_request_solution.request action_request_solution action_request_solution.method -> action_method
START_REQUEST	versioned_action_request	41	31, 32	{versioned_action_request versioned_action_request.purpose = 'start'}
request_description	versioned_action_-request.description	41		
request_justification	versioned_action_-request.purpose	41		
WORK_REQUEST	versioned_action_request	41	31, 32	

Table 13 - Mapping table for work_request (continued)

Application element	AIM element	Source	Rules	Reference path
date_and_time_of_-request #1: If date is the number of the day in the week, the number of the week in the year, and the number of the year. #2: If date is the number of the day in the month, the number of the month in the year, and the number of the year. #3: If date is the number of the day in the year and the number of the year. #4: Same as case 1, but with the time included. #5: Same as case 2, but with the time included. #6: Same as case 3, but with the time included.	#1, #2, #3 (sheet_metal_-date_assignment) #4, #5, #6 (sheet_metal_-date_and_time_-assignment)	207 207	21, 22, 31	versioned_action_request #1-#3 (date_assigned_item = versioned_action_request date_assigned_item <- sheet_metal_date_assignment.items[i] sheet_metal_date_assignment <= date_assignment {date_assignment [date_assignment.assigned_date -> date => #1 (week_of_year_and_day_date) #2 (calendar_date) #3 (ordinal_date)] [date_assignment.role -> date_role date_role.name = 'request date']}) #4-#6 (date_and_time_assigned_item = versioned_action_request date_and_time_assigned_item <- sheet_metal_date_and_time_assignment.items[i] sheet_metal_date_and_time_assignment <= date_and_time_assignment {date_and_time_assignment [date_and_time_assignment.assigned_date_and_time -> date_and_time date_and_time.date_component -> date => #4 (week_of_year_and_day_date) #5 (calendar_date) #6 (ordinal_date)] date_and_time_assignment.role -> date_time_role date_time_role.name = 'request date and time']})

Table 13 - Mapping table for work_request (concluded)

Application element	AIM element	Source	Rules	Reference path
requestor	sheet_metal_person_-assignment	207	16, 23	versioned_action_request person_assigned_item = versioned_action_request person_assigned_item <- sheet_metal_person_assignment.items[i] sheet_metal_person_assignment <= person_assignment {person_assignment person_assignment.role ->person_role person_role.name = 'work requestor'}
work_request_-identification	[versioned_-action_request.id] [versioned_-action_request.version]	41 41		
work_request to work_-order	PATH			versioned_action_request <- action_directive.requests[i] action_directive <- directed_action.directive directed_action <= executed_action => work_order => #1 (change_order) #2 (start_order)

The following rules are referenced in the preceding tables:

1) approval_requires_approval_date_time
2) approval_requires_approval_person_organization
3) cardinality_action_method_to_process_plan
4) cardinality_product_definition_process_to_action_resource_requirement
5) cardinality_sm_action_assignment_to_process_plan
6) cardinality_sm_action_assignment_to_product_definition
7) cardinality_sm_action_assignment_to_product_definition_formation
8) cardinality_sm_organization_assignment_to_work_order
9) cardinality_start_order_to_sm_action_assignment
10) cardinality_work_order_to_sm_contract_assig_or_sm_org_assig
11) executed_action_requires_approval
12) executed_action_requires_date_or_date_and_time
13) executed_action_requires_organization
14) global_uncertainty_assigned_context_constraint
15) process_plan_requires_product_definition
16) product_definition_formation_requires_date_or_date_and_time
17) product_definition_formation_requires_product_definition
18) product_requires_product_definition_formation
19) product_requires_product_type
20) restrict_assembly_component_usage_substitute
21) restrict_date_role
22) restrict_date_time_role
23) restrict_organization_role
24) restrict_person_and_organization_role
25) restrict_person_role
26) restrict_product_definition_context
27) restrict_quantified_assembly_component_usage
28) same_type_for_relationships
29) subtype_mandatory_for_shape_representation
30) subtype_mandatory_work_order
31) work_request_requires_date_or_date_and_time
32) work_request_requires_person

5.2 AIM EXPRESS short listing

This clause specifies the EXPRESS schema that uses elements from the integrated resources and the AICs and contains the types, entity specializations, rules, and functions that are specific to this part of ISO 10303. This clause also specifies modifications to the textual material for constructs that are imported from the integrated resources and the AICs. The definitions and EXPRESS provided in the integrated resources for constructs

used in the AIM may include select list items and subtypes 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 sheet_metal;

USE FROM aic_advanced_brep;

USE FROM aic_csg;

USE FROM aic_edge_based_wireframe;

USE FROM aic_faceted_brep;

USE FROM aic_geometrically_bounded_surface;

USE FROM aic_geometrically_bounded_wireframe;

USE FROM aic_manifold_surface;

USE FROM aic_shell_based_wireframe;

USE FROM application_context_schema
   (application_context,
   application_protocol_definition,
   product_concept_context,
   product_context,
   product_definition_context);

USE FROM product_definition_schema
   (product,
   product_related_product_category,
   product_category_relationship,
   product_definition,
   product_definition_effectivity,
   product_definition_formation,
   product_definition_formation_with_specified_source,
   product_definition_relationship,
   product_definition_substitute,
   product_definition_with_associated_documents);

USE FROM product_property_definition_schema
   (shape_aspect);

USE FROM product_property_representation_schema
   (shape_definition_representation,
   shape_representation,
   shape_representation_relationship);

USE FROM management_resources_schema
   (action_assignment,
   approval_assignment,
   contract_assignment,
   date_assignment,
```

```
   date_and_time_assignment,
   document_reference,
   organization_assignment,
   person_and_organization_assignment,
   person_assignment,
   security_classification_assignment);

USE FROM action_schema
   (action,
   action_method,
   action_relationship,
   action_request_solution,
   action_resource_relationship,
   directed_action,
   executed_action,
   versioned_action_request);

USE FROM approval_schema
   (approval,
   approval_date_time,
   approval_person_organization);

USE FROM contract_schema
   (contract);

USE FROM person_organization_schema
   (organization_role,
   person_and_organization_role,
   person_role);

USE FROM date_time_schema
   (calendar_date,
   date_role,
   date_time_role,
   ordinal_date,
   week_of_year_and_day_date);

USE FROM effectivity_schema
   (dated_effectivity,
   serial_numbered_effectivity);

USE FROM support_resource_schema;

USE FROM measure_schema
   (context_dependent_unit,
   conversion_based_unit,
   derived_unit,
   length_measure_with_unit,
   length_unit,
   plane_angle_measure_with_unit,
   plane_angle_unit,
   si_unit);

USE FROM geometry_schema
   (direction,
   surface);
```

```
USE FROM topology_schema
  (face_surface);

USE FROM geometric_model_schema
  (geometric_set);

USE FROM representation_schema
  (functionally_defined_transformation,
  global_uncertainty_assigned_context,
  item_defined_transformation,
  representation_relationship_with_transformation);

USE FROM product_structure_schema
  (assembly_component_usage_substitute,
  make_from_usage_option_group,
  promissory_usage_occurrence,
  quantified_assembly_component_usage);

USE FROM product_concept_schema;

USE FROM configuration_management_schema
  (configuration_design);

USE FROM material_property_definition_schema
  (material_property);

USE FROM qualified_measure_schema
  (descriptive_representation_item,
  measure_representation_item);

USE FROM shape_dimension_schema
  (angular_location,
  dimensional_characteristic_representation,
  dimensional_location_with_path,
  dimensional_size,
  shape_dimension_representation);

USE FROM shape_tolerance_schema
  (geometric_tolerance_with_datum_reference,
  limits_and_fits,
  modified_geometric_tolerance,
  plus_minus_tolerance,
  tolerance_value,
  tolerance_zone_definition);

USE FROM method_definition_schema
  (action_method_with_associated_documents,
  sequential_method);

USE FROM process_property_representation_schema
  (action_property_representation,
  resource_property_representation);

USE FROM process_property_schema
  (action_resource_requirement,
```

```
   action_resource_requirement_relationship,
   process_product_association,
   product_definition_process,
   replacement_relationship,
   requirement_for_action_resource);
(*
```

NOTES

1- The schemas referenced above may be found in the following parts of ISO 10303:

application_context_schema	ISO 10303-41
product_definition_schema	ISO 10303-41
product_property_definition_schema	ISO 10303-41
product_property_representation_schema	ISO 10303-41
management_resources_schema	ISO 10303-41
action_schema	ISO 10303-41
approval_schema	ISO 10303-41
contract_schema	ISO 10303-41
person_organization_schema	ISO 10303-41
date_time_schema	ISO 10303-41
effectivity_schema	ISO 10303-41
support_resource_schema	ISO 10303-41
measure_schema	ISO 10303-41
geometry_schema	ISO 10303-42
topology_schema	ISO 10303-42
geometric_model_schema	ISO 10303-42
representation_schema	ISO 10303-43
product_structure_schema	ISO 10303-44
product_concept_schema	ISO 10303-44
configuration_management_schema	ISO 10303-44
material_property_definition_schema	ISO/DIS 10303-45:—[1)]
qualified_measure_schema	ISO/DIS 10303-45:—[1)]
shape_dimension_schema	ISO/DIS 10303-47:
shape_tolerance_schema	ISO/DIS 10303-47:
method_definition_schema	ISO/DIS 10303-49:—[1)]
process_property_representation_schema	ISO/DIS 10303-49:—[1)]
process_property_schema	ISO/DIS 10303-49:—[1)]
aic_edge_based_wireframe	Annex F
aic_shell_based_wireframe	Annex F
aic_geometrically_bounded_surface	Annex F
aic_manifold_surface	Annex F
aic_geometrically_bounded_wireframe	Annex F

[1)] To be published.

aic_faceted_brep	Annex F
aic_advanced_brep	Annex F
aic_csg	Annex F

2 - aic_advanced_brep and aic_manifold_surface use the following additional schema:

aic_topology_bounded_surface	Annex F

3 - aic_csg uses the following schemas:

aic_faceted_brep	Annex F
aic_advanced_brep	Annex F

5.2.1 Fundamental concepts and assumptions

The ARM is detailed in clause 4. A number of entities in the ARM are optional, but are mapped to required declarations attributes in the AIM. These AIM attributes may be instantiated as a blank if their base type is STRING; or a zero (0) if their base type is INTEGER or NUMBER or REAL.

5.2.2 Sheet metal types

5.2.2.1 Sheet metal type definitions

5.2.2.1.1 action_assigned_item

An **action_assigned_item** is the particular piece of product data upon which an **action** has been, is or will be taken. The types of product data to which an **action** may be associated shall be at least one of the following: **executed_action**, **process_plan**, **product_definition**, and **product_definition_formation**.

EXPRESS specification:

```
*)
TYPE action_assigned_item = SELECT
  (executed_action,
   process_plan,
   product_definition,
   product_definition_formation);
END_TYPE;
(*
```

5.2.2.1.2 approval_assigned_item

An **approval_assigned_item** is the product data to which **approval** information is assigned. The types of

product data that may have **approval** information assigned shall be at least one of the following: **executed_action**, **process_plan**, **product_definition**, **product_definition_formation**, and **sheet_metal_action_assignment**.

EXPRESS specification:

```
*)
TYPE approval_assigned_item = SELECT
  (executed_action,
   process_plan,
   product_definition,
   product_definition_formation,
   sheet_metal_action_assignment);
END_TYPE;
(*
```

5.2.2.1.3 contract_assigned_item

A **contract_assigned_item** is the particular piece of product data that has associated **contract** information. The types of product data that may have associated contractual information shall be at least one of the following: **change_order**, **executed_action**, **product_definition_formation_with_specified_source**, and **start_order**.

EXPRESS specification:

```
*)
TYPE contract_assigned_item = SELECT
  (change_order,
   executed_action,
   product_definition_formation_with_specified_source,
   start_order);
END_TYPE;
(*
```

5.2.2.1.4 date_assigned_item

A **date_assigned_item** is the product data that has associated **date** information. The types of product data that may have associated **date** information shall be at least one of the following: **executed_action**, **process_plan**, **product_concept**, **product_definition**, **product_definition_formation**, **versioned_action_request**, or **sheet_metal_action_assignment**.

EXPRESS specification:

```
*)
TYPE date_assigned_item = SELECT
  (executed_action,
   process_plan,
```

```
   product_concept,
   product_definition,
   product_definition_formation,
   sheet_metal_action_assignment,
   versioned_action_request);
END_TYPE;
(*
```

5.2.2.1.5 date_and_time_assigned_item

A **date_and_time_assigned_item** is the product data that has associated **date_and_time** information. The types of product data that may have associated **date_and_time** information shall be at least one of the following: **executed_action**, **process_plan**, **product_concept**, **product_definition**, **product_definition_formation**, **versioned_action_request**, or **sheet_metal_action_assignment**.

EXPRESS specification:

```
*)
TYPE date_and_time_assigned_item = SELECT
  (executed_action,
   process_plan,
   product_concept,
   product_definition,
   product_definition_formation,
   sheet_metal_action_assignment,
   versioned_action_request);
END_TYPE;
(*
```

5.2.2.1.6 document_referenced_item

A **document_referenced_item** is the product data that makes reference to a **document**. The types of product data that may reference a document shall be at least one of the following: **action_resource_requirement**, **executed_action**, **process_plan**, **product_definition**, or **sheet_metal_action_assignment**.

EXPRESS specification:

```
*)
TYPE document_referenced_item = SELECT
  (action_resource_requirement,
   executed_action,
   process_plan,
   product_definition,
   sheet_metal_action_assignment);
END_TYPE;
(*
```

5.2.2.1.7 organization_assigned_item

An **organization_assigned_item** is the product data that has associated information about an **organization**. The types have product data that may have associated **organization** information shall be at least one of the following: **change_order**, **contract**, **executed_action**, **product_definition**, and **start_order**.

EXPRESS specification:

```
*)
TYPE organization_assigned_item = SELECT
  (change_order,
   contract,
   executed_action,
   product_definition,
   start_order);
END_TYPE;
(*
```

5.2.2.1.8 person_and_organization_assigned_item

A **person_and_organization_assigned_item** is the product data that has associated information about a **person_and_organization**. The types of product data that may have associated **person_and_organization** information shall be at least one of the following: **change_order**, **process_plan**, and **product_definition**.

EXPRESS specification:

```
*)
TYPE person_and_organization_assigned_item = SELECT
  (change_order,
   process_plan,
   product_definition);
END_TYPE;
(*
```

5.2.2.1.9 person_assigned_item

A **person_assigned_item** is the product data that has associated information about a **person**. The types of product data that may have associated **person_and_organization** information shall be at least one of the following: **change_order**, **contract**, **product_definition**, **versioned_action_request**, and **shape_aspect**.

EXPRESS specification:

```
*)
TYPE person_assigned_item = SELECT
  (change_order,
   contract,
   product_definition,
   shape_aspect,
   versioned_action_request);
```

```
END_TYPE;
(*
```

5.2.2.1.10 security_classification_assigned_item

A **security_classification_assigned_item** is the product data that has associated **security_classification** information. The types of product data that may have associated **security_classification** information shall be at least one of the following: **product_definition** or **process_plan**.

EXPRESS specification:

```
*)
TYPE security_classification_assigned_item = SELECT
  (process_plan,
   product_definition);
END_TYPE;
(*
```

5.2.2 Die planning and design imported type modifications

5.2.2.1 characterized_definition

The base definition of the type **characterized_definition** is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

The select list has been pruned. The list entry **characterized_object** has been removed.

5.2.2.2 measure_value

The base definition of the type **measure_value** is given in ISO 10303-41 The following modifications apply to this part of ISO 10303.

The select list has been pruned. All list entries except for **count_measure, length_measure, mass_measure, parameter_value, plane_angle_measure, positive_length_measure**, and **positive_plane_angle_measure** have been removed.

5.2.3 Sheet_metal entities

5.2.3.1 Sheet metal entity definitions

5.2.3.1.1 change_order

A **change_order** is the formal notification that authority has been given to perform an **action**. This is the

same as a **work_order** except that the **work_order** is for changing something for which a prior version exists.

EXPRESS specification:

```
*)
ENTITY change_order
  SUBTYPE OF (work_order);
END_ENTITY;
(*
```

5.2.3.1.2 die_definition_constraint_relationship

A **die_definition_constraint_relationship** relates the design of a die to a description of the items that place constraints upon the design of the die. This **die_definition_constraint_relationship** specifically points out the designs of dies that meet or incorporate defined constraints.

EXPRESS specification:

```
*)
ENTITY die_definition_constraint_relationship
  SUBTYPE OF (product_definition_relationship);
WHERE
  WR1: SELF\product_definition_relationship.
   related_product_definition.frame_of_reference.
   life_cycle_stage = 'as design constrained';
  WR2: product_type_of_product
    ( SELF.relating_product_definition.formation.of_product, product_type)
    .name IN       ['die'];
  WR3: product_type_of_product
    ( SELF.related_product_definition.formation.of_product, product_type)
    .name IN       ['die'];
END_ENTITY;
(*
```

Formal propositions:

WR1: The **die_definition_constraint_relationship.related_product_definition** represents the description of the constrains of a die and thus shall have the definitional viewpoint of "as design constrained".

WR2: For each instance of the **die_definition_constraint_relationship** entity, the **product_type** of the **die_-definition_constraint_relationship.relating_product_definition** shall be "die".

WR3: For each instance of the **die_definition_constraint_relationship** entity, the **product_type** of the **die_-definition_constraint_relationship.related_product_definition** shall be "die".

5.2.3.1.3 input_item_die_relationship

An **input_item_die_relationship** relates the description of a part or material to a description of a die. The relationship specifically describes how a part or material is placed within a die for the purpose of processing the part or material in a product stamping operation.

EXPRESS specification:

```
*)
ENTITY input_item_die_relationship
  SUBTYPE OF (product_definition_relationship);
WHERE
  WR1: product_type_of_product
  ( SELF.relating_product_definition.formation.of_product, product_type)
  .name = 'die';
  WR2: product_type_of_product
  ( SELF.related_product_definition.formation.of_product, product_type)
      .name IN ['material', 'part'];
END_ENTITY;
(*
```

Attribute definitions:

SELF.relating_product_definition: the design or description of a die.

SELF.related_product_definition: the design or description of a part or material.

Formal propositions:

WR1: For each instance of the **input_item_die_relationship** entity, the **product_type** of the **input_item_-die_relationship.relating_product_definition** shall be "die".

WR2: For each instance of the **input_item_die_relationship** entity, the **product_type** of the **input_item_-die_relationship.related_product_definition** shall be either "part" or "material".

5.2.3.1.4 physically_modelled_product_definition

A **physically_modelled_product_definition** is the design or description of a product in terms of a "hard" model as opposed to an electronic model. "Hard" models are things such as wood models, plaster models, clay models and other such models of a product. The **physically_modelled_product_definition** is the "hard" model description of a product.

EXPRESS specification:

```
*)
ENTITY physically_modelled_product_definition
  SUBTYPE OF (product_definition);
WHERE
  WR1: SELF\product_definition.frame_of_reference.
       life_cycle_stage = 'as physically modelled';
  WR2: SIZEOF( USEDIN (SELF, 'SHEET_METAL.' +
                'PRODUCT_DEFINITION_RELATIONSHIP.' +
                'RELATED_PRODUCT_DEFINITION')) > 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: The **physically_modelled_product_definition** shall be defined from the life cycle point of view of "as physically modelled".

WR2: The **physically_modelled_product_definition** shall be related to another design or description of the **product**.

5.2.3.1.5 process_plan

A **process_plan** is the description of a manufacturing process for producing a discrete sheet metal part. In this case a discrete sheet metal part is a product made of sheet metal that is not the result of an assembly type of process. The sheet metal part produced by the described manufacturing process can be one that is intended for use in an assemblage of parts or one that will be further refined in a subsequent set of processes that remove or alter sheet metal material.

EXPRESS specification:

```
*)
ENTITY process_plan
  SUBTYPE OF (action);
END_ENTITY;
(*
```

Associated global rules:

The following global rules are associated with this entity and restrict its use or its relationships with other entities:

cardinality_action_method_to_process_plan: The **cardinality_action_method_to_process_plan** rule specifies that **action_method**s with name attributes equal to “part porcess plan template” shall be referenced by at least one **process_plan**. This rule enforces the ARM requirement for part_process_plan_templates to

be used by no fewer than one **part_process_plan**.

cardinality_sm_action_assignment_to_process_plan: The **cardinality_sm_action_assignment_to_process_-plan** rule specifies that a **sheet_metal_action_assignment** shall correspond to no more than one **process_plan** through its items attribute. This rule enforces the requirement in the ARM that a work_item be a supertype of a part_process_plan, and therefore associated with only zero or one of them at a time.

process_plan_requires_product_definition: The **process_plan_requires_product_definition** rule specifies that each instance of a **process_plan** shall be related to exactly one instance of the **product_definition** entity that describes the sheet metal part that the manufacturing process produces.

5.2.3.1.6 product_type

A **product_type** is a **product_related_product_category** that categorizes a **product** as being a "part", "die", "material" or an "unspecified" item. The **product_type** is used to categorize all product items within the domain of this part of ISO 10303.

EXPRESS specification:

```
*)
ENTITY product_type
  SUBTYPE OF (product_related_product_category);
WHERE
  WR1: SIZEOF (USEDIN (SELF,'SHEET_METAL.PRODUCT_CATEGORY_RELATIONSHIP.'
         + 'RELATING_PRODUCT_CATEGORY ') + USEDIN (SELF,
         'SHEET_METAL.PRODUCT_CATEGORY_RELATIONSHIP.' +
         'RELATED_PRODUCT_CATEGORY ')) = 0;
  WR2: SELF.name IN [ 'die', 'part', 'material', 'unspecified'];
END_ENTITY;
(*
```

Formal propositions:

WR1: A **product_type** is used to classify the base types of products in this part of ISO 10303 and shall not be instantiated within a **product_category_relationship** entity.

WR2: The allowable **product_category.name**s for a **product_type** are "part", "die", "material" and "unspecified".

Associated global rules:

The following global rules are associated with this entity and restrict its use or its relationships with other entities:

product_requires_product_type: The **product_requires_product_type** rule specifies that each instance of a **product** shall be related to exactly one instance of the **product_type** entity. This rule enforces the requirement all products are categorized as being a certain base type.

restrict_assembly_component_usage_substitute: The **restrict_assembly_component_usage_substitute** rule specifies that the two **assembly_component_usage** instances that are related shall be of the type **quantified_assembly_component_usage** and the **product_type**s in the **base** and **substitute assembly_component_usage**s shall be of the same base product type.

restrict_die_definition_constraint_relationship: The **restrict_die_definition_constraint_relationship** rule constrains each instance of the **die_definition_constraint_relationship** entity such that both **product_definition**s referenced in each of the instances are for a "die" product.

restrict_input_item_die_relationship: The **restrict_input_item_die_relationship** rule constrains the **input_item_die_relationship.relating_product_definition** describes a "die" product while the **input_item_die_relationship.related_product_definition** describes a "part" or "material" product.

restrict_quantified_assembly_component_usage: The **restrict_quantified_assembly_component_usage** rule specifies that the **product**s that are related by each **quantified_assembly_component_usage** entity shall each correspond to the same type of product.

same_type_for_relationships: The **same_type_for_relationships** rule specifies that the **product**s that are related by each **product_definition_relationship** entity representing a mating or symmetrical relationship shall each correspond to the same type of product.

5.2.3.1.7 sequenced_product_definition_relationship

A **sequenced_product_definition_relationship** relates two **product_definition**s representing prior and resultant definitions in the context of an exchange. One **product_definition** exists prior to the exchange of it. The other **product_definition** is the resultant form of the prior **product_definition** after it has been exchanged.

> EXAMPLE 44 - A product_definition may exist in the proprietary format of CAD system A and reside in a computer file on a tape. It may then be translated into the proprietary format of CAD system B and transmitted to a different location where it resides in a file on a hard drive. A sequenced_product_definition_relationship may be used to show the relationship between these two product_definitions, that are not identical to each other.

EXPRESS specification:

```
*)
ENTITY sequenced_product_definition_relationship
  SUBTYPE OF (product_definition_relationship);
WHERE
  WR1: (('SHEET_METAL.' +
        'PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS') IN (TYPEOF (
        SELF\product_definition_relationship.
        relating_product_definition) AND ('SHEET_METAL.' +
        'PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS'))) IN TYPEOF (
        SELF\product_definition_relationship.related_product_definition);
END_ENTITY;
(*
```

Attribute definitions:

SELF\product_definition_relationship.relating_product_definition: the **product_definition** that is the design organization's definition and identification of the product.

SELF\product_definition_relationship.related_product_definition: the **product_definition** that is the exchanged definition and identification of the product.

Formal propositions:

WR1: The **product_definition**s referenced by the **sequenced_product_definition_relationship** entity shall have associated documentation.

5.2.3.1.8 sheet_metal_action_assignment

A **sheet_metal_action_assignment** is an association of an **action** with product data. The association of an **action** with product data indicates a process that has been or will be carried out with respect to the product data.

EXPRESS specification:

```
*)
ENTITY sheet_metal_action_assignment
  SUBTYPE OF (action_assignment);
  items  : SET [1:?] OF action_assigned_item;
WHERE
  WR1: 'SHEET_METAL.EXECUTED_ACTION' IN
       TYPEOF (SELF\action_assignment.assigned_action);
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **action_assigned_item**s that identify the particular pieces of product data to which the **action** is associated.

Formal propositions:

WR1: The type of **action** assigned by the **sheet_metal_action_assignment** shall be an **executed_action** to indicate that the only types of actions associated with product data are those that have been or actually will be carried out.

Associated global rule:

The following global rule is associated with this entity and restrict its use or its relationships with other entities:

cardinality_sm_action_assignment_to_process_plan: The **cardinality_sm_action_assignment_to_process_-plan** rule specifies that a **sheet_metal_action_assignment** shall correspond to no more than one **process_plan** through its items attribute. This rule enforces the requirement in the ARM that a work_item be a supertype of a part_process_plan, and therefore associated with only zero or one of them at a time.

cardinality_sm_action_assignment_to_product_definition: The **cardinality_sm_action_assignment_to_-product_definition** rule specifies that a **sheet_metal_action_assignment** shall correspond to no more than one **product_definition** through its items attribute. This rule enforces the requirement in the ARM that a work_item be a supertype of an item_definition, and therefore associated with only zero or one of them at a time.

cardinality_sm_action_assignment_to_product_definition_formation: The **cardinality_sm_action_-assignment_to_product_definition_formation** rule specifies that a **sheet_metal_action_assignment** shall correspond to no more than one **product_definition_formation** through its items attribute. This rule enforces the requirement in the ARM that a work_item be a supertype of an item_version, and therefore associated with only zero or one of them at a time.

The **cardinality_start_order_to_sm_action_assignment** rule specifies that a **start_order** shall correspond to no fewer than one **sheet_metal_action_assignment** through its assigned_action attribute. This rule enforces the requirement in the ARM that a work_order has one or more work_items.

process_plan_requires_product_definition: The **process_plan_requires_product_definition** rule specifies that each instance of a **process_plan** shall be related to exactly one instance of the **product_definition** entity by means of a **sheet_metal_action_assignment**. This rule enforces the requirement that each **process_plan** be the process for manufacturing exactly one **product**.

5.2.3.1.9 sheet_metal_approval_assignment

A **sheet_metal_approval_assignment** associates an **approval** to product data. This association points out what type of approval has been given to product data.

EXPRESS specification:

```
*)
ENTITY sheet_metal_approval_assignment
  SUBTYPE OF (approval_assignment);
  items  : SET [1:?] OF approval_assigned_item;
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **approval_assigned_item**s that identify the pieces of product data that may have **approval** information associated.

Associated global rules:

The following global rules are associated with this entity and restrict its use or its relationships with other entities:

executed_action_requires_approval: The **executed_action_requires_approval** rule specifies that each instance of an **executed_action** shall be related to one or more instances of the **approval** entity by means of the **sheet_metal_approval_assignment** entity. This rule specifies the need for every **executed_action** to have an associated approval of approval or approvals.

5.2.3.1.10 sheet_metal_contract_assignment

A **sheet_metal_contract_assignment** relates a **contract** to product data. This relationship outlines the role that a **contract** plays with respect to product data.

EXPRESS specification:

```
*)
ENTITY sheet_metal_contract_assignment
  SUBTYPE OF (contract_assignment);
  items  : SET [1:?] OF contract_assigned_item;
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **contract_assigned_item**s that identify the particular pieces of product data to which a **contract** applies.

Associated global rule:

The following global rule is associated with this entity and restricts its use or its relationships with other entities:

cardinality_work_order_to_sm_contract_assign_or_sm_org_assign: The **cardinality_work_order_-to_sm_contract_assign_or_sm_org_assign** rule specifies that each instance of **work_order** shall be a **contract_assigned_item** or an **organization_assigned_item**. This rule enforces the requirement for every **work_order** to be external, which is equivalent to it being the subject of a **sheet_metal_contract_assignment**, or internal, which is equivalent to it being the subject of a **sheet_metal_organization_assignment**.

5.2.3.1.11 sheet_metal_date_and_time_assignment

A **sheet_metal_date_and_time_assignment** relates **date_and_time** information to product data. The assignment of a **date_and_time** to product data are an indication of when something occurred or will occur relative to the specific piece or pieces of product data.

EXPRESS specification:

```
*)
ENTITY sheet_metal_date_and_time_assignment
  SUBTYPE OF (date_and_time_assignment);
  items  : SET [1:?] OF date_and_time_assigned_item;
WHERE
  WR1: sheet_metal_date_correlation (SELF);
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **date_and_time_assigned_item**s that identify the particular pieces of product data to which the date is assigned.

Formal propositions:

WR1: The **sheet_metal_date_correlation** function, that correlates the roles that dates may play relative to the various types of product data, shall be satisfied.

Associated global rules:

The following global rules are associated with this entity and restrict its use or its relationships with other entities:

executed_action_requires_date_or_date_and_time: The **executed_action_requires_date_or_date_and_-**

time rule specifies that each instance of **executed_action** shall be related to exactly one instance of the **date** entity by means of the **sheet_metal_date_assignment** entity, or exactly one instance of the **date_and_time** entity by means of the **sheet_metal_date_and_time_assignment** entity. This rule specifies the need for every **executed_action** to have exactly one associated date, and optionally exactly one associated time.
product_definition_formation_requires_date_or_date_and_time: The **product_definition_formation_-requires_date_or_date_and_time** rule specifies that each instance of **product_definition_formation** shall be related to exactly one instance of the **date** entity by means of the **sheet_metal_date_assignment** entity, or exactly one instance of the **date_and_time** entity by means of the **sheet_metal_date_and_time_assignment** entity. This rule specifies the need for every **product_definition_formation** to have exactly one associated date, and optionally exactly one associated time.

work_request_requires_date_or_date_and_time: The **work_request_requires_date_or_date_and_time** rule specifies that each instance of **versioned_action_request** shall be related to exactly one instance of the **date** entity by means of the **sheet_metal_date_assignment** entity, or exactly one instance of the **date_and_time** entity by means of the **sheet_metal_date_and_time_assignment** entity. This rule specifies the need for every **versioned_action_request** to have exactly one associated date, and optionally exactly one associated time.

5.2.3.1.12 sheet_metal_date_assignment

A **sheet_metal_date_assignment** relates **date** information to product data. The assignment of a **date** to product data are an indication of when something occurred or will occur relative to the specific piece or pieces of product data.

EXPRESS specification:

```
*)
ENTITY sheet_metal_date_assignment
  SUBTYPE OF (date_assignment);
  items  : SET [1:?] OF date_assigned_item;
WHERE
  WR1: sheet_metal_date_correlation (SELF);
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **date_assigned_item**s that identify the particular pieces of product data to which the date is assigned.

Formal propositions:

WR1: The **sheet_metal_date_correlation** function, that correlates the roles that dates may play relative to the various types of product data, shall be satisfied.

Associated global rules:

The following global rules are associated with this entity and restrict its use or its relationships with other entities:

executed_action_requires_date_or_date_and_time: The **executed_action_requires_date_or_date_and_-time** rule specifies that each instance of **executed_action** shall be related to exactly one instance of the **date** entity by means of the **sheet_metal_date_assignment** entity, or exactly one instance of the **date_and_time** entity by means of the **sheet_metal_date_and_time_assignment** entity. This rule specifies the need for every **executed_action** to have exactly one associated date, and optionally exactly one associated time.

product_definition_formation_requires_date_or_date_and_time: The **product_definition_formation_-requires_date_or_date_and_time** rule specifies that each instance of **product_definition_formation** shall be related to exactly one instance of the **date** entity by means of the **sheet_metal_date_assignment** entity, or exactly one instance of the **date_and_time** entity by means of the **sheet_metal_date_and_time_assignment** entity. This rule specifies the need for every **product_definition_formation** to have exactly one associated date, and optionally exactly one associated time.

work_request_requires_date_or_date_and_time: The **work_request_requires_date_or_date_and_time** rule specifies that each instance of **versioned_action_request** shall be related to exactly one instance of the **date** entity by means of the **sheet_metal_date_assignment** entity, or exactly one instance of the **date_and_time** entity by means of the **sheet_metal_date_and_time_assignment** entity. This rule specifies the need for every **versioned_action_request** to have exactly one associated date, and optionally exactly one associated time.

5.2.3.1.13 sheet_metal_document_reference

A **sheet_metal_document_reference** relates a **document** to a particular piece of product data. This relationship indicates a document reference from a specific piece of product data.

EXPRESS specification:

```
*)
ENTITY sheet_metal_document_reference
  SUBTYPE OF (document_reference);
  items : SET [1:?] OF document_referenced_item;
WHERE
  WR1: sheet_metal_document_correlation (SELF);
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **document_referenced_item**s that identify the particular pieces of product data that reference the **document**.

Formal propositions:

WR1: The **sheet_metal_document_correlation** function, that correlates the types of documents that may be associated with specific pieces of product data, shall be satisfied.

5.2.3.1.14 sheet_metal_organization_assignment

A **sheet_metal_organization_assignment** associates an **organization** with product data. This association shows the role that an **organization** plays relative to the product data.

EXPRESS specification:

```
*)
ENTITY sheet_metal_organization_assignment
  SUBTYPE OF (organization_assignment);
  items  : SET [1:?] OF organization_assigned_item;
WHERE
  WR1: sheet_metal_organization_correlation (SELF);
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **organization_assigned_item**s that identify the particular pieces of product data in which an **organization** plays a role.

Formal propositions:

WR1: The **sheet_metal_organization_correlation** function that correlates roles of organizations to elements of product data shall be satisfied.

Associated global rules:

The following global rules are associated with this entity and restrict its use or its relationships with other entities:

cardinality_work_order_to_sm_contract_assig_or_sm_org_assig: The **cardinality_work_order_to_sm_-contract_assig_or_sm_org_assig** rule specifies that each instance of **work_order** shall be a **contract_-assigned_item** or an **organization_assigned_item**. This rule enforces the requirement for every **work_order** to be external, which is equivalent to it being the subject of a **sheet_metal_contract_assignment**, or internal, which is equivalent to it being the subject of a **sheet_metal_organization_assignment**.

cardinality_sm_organization_assignment_to_work_order: The **cardinality_sm_organization_-assignment_to_work_order** rule specifies that a **sheet_metal_organization_assignment** shall correspond to no fewer than one **work_order** through its items attribute. This rule enforces the requirement in the ARM that a **work_order_responsibility** is responsible for one or more **work_orders**.

executed_action_requires_organization: The **executed_action_requires_organization** rule specifies that each instance of **executed_action** shall be related to one or more instances of the **organization** entity by means of the **sheet_metal_organization_assignment** entity. This rule specifies the need for every **executed_action** to have an associated organization or organization or organizations.

5.2.3.1.15 sheet_metal_person_and_organization_assignment

A **sheet_metal_person_and_organization_assignment** relates a **person_and_organization** to product data. This association shows the role that a **person_and_organization** plays relative to the product data.

EXPRESS specification:

```
*)
ENTITY sheet_metal_person_and_organization_assignment
  SUBTYPE OF (person_and_organization_assignment);
  items  : SET [1:?] OF person_and_organization_assigned_item;
WHERE
  WR1: sheet_metal_person_and_organization_correlation (SELF);
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **person_and_organization_assigned_item**s that identify the particular pieces of product data to which the **person_and_organization** is assigned.

Formal propositions:

WR1: The **sheet_metal_person_and_organization_correlation** function that correlates roles of persons and organizations to elements of product data shall be satisfied.

5.2.3.1.16 sheet_metal_person_assignment

A **sheet_metal_person_assignment** relates a **person** to product data. This association shows the role that a **person** plays relative to the product data.

EXPRESS specification:

```
*)
ENTITY sheet_metal_person_assignment
  SUBTYPE OF (person_assignment);
  items  : SET [1:?] OF person_assigned_item;
WHERE
  WR1: sheet_metal_person_correlation (SELF);
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **person_assigned_item**s that identify the pieces of product data to which the person is assigned.

Formal propositions:

WR1: The **sheet_metal_person_correlation** function that correlates roles of persons to elements of product data shall be satisfied.

Associated global rule:

The following global rule is associated with this entity and restrict its use or its relationships with other entities:

work_request_requires_person: The **work_request_requires_person** rule specifies that each instance of **versioned_action_request** shall be related to one or more instances of the **person** entity by means of the **sheet_metal_person_assignment** entity. This rule specifies the need for every **versioned_action_request** to have an individual or individuals who is the work requestor.

5.2.3.1.17 sheet_metal_security_classification_assignment

A **sheet_metal_security_classification_assignment** relates a **security_classification** to a particular piece of product data. This particular relationship indicates security information relative to the product data to which the **security_classification** is assigned.

EXPRESS specification:

```
*)
ENTITY sheet_metal_security_classification_assignment
  SUBTYPE OF (security_classification_assignment);
  items  : SET [1:?] OF security_classification_assigned_item;
END_ENTITY;
(*
```

Attribute definitions:

items: a set of **security_classification_assigned_item**s that identify the particular pieces of product data to which the security classification is assigned.

5.2.3.1.18 start_order

A **start_order** is the formal notification that authority has been given to perform an **action**. This is the same as a **work_order** except that it is only for creating something for which no prior version exists.

EXPRESS specification:

```
*)
ENTITY start_order
  SUBTYPE OF (work_order);
END_ENTITY;
(*
```

Associated global rule:

The following global rule is associated with this entity and restrict its use or its relationships with other entities:

cardinality_start_order_to_sm_action_assignment: The **cardinality_start_order_to_sm_action_assignment** rule specifies that a **start_order** shall correspond to no fewer than one **sheet_metal_action_assignment** through its assigned_action attribute. This rule enforces the requirement in the ARM that a **work_order** has one or more **work_item**s.

5.2.3.1.19 work_order

A **work_order** is the formal notification that authority has been given to perform an **action**. This is the same as an **executed_action** except that uniqueness is enforced on the value of the name attribute.

EXPRESS specification:

```
*)
ENTITY work_order
  SUPERTYPE OF (ONEOF (change_order, start_order))
  SUBTYPE OF (executed_action);
UNIQUE
  UR1:name;
END_ENTITY;
(*
```

Attribute definitions:

SELF\action.name: a unique identifier for the **work_order**.

Formal propositions:

WR1: The value assigned to the inherited name attribute of **work_order** shall be unique among all instances of **work_order**.

Associated global rule:

The following global rule is associated with this entity and restricts its use or its relationships with other entities:

cardinality_sm_organization_assignment_to_work_order: The **cardinality_sm_organization_-assignment_to_work_order** rule specifies that a **sheet_metal_organization_assignment** shall correspond to no fewer than one **work_order** through its items attribute. This rule enforces the requirement in the ARM that a **work_order_responsibility** is responsible for one or more **work_order**s.

cardinality_work_order_to_sm_contract_assig_or_sm_org_assig: The **cardinality_work_order_to_sm_-contract_assig_or_sm_org_assig** rule specifies that each instance of **work_order** shall be a **contract_-assigned_item** or an **organization_assigned_item**. This rule enforces the requirement for every **work_order** to be external, which is equivalent to it being the subject of a **sheet_metal_contract_assignment**, or internal, which is equivalent to it being the subject of a **sheet_metal_organization_assignment**.

subtype_mandatory_work_order: The **subtype_mandatory_work_order** rule specifies that each instance of **work_order** shall be an instance of **change_order** or **start_order**. The rule enforces the requirement that work orders must be for starting a new item or changing an existing one.

5.2.3.1.20 work_order_relationship

A **work_order_relationship** is a type of **action_relationship** that is further constrained so that the two **action**s related thereby are each either **change_order**s or **start_order**s.

EXPRESS specification:

```
*)
ENTITY work_order_relationship
  SUBTYPE OF (action_relationship);
WHERE
  WR1:(('CHANGE_ORDER' IN (TYPEOF(SELF\action_relationship.
   relating_action)  OR 'START_ORDER')) IN TYPEOF(SELF\
   action_relationship.relating_action)) AND (('CHANGE_ORDER'
   IN (TYPEOF(SELF\action_relationship.related_action )  OR
   'START_ORDER')) IN TYPEOF (SELF\action_relationship.
   related_action ));
END_ENTITY;
(*
```

Formal propositions:

WR1: The types of the two **action**s related by this relationship shall each be either **change_order**s or **start_-order**s.

5.2.3.2 Die planning and design imported entity modifications

5.2.3.2.1 action

The base definition of the **action** is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

The following global rules defined in this part of ISO 10303 apply to the action entity:

— dependent_instantiation_for_action (see 5.2.4.12).

5.2.3.2.2 action_method

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

The following global rules defined in this part of ISO 10303 apply to the **action_method** entity:

— cardinality_action_method_to_process_plan (see 5.2.4.4).

5.2.3.2.3 action_request_solution

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

The following global rules defined in this part of ISO 10303 apply to the **action_request_solution** entity:

— dependent_instantiation_for_action_request_solution (see 5.2.4.13).

5.2.3.2.4 action_resource_requirement

The base definintion of the **action_resource_requirement** entity is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

The following global rules defined in this part of ISO 10303 apply to the **action_resource_requirement** entity:

— cardinality_product_definition_process_to_action_resource_requirement (see 5.2.4.5).

5.2.3.2.5 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.

The following global rules defined in this part of ISO 10303 apply to the **application_context** entity:

— dependent_instantiation_for_application_context (see 5.2.4.14).

5.2.3.2.6 approval

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

The following global rules defined in this part of ISO 10303 apply to the **approval** entity:

— approval_requires_approval_date_time (see 5.2.4.2).

— approval_requires_approval_person_organization (see 5.2.4.3).

5.2.3.2.7 approval_date_time

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

The following global rules defined in this part of ISO 10303 apply to the **approval_date_time** entity:

— approval_requires_approval_date_time (see 5.2.4.2).

5.2.3.2.8 approval_person_organization

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

The following global rules defined in this part of ISO 10303 apply to the **approval_person_organization** entity:

— approval_requires_approval_person_organization (see 5.2.4.3).

5.2.3.2.9 contract

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

The following global rules defined in this part of ISO 10303 apply to the **contract** entity:

— dependent_instantiation_for_contract (see 5.2.4.15).

5.2.3.2.10 date_role

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

The following global rules defined in this part of ISO 10303 apply to the **date_role** entity:

— dependent_instantiation_for_date_role (see 5.2.4.16).

— restrict_date_role (see 5.2.4.36).

5.2.3.2.11 date_time_role

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

The following global rules defined in this part of ISO 10303 apply to the **date_time_role** entity:

— dependent_instantiation_for_date_time_role (see 5.2.4.17).

— restrict_date_time_role (see 5.2.4.37).

5.2.3.2.12 derived_unit

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

The following global rules defined in this part of ISO 10303 apply to the **derived_unit** entity:

— dependent_instantiation_for_derived_unit (see 5.2.4.18).

5.2.3.2.13 document_reference

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

The following global rules defined in this part of ISO 10303 apply to the **document_reference** entity:

— dependent_instantiation_for_document_reference (see 5.2.4.19).

5.2.3.2.14 executed_action

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

The following global rules defined in this part of ISO 10303 apply to the **executed_action** entity:

— executed_action_requires_approval (see 5.2.4.26).

— executed_action_requires_date_or_date_and_time (see 5.2.4.27).

— executed_action_requires_organization (see 5.2.4.28).

5.2.3.2.15 functionally_defined_transformation

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

The following global rules defined in this part of ISO 10303 apply to the **functionally_defined_transformation** entity:

— dependent_instantiation_for_functionally_defined_transformation (see 5.2.4.20).

5.2.3.2.16 global_uncertainty_assigned_context

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

The following global rules defined in this part of ISO 10303 apply to the **global_uncertainty_assigned**_context entity:

— global_uncertainty_assigned_context_constraint (see 5.2.4.29).

5.2.3.2.17 item_defined_transformation

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

The following global rules defined in this part of ISO 10303 apply to the **item_defined_transformation** entity:

— dependent_instantiation_for_item_defined_transformation (see 5.2.4.21).

5.2.3.2.18 organization_role

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

The following global rules defined in this part of ISO 10303 apply to the **organization_role** entity:

— dependent_instantiation_for_organization_role (see 5.2.4.22).

— restrict_organization_role (see 5.2.4.38).

5.2.3.2.19 person_and_organization_role

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

The following global rules defined in this part of ISO 10303 apply to the **person_and_organization_role** entity:

— dependent_instantiation_for_person_and_organization_role (see 5.2.4.23).

— restrict_person_and_organization_role (see 5.2.4.40).

5.2.3.2.20 person_role

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

The following global rules defined in this part of ISO 10303 apply to the **person_role** entity:

— dependent_instantiation_for_person_role (see 5.2.4.24).

— restrict_person_role (see 5.2.4.39).

5.2.3.2.21 product

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

The following global rules defined in this part of ISO 10303 apply to the **product** entity:

— product_requires_product_definition_formation (see 5.2.4.33).

— product_requires_product_type (see 5.2.4.34).

5.2.3.2.22 product_definition

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

The following global rules defined in this part of ISO 10303 apply to the **product_definition entity**:

— cardinality_sm_action_assignment_to_product_definition (see 5.2.4.7).

— product_definition_formation_requires_product_definition (see 5.2.4.32).

The following additional informal propositions apply:

Instances of **product_definition** which are pointed to by one or more instances of **product_related_product_-category** which are also instances of the **product_related_product_category** subtype, **product_type**, when that **product_related_product_category** has a name attribute with the value of either 'die', 'material', 'part', or 'unspecified' shall always meet one of the four following conditions:

— if the value of their associated **product_related_product_category** name attribute is 'part', they shall also have an instance of **product_related_product_category** pointing to them which is not an instance of the subtype **product_type**, and which has a name attribute with the value of 'part_definition';

— if the value of their associated **product_related_product_category** name attribute is 'die', they shall also be an instance of **product_definition_with_associated_documents,** and also have an instance of **product_related_product_category** pointing to them which is not an instance of the subtype product_type, and which has a name attribute with the value of 'die definition';

— if the value of their associated **product_related_product_category** name attribute is 'material', they shall also be an instance of **product_definition_with_associated_documents,** and also have an instance of **product_related_product_category** pointing to them which is not an instance of the subtype

product_type, and which has a name attribute with the value of 'material definition' or 'scrap definition';

— if the value of their associated **product_related_product_category** name attribute is 'unspecified', they shall also be an instance of **product_definition_with_associated_documents,** and also have an instance of **product_related_product_category** pointing to them which is not an instance of the subtype **product_type**, and which has a name attribute with the value of 'scrap definition'.

5.2.3.2.23 product_definition_context

The base definition of the **product_definition_context** is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

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

— restrict_product_definition_context (see 5.2.4.41).

5.2.3.2.24 product_definition_formation

The base definition of the **product_definition_formation** is given in ISO 10303-41. The following modifications apply to this part of ISO 10303.

The following global rules defined in this part of ISO 10303 apply to the **product_definition_formation** entity:

— cardinality_sm_action_assignment_to_product_definition_formation (see 5.2.4.8).

— product_definition_formation_requires_date_or_date_and_time (see 5.2.4.31).

— product_requires_product_definition_formation (see 5.2.4.33).

— product_definition_formation_requires_product_definition (see 5.2.4.32).

5.2.3.2.25 product_definition_process

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

The following global rules defined in this part of ISO 10303 apply to the **product_definition_process** entity:

— cardinality_product_definition_process_to_action_resource_requirement (see 5.2.4.5).

5.2.3.2.26 product_definition_relationship

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

The following global rules defined in this part of ISO 10303 apply to the **product_definition_relationship** entity:

— same_type_for_relationships (see 5.2.4.43).

The following additional informal propositions apply:

If the **product_definition_relationship** is a mating relationship, then it shall have at most one **shape_aspect** that specifies the shape of the mating area.

5.2.3.2.27 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.

The following global rules defined in this part of ISO 10303 apply to the **shape_representation** entity:

— subtype_mandatory_for_shape_representation (see 5.2.4.44).

5.2.3.2.28 tolerance_value

The base definition of the **tolerance_value** entity is given in ISO 10303-47. The following modifications apply to this part of ISO 10303.

The following global rules defined in this part of ISO 10303 apply to the **tolerance_value** entity:

— dependent_instantiation_for_tolerance_value (see 5.2.4.25).

5.2.3.2.29 versioned_action_request

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

The following global rules defined in this part of ISO 10303 apply to the **versioned_action_request** entity:

— work_request_requires_date_or_date_and_time (see 5.2.4.46).

— work_request_requires_person (see 5.2.4.47).

5.2.3.2.30 item_defined_transformation

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

The following global rules defined in this part of ISO 10303 apply to the **item_defined_transformation** entity:

— dependent_instantiation_for_item_defined_transformation (see 5.2.4.21).

5.2.3.2.31 assembly_component_usage

The base definition of the entity **assembly_component_usage** is given in ISO 10303-44. The following modifications apply to this part of ISO 10303.

The supertype clause has been pruned. The list entries **next_assembly_usage_occurrence** and **specified_higher_usage_occurrence**, the keyword ONEOF, and the brackets surrounding the ONEOF list have been removed.

5.2.3.2.32 assembly_component_usage_substitute

The base definition of the **assembly_component_usage_substitute** entity is given in ISO 10303-44. The following modifications apply to this part of ISO 10303.

The following global rules defined in this part of ISO 10303 apply to the **assembly_component_usage_substitute** entity:

— restrict_assembly_component_usage_substitute (see 5.2.4.35).

5.2.3.2.33 quantified_assembly_component_usage

The base definition of the **quantified_assembly_component_usage** entity is given in ISO 10303-44. The following modifications apply to this part of ISO 10303.

The following global rules defined in this part of ISO 10303 apply to the **quantified_assembly_component_usage** entity:

— restrict_quantified_assembly_component_usage (see 5.2.4.42).

5.2.3.2.34 datum_reference

The base definition of the entity **datum_reference** is given in ISO 10303-47. The following modifications apply to this part of ISO 10303.

The supertype clause has been removed.

5.2.3.2.35 geometric_tolerance

The base definition of the entity **geometric_tolerance** is given in ISO 10303-47. The following modifications apply to this part of ISO 10303.

The supertype clause has been pruned. The list entry **geometric_tolerance_with_defined_unit**, the keyword ONEOF, and the brackets surrounding the ONEOF list have been removed.

5.2.3.2.36 shape_dimension_representation

The base definition of the entity **shape_dimension_representation** is given in ISO 10303-47. The following modifications apply to this part of ISO 10303.

The supertype clause has been removed.

5.2.3.2.37 tolerance_value

The base definition of the entity **tolerance_value** is given in ISO 10303-47. The following modifications apply to this part of ISO 10303.

The supertype clause has been removed.

5.2.3.2.38 tolerance_zone_definition

The base definition of the entity **tolerance_zone_definition** is given in ISO 10303-47. The following modifications apply to this part of ISO 10303.

The supertype clause has been removed.

5.2.3.2.39 action_method_with_associated_documents

The base definition of the entity **action_method_with_associated_documents** is given in ISO 10303-49:—[1)]. The following modifications apply to this part of ISO 10303.

[1)] To be published.

The supertype clause has been removed.

5.2.4 Sheet metal rule definitions

5.2.4.1 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 =
    'sheet_metal'))) = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

application_context: identifies the set of all instances of the **application_context** entity.

application_protocol_definition: identifies the set of all instances of the **application_protocol_definition** entity.

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 'sheet_metal' as its **application_interpreted_model_schema_name**.

5.2.4.2 approval_requires_approval_date_time

The **approval_requires_approval_date_time** rule specifies that each instance of **approval** shall be referenced by exactly one **approval_date_time**. This rule enforces the requirement for every approval to have a date on which the approval obtained its specified status.

EXPRESS specification:

```
*)
RULE approval_requires_approval_date_time FOR
  (approval, approval_date_time);
WHERE
  WR1: SIZEOF (QUERY (a <* approval |
       NOT (SIZEOF (QUERY (adt <* approval_date_time |
       a :=: adt.dated_approval )) = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

approval_date_time: identifies the set of all instances of the **approval_date_time** entity.

Formal propositions:

WR1: For each instance of **approval**, there shall be exactly one instance of **approval_date_time** that contains the instance of **approval** as its **dated_approval** attribute.

5.2.4.3 approval_requires_approval_person_organization

The **approval_requires_approval_person_organization** rule specifies that each instance of **approval** shall be referenced by at least one **approval_person_organization**. This rule enforces the requirement for an approval to be authorized by one or more people within an organization.

EXPRESS specification:

```
*)
RULE approval_requires_approval_person_organization FOR
  (approval, approval_person_organization);
WHERE
  WR1: SIZEOF (QUERY (a <* approval |
       NOT (SIZEOF (QUERY (apo <* approval_person_organization |
       a :=: apo.authorized_approval )) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

approval: identifies the set of all instances of the **approval** entity.

approval_person_organization: identifies the set of all instances of the **approval_person_organization** entity.

Formal propositions:

WR1: For each instance of **approval**, there shall be one or more instances of **approval_person_organization** which contain the instance of **approval** as its **authorized_approval** attribute.

5.2.4.4 cardinality_action_method_to_process_plan

The **cardinality_action_method_to_process_plan** rule specifies that **action_method**s with name attributes equal to "part process plan template" shall be referenced by at least one **process_plan**. This rule enforces the ARM requirement for part_process_plan_templates to be used by no fewer than one **part_process_plan**.

EXPRESS specification:

```
*)
RULE cardinality_action_method_to_process_plan FOR
  (action_method, process_plan);
WHERE
  WR1: SIZEOF (QUERY (am <* action_method |
   (am.name = 'part process plan template') AND
   (NOT (SIZEOF (QUERY (pp <* process_plan |
       am :=: pp.chosen_method )) = 1 )))) = 0;
END_RULE;
(*
```

Argument definitions:

action_method: identifies the set of all instances of the **action_method** entity.

process_plan: identifies the set of all instances of the **process_plan** entity.

Formal propositions:

WR1: For each instance of **action_method**, there shall be at least one instance of **process_plan** that references it.

5.2.4.5 cardinality_product_definition_process_to_action_resource_requirement

The **cardinality_product_definition_process_to_action_resource_requirement** rule specifies that a **product_definition_process** shall be the operation of no more than one **action_resource_requirement**. This rule enforces the requirement in the ARM that a **process_operation** have no more than one **press_definition** defined for it.

EXPRESS specification:

```
*)
```

```
RULE cardinality_product_definition_process_to_action_resource_requirement
    FOR (product_definition_process, action_resource_requirement);
WHERE
    WR1: SIZEOF (QUERY (pdp <* product_definition_process |
         NOT (SIZEOF (QUERY (arr <* action_resource_requirement |
         pdp :=: arr.operations)) <= 1))) = 0;
END_RULE;
(*
```

Argument definitions:

action_resource_requirement: identifies the set of all instances of the **action_resource_requirement** entity.

product_definition_process: identifies the set of all instances of the **product_definition_process** entity.

Formal propositions:

WR1: For each instance of **product_definition_process**, there shall be no more than one instance of **action_resource_requirement** pointing to it via its operation attribute.

5.2.4.6 cardinality_sm_action_assignment_to_process_plan

The **cardinality_sm_action_assignment_to_process_plan** rule specifies that a **sheet_metal_action_-assignment** shall correspond to no more than one **process_plan** through its items attribute. This rule enforces the requirement in the ARM that a **work_item** be a supertype of a **part_process_plan**, and therefore associated with only zero or one of them at a time.

EXPRESS specification:

```
*)
RULE cardinality_sm_action_assignment_to_process_plan
    FOR (sheet_metal_action_assignment, process_plan);
WHERE
    WR1: SIZEOF (QUERY (pp <* process_plan |
         NOT (SIZEOF (QUERY (smaa <* sheet_metal_action_assignment |
         pp IN smaa.items)) <=1))) = 0;
END_RULE;
(*
```

Argument definitions:

process_plan: identifies the set of all instances of the **process_plan** entity.

sheet_metal_action_assignment: identifies the set of all instances of the **sheet_metal_action_assignment** entity.

Formal propositions:

WR1: For each instance of **sheet_metal_action_assignment**, there shall be no more than one instance of **process_plan** in its set of items.

5.2.4.7 cardinality_sm_action_assignment_to_product_definition

The **cardinality_sm_action_assignment_to_product_definition** rule specifies that a **sheet_metal_action_-assignment** shall correspond to no more than one **product_definition** through its items attribute. This rule enforces the requirement in the ARM that a **work_item** be a supertype of an **item_definition**, and therefore associated with only zero or one of them at a time.

EXPRESS specification:

```
*)
RULE cardinality_sm_action_assignment_to_product_definition
    FOR (sheet_metal_action_assignment, product_definition);
WHERE
    WR1: SIZEOF (QUERY (pd <* product_definition |
         NOT (SIZEOF (QUERY (smaa <* sheet_metal_action_assignment |
         pd IN smaa.items)) <=1))) = 0;
END_RULE;
(*
```

Argument definitions:

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

sheet_metal_action_assignment: identifies the set of all instances of the **sheet_metal_action_assignment** entity.

Formal propositions:

WR1: For each instance of **sheet_metal_action_assignment**, there shall be no more than one instance of **product_definition** in its set of items.

5.2.4.8 cardinality_sm_action_assignment_to_product_definition_formation

The **cardinality_sm_action_assignment_to_product_definition_formation** rule specifies that a **sheet-_metal_action_assignment** shall correspond to no more than one **product_definition_formation** through its items attribute. This rule enforces the requirement in the ARM that a **work_item** be a supertype of an **item_version**, and therefore associated with only zero or one of them at a time.

EXPRESS specification:

```
*)
RULE cardinality_sm_action_assignment_to_product_definition_formation
    FOR (sheet_metal_action_assignment, product_definition_formation);
WHERE
    WR1: SIZEOF (QUERY (pdf <* product_definition_formation |
         NOT (SIZEOF (QUERY (smaa <* sheet_metal_action_assignment |
         pdf IN smaa.items)) <=1))) = 0;
END_RULE;
(*
```

Argument definitions:

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

sheet_metal_action_assignment: identifies the set of all instances of the **sheet_metal_action_assignment** entity.

Formal propositions:

WR1: For each instance of **sheet_metal_action_assignment**, there shall be no more than one instance of **product_definition_formation** in its set of items.

5.2.4.9 cardinality_sm_organization_assignment_to_work_order

The **cardinality_sm_organization_assignment_to_work_order** rule specifies that a **sheet_metal-_organization_assignment** shall correspond to no fewer than one **work_order** through its items attribute. This rule enforces the requirement in the ARM that a **work_order_responsibility** is responsible for one or more **work_order**s.

EXPRESS specification:

```
*)
RULE cardinality_sm_organization_assignment_to_work_order
    FOR (sheet_metal_organization_assignment, work_order);
WHERE
    WR1: SIZEOF (QUERY (wo <* work_order |
    NOT (SIZEOF (QUERY (smoa <* sheet_metal_organization_assignment |
    ('effector' = smoa.role) AND (wo IN smoa.items))) >=1))) = 0;
END_RULE;
(*
```

Argument definitions:

work_order: identifies the set of all instances of the **work_order** entity.

sheet_metal_organization_assignment: identifies the set of all instances of the **sheet_metal_organization-_assignment** entity.
Formal propositions:

WR1: For each instance of **sheet_metal_organization_assignment**, there shall be at least one instance of **work_order** in its set of items.

5.2.4.10 cardinality_start_order_to_sm_action_assignment

The **cardinality_start_order_to_sm_action_assignment** rule specifies that a **start_order** shall correspond to no fewer than one **sheet_metal_action_assignment** through its assigned_action attribute. This rule enforces the requirement in the ARM that a **work_order** has one or more **work_item**s.

EXPRESS specification:

```
*)
RULE cardinality_start_order_to_sm_action_assignment
    FOR (sheet_metal_action_assignment, start_order);
WHERE
    WR1: SIZEOF (QUERY (so <* start_order |
         NOT (SIZEOF (QUERY (smaa <* sheet_metal_action_assignment |
         so :=:smaa.assigned_action)) >= 1))) = 0;
END_RULE;
(*
```

Argument definitions:

sheet_metal_action_assignment: identifies the set of all instances of the **sheet_metal_action_assignment** entity.

start_order: identifies the set of all instances of **start_order** entity.

Formal propositions:

WR1: For each instance of **start_order**, there shall be at least one instance of **sheet_metal_action_assignment** corresponding to it throught the **sheet_metal_action_assignment** assigned_action attribute.

5.2.4.11 cardinality_work_order_to_sm_contract_assig_or_sm_org_assig

The **cardinality_work_order_to_sm_contract_assig_or_sm_org_assig** rule specifies that each instance of **work_order** shall be a **contract_assigned_item** or an **organization_assigned_item**. This rule enforces the requirement for every **work_order** to be external, which is equivalent to it being the subject of a **sheet_metal_-contract_assignment**, or internal, which is equivalent to it being the subject of a **sheet_metal_organization_-assignment**.

EXPRESS specification:

```
*)
RULE cardinality_work_order_to_sm_contract_assig_or_sm_org_assig FOR
  (work_order, sheet_metal_contract_assignment,
   sheet_metal_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (wo <* work_order |
       NOT (SIZEOF (QUERY (smca <* sheet_metal_contract_assignment |
       wo IN smca.items)) = 1)
   XOR
   (SIZEOF (QUERY (smoa <* sheet_metal_organization_assignment |
   wo IN smoa.items)) = 1) )) = 0;
END_RULE;
(*
```

Argument definitions:

work_order: identifies the set of all instances of the **work_order** entity.

sheet_metal_contract_assignment: identifies the set of all instances of the **sheet_metal_contract_assignment** entity.

sheet_metal_organization_assignment: identifies the set of all instances of the **sheet_metal_organization_-assignment** entity.

Formal propositions:

WR1: Every instance of **work_order** shall be used exclusively by exactly one of either **sheet_metal_-contract_assignment** or **sheet_metal_organization_assignment**.

5.2.4.12 dependent_instantiation_for_action

This rule enforces the requirement that the entity **action**, that is not directly instantiable, but that is not an intermediate supertype, shall not be allowed to be directly instantiated.

NOTE - The correlative requirement that intermediate supertypes not be directly instantiated is enforced by the non-

inclusion of these entities in the EXPRESS "USE FROM" declarations that are found at the beginning of 5.2.

The **dependent_instantiation_for_action** rule specifies that each instance of **action** shall be referenced by at least one instance of an entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_for_action FOR (action);
WHERE
  WR1: SIZEOF (QUERY (a <* action |
       NOT (SIZEOF (USEDIN (a, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of **action** there shall be at least one instance of an entity that references **action**.

5.2.4.13 dependent_instantiation_for_action_request_solution

This rule enforces the requirement that the entity **action_request_solution**, that is not directly instantiable, but that is not an intermediate supertype, shall not be allowed to be directly instantiated.

NOTE - The correlative requirement that intermediate supertypes not be directly instantiated is enforced by the non-inclusion of these entities in the EXPRESS "USE FROM" declarations that are found at the beginning of 5.2.

The **dependent_instantiation_for_action_request_solution** rule specifies that each instance of **action_request_solution** shall be referenced by at least one instance of an entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_for_action_request_solution
  FOR (action_request_solution);
WHERE
  WR1: SIZEOF (QUERY (ars <* action_request_solution |
       NOT (SIZEOF (USEDIN (ars, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of **action_request_solution** there shall be at least one instance of an entity that references **action_request_solution**.

5.2.4.14 dependent_instantiation_for_application_context

This rule enforces the requirement that the entity **application_context**, that is not directly instantiable, but that is not an intermediate supertype, shall not be allowed to be directly instantiated.

NOTE - The correlative requirement that intermediate supertypes not be directly instantiated is enforced by the non-inclusion of these entities in the EXPRESS "USE FROM" declarations that are found at the beginning of 5.2.

The **dependent_instantiation_for_application_context** rule specifies that each instance of **application_context** shall be referenced by at least one instance of an entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_for_application_context
   FOR (application_context);
WHERE
  WR1: SIZEOF (QUERY (ac <* application_context |
       NOT (SIZEOF (USEDIN (ac, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of **application_context** there shall be at least one instance of an entity that references **application_context**.

5.2.4.15 dependent_instantiation_for_contract

This rule enforces the requirement that the entity **contract**, that is not directly instantiable, but that is not an intermediate supertype, shall not be allowed to be directly instantiated.

NOTE - The correlative requirement that intermediate supertypes not be directly instantiated is enforced by the non-inclusion of these entities in the EXPRESS "USE FROM" declarations that are found at the beginning of 5.2.

The **dependent_instantiation_for_contract** rule specifies that each instance of **contract** shall be referenced by at least one instance of an entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_for_contract
   FOR (contract);
WHERE
  WR1: SIZEOF (QUERY (c <* contract |
       NOT (SIZEOF (USEDIN (c, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of **contract** there shall be at least one instance of an entity that references **contract**.

5.2.4.16 dependent_instantiation_for_date_role

This rule enforces the requirement that the entity **date_role**, that is not directly instantiable, but that is not an intermediate supertype, shall not be allowed to be directly instantiated.

NOTE - The correlative requirement that intermediate supertypes not be directly instantiated is enforced by the non-inclusion of these entities in the EXPRESS "USE FROM" declarations that are found at the beginning of 5.2.

The **dependent_instantiation_for_date_role** rule specifies that each instance of **date_role** shall be referenced by at least one instance of an entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_for_date_role
   FOR (date_role);
WHERE
  WR1: SIZEOF (QUERY (dr <* date_role |
       NOT (SIZEOF (USEDIN (dr, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of **date_role** there shall be at least one instance of an entity that references **date_role**.

5.2.4.17 dependent_instantiation_for_date_time_role

This rule enforces the requirement that the entity **date_time_role**, that is not directly instantiable, but that is not an intermediate supertype, shall not be allowed to be directly instantiated.

NOTE - The correlative requirement that intermediate supertypes not be directly instantiated is enforced by the non-inclusion of these entities in the EXPRESS "USE FROM" declarations that are found at the beginning of 5.2.

The **dependent_instantiation_for_date_time_role** rule specifies that each instance of **date_time_role** shall be referenced by at least one instance of an entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_for_date_time_role
   FOR (date_time_role);
WHERE
  WR1: SIZEOF (QUERY (dtr <* date_time_role |
       NOT (SIZEOF (USEDIN (dtr, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of **date_time_role** there shall be at least one instance of an entity that references **date_time_role**.

5.2.4.18 dependent_instantiation_for_derived_unit

This rule enforces the requirement that the entity **derived_unit**, that is not directly instantiable, but that is not an intermediate supertype, shall not be allowed to be directly instantiated.

NOTE - The correlative requirement that intermediate supertypes not be directly instantiated is enforced by the non-inclusion of these entities in the EXPRESS "USE FROM" declarations that are found at the beginning of 5.2.

The **dependent_instantiation_for_derived_unit** rule specifies that each instance of **derived_unit** shall be referenced by at least one instance of an entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_for_derived_unit
   FOR (derived_unit);
WHERE
  WR1: SIZEOF (QUERY (du <* derived_unit |
       NOT (SIZEOF (USEDIN (du, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of **derived_unit** there shall be at least one instance of an entity that references **derived_unit**.

5.2.4.19 dependent_instantiation_for_document_reference

This rule enforces the requirement that the entity **document_reference**, that is not directly instantiable, but that is not an intermediate supertype, shall not be allowed to be directly instantiated.

NOTE - The correlative requirement that intermediate supertypes not be directly instantiated is enforced by the non-inclusion of these entities in the EXPRESS "USE FROM" declarations that are found at the beginning of 5.2.

The **dependent_instantiation_for_document_reference** rule specifies that each instance of **document_reference** shall be referenced by at least one instance of an entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_for_document_reference
   FOR (document_reference);
WHERE
  WR1: SIZEOF (QUERY (dr <* document_reference |
       NOT (SIZEOF (USEDIN (dr, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of **document_reference** there shall be at least one instance of an entity that references **document_reference**.

5.2.4.20 dependent_instantiation_for_functionally_defined_transformation

This rule enforces the requirement that the entity **functionally_defined_transformation**, that is not directly instantiable, but that is not an intermediate supertype, shall not be allowed to be directly instantiated.

NOTE - The correlative requirement that intermediate supertypes not be directly instantiated is enforced by the non-inclusion of these entities in the EXPRESS "USE FROM" declarations that are found at the beginning of 5.2.

The **dependent_instantiation_for_functionally_defined_transformation** rule specifies that each instance of **functionally_defined_transformation** shall be referenced by at least one instance of an entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_for_functionally_defined_transformation
  FOR (functionally_defined_transformation);
WHERE
  WR1: SIZEOF (QUERY (fdt <* functionally_defined_transformation |
       NOT (SIZEOF (USEDIN (fdt, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

functionally_defined_transformation: identifies the set of all instances of **functionally_defined_-transformation**.

Formal propositions:

WR1: For each instance of **functionally_defined_transformation** there shall be at least one instance of an entity that references **functionally_defined_transformation**.

5.2.4.21 dependent_instantiation_for_item_defined_transformation

This rule enforces the requirement that the entity **x**, that is not directly instantiable, but that is not an intermediate supertype, shall not be allowed to be directly instantiated.

NOTE - The correlative requirement that intermediate supertypes not be directly instantiated is enforced by the non-inclusion of these entities in the EXPRESS "USE FROM" declarations that are found at the beginning of 5.2.

The **dependent_instantiation_for_item_defined_transformation** rule specifies that each instance of **item_defined_transformation** shall be referenced by at least one instance of an entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_for_item_defined_transformation
   FOR (item_defined_transformation);
WHERE
  WR1: SIZEOF (QUERY (idt <* item_defined_transformation |
       NOT (SIZEOF (USEDIN (idt, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of **item_defined_transformation** there shall be at least one instance of an entity that references **item_defined_transformation**.

5.2.4.22 dependent_instantiation_for_organization_role

This rule enforces the requirement that the entity **organization_role**, that is not directly instantiable, but that is not an intermediate supertype, shall not be allowed to be directly instantiated.

NOTE - The correlative requirement that intermediate supertypes not be directly instantiated is enforced by the non-inclusion of these entities in the EXPRESS "USE FROM" declarations that are found at the beginning of 5.2.

The **dependent_instantiation_for_organization_role** rule specifies that each instance of **organization_role** shall be referenced by at least one instance of an entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_for_organization_role
   FOR (organization_role);
WHERE
  WR1: SIZEOF (QUERY (o_r <* organization_role |
       NOT (SIZEOF (USEDIN (o_r, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of **organization_role** there shall be at least one instance of an entity that references **organization_role**.

5.2.4.23 dependent_instantiation_for_person_and_organization_role

This rule enforces the requirement that the entity **person_and_organization_role**, that is not directly instantiable, but that is not an intermediate supertype, shall not be allowed to be directly instantiated.

NOTE - The correlative requirement that intermediate supertypes not be directly instantiated is enforced by the non-inclusion of these entities in the EXPRESS "USE FROM" declarations that are found at the beginning of 5.2.

The **dependent_instantiation_for_person_and_organization_role** rule specifies that each instance of **person_and_organization_role** shall be referenced by at least one instance of an entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_for_person_and_organization_role
  FOR (person_and_organization_role);
WHERE
  WR1: SIZEOF (QUERY (paor <* person_and_organization_role |
       NOT (SIZEOF (USEDIN (paor, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of **person_and_organization_role** there shall be at least one instance of an entity that references **person_and_organization_role**.

5.2.4.24 dependent_instantiation_for_person_role

This rule enforces the requirement that the entity **person_role**, that is not directly instantiable, but that is not an intermediate supertype, shall not be allowed to be directly instantiated.

NOTE - The correlative requirement that intermediate supertypes not be directly instantiated is enforced by the non-inclusion of these entities in the EXPRESS "USE FROM" declarations that are found at the beginning of 5.2.

The **dependent_instantiation_for_person_role** rule specifies that each instance of **person_role** shall be referenced by at least one instance of an entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_for_person_role
  FOR (person_role);
WHERE
  WR1: SIZEOF (QUERY (pr <* person_role |
       NOT (SIZEOF (USEDIN (pr, '')) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of **person_role** there shall be at least one instance of an entity that references **person_role**.

5.2.4.25 dependent_instantiation_for_tolerance_value

This rule enforces the requirement that the entity **tolerance_value**, that is not directly instantiable, but that is not an intermediate supertype, shall not be allowed to be directly instantiated.

NOTE - The correlative requirement that intermediate supertypes not be directly instantiated is enforced by the non-inclusion of these entities in the EXPRESS "USE FROM" declarations that are found at the beginning of 5.2.

The **dependent_instantiation_for_tolerance_value** rule specifies that each instance of **tolerance_value** shall be referenced by at least one instance of an entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_for_tolerance_value
  FOR (tolerance_value);
WHERE
  WR1: SIZEOF (QUERY (tr <* tolerance_value |
       NOT (SIZEOF (USEDIN (tr, '')) >= 1 ))) = 0;
END_RULE;
```

```
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of **tolerance_value** there shall be at least one instance of an entity that references **tolerance_value**.

5.2.4.26 executed_action_requires_approval

The **executed_action_requires_approval** rule specifies that each instance of **executed_action** shall be related to one or more instances of the **approval** entity by means of the **sheet_metal_approval_assignment** entity. This rule specifies the need for every **executed_action** to have an an associated approval(s).

EXPRESS specification:

```
*)
RULE executed_action_requires_approval FOR
  (executed_action, sheet_metal_approval_assignment);
WHERE
  WR1: SIZEOF (QUERY (ea <* executed_action |
       NOT (SIZEOF (QUERY (smaa <*
       sheet_metal_approval_assignment |
       ea IN smaa.items)) >=1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

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

Formal propositions:

WR1: For each instance of **executed_action**, one or more related instances of the **sheet_metal_approval_-assignment** entity shall exist.

5.2.4.27 executed_action_requires_date_or_date_and_time

The **executed_action_requires_date_or_date_and_time** rule specifies that each instance of **executed_action**

shall be related to exactly one instance of the **date** entity by means of the **sheet_metal_date_assignment** entity, or exactly one instance of the **date_and_time** entity by means of the **sheet_metal_date_and_time_assignment** entity. This rule specifies the need for every **executed_action** to have exactly one associated date, and optionally exactly one associated time.

EXPRESS specification:

```
*)
RULE executed_action_requires_date_or_date_and_time FOR
  (executed_action, sheet_metal_date_assignment,
   sheet_metal_date_and_time_assignment);
WHERE
  WR1: SIZEOF (QUERY (ea <* executed_action |
       NOT (SIZEOF (QUERY (smda <* sheet_metal_date_assignment |
   ea IN smda.items)) +
       (SIZEOF (QUERY (smdata <* sheet_metal_date_and_time_assignment |
   ea IN smdata.items))) =1 ) )) = 0;
END_RULE;
(*
```

Argument definitions:

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

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

sheet_metal_date_and_time_assignment: identifies the set of all instances of **sheet_metal_date_and_time_-assignment** entities

Formal propositions:

WR1: For each instance of **executed_action**, exactly one instance of the related **sheet_metal_date_assignment** entity or exactly one instance of the related **sheet_metal_date_and_time_assignment** entity shall exist .

5.2.4.28 executed_action_requires_organization

The **executed_action_requires_organization** rule specifies that each instance of **executed_action** shall be related to one or more instances of the **organization** entity by means of the **sheet_metal_organization_-assignment** entity. This rule specifies the need for every **executed_action** to have an associated organization(s).

EXPRESS specification:

```
*)
RULE executed_action_requires_organization FOR
  (executed_action, sheet_metal_organization_assignment);
WHERE
  WR1: SIZEOF (QUERY (ea <* executed_action |
       NOT (SIZEOF (QUERY (smoa <*
       sheet_metal_organization_assignment |
       ea IN smoa.items)) >=1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

sheet_metal_organization_assignment: identifies the set of all instances of **sheet_metal_organization_-assignment** entities.

Formal propositions:

WR1: For each instance of **executed_action**, one or more related instances of the **sheet_metal_organization_-assignment** entity shall reference it.

5.2.4.29 global_uncertainty_assigned_context_constraint

The **global_uncertainty_assigned_context_constraint** rule enforces the requirement that the entity **global_-uncertainty_assigned_context** be associated with information concerning the maximum linear and maximum angular distance between two elements in a product_definition_shape or shape_aspect under which those elements may still be considered to be coincident.

EXPRESS specification:

```
*)
RULE global_uncertainty_assigned_context_constraint
 FOR (global_uncertainty_assigned_context);
WHERE
  WR1: SIZEOF (QUERY (guac <* global_uncertainty_assigned_context |
       NOT (SIZEOF (guac.uncertainty)=2)))=0;
  WR2: SIZEOF (QUERY (guac <* global_uncertainty_assigned_context |
       NOT ((SIZEOF (QUERY (x <* guac.uncertainty |
              'SHEET_METAL.LENGTH_UNIT' IN TYPEOF (x.unit_component)))=1) AND
             (SIZEOF (QUERY (x <* guac.uncertainty |
              'SHEET_METAL.PLANE_ANGLE_UNIT' IN TYPEOF (x.unit_component)))=1)
       )))=0;
END_RULE;
(*
```

Argument definitions:

global_uncertainty_assigned_context: the set of all instances of the **global_uncertainty_assigned_context** entity.

Formal propositions:

WR1: Each **global_uncertainty_assigned_context** shall have exactly two elements in its set of **units**.

WR2: Each **global_uncertainty_assigned_context** shall have exactly one **length_unit** and exactly one **plane_angle_unit** in its set of unit_components via its set of uncertainties.

5.2.4.30 process_plan_requires_product_definition

The **process_plan_requires_product_definition** rule specifies that each instance of **process_plan** shall make reference to at least one instance of the **product_definition** entity through the **sheet_metal_action_assignment** entity.

EXPRESS specification:

```
*)
RULE process_plan_requires_product_definition FOR
  (process_plan, sheet_metal_action_assignment);
WHERE
  WR1: SIZEOF (QUERY (pp <* process_plan |
       NOT (SIZEOF (QUERY ( smaa <* sheet_metal_action_assignment |
       (smaa\action_assignment.assigned_action :=: pp\action) AND
       (SIZEOF (QUERY (i <* smaa.items | 'SHEET_METAL.' +
       'PRODUCT_DEFINITION' IN TYPEOF(i))) >= 1))) = 1))) = 0;
END_RULE;
(*
```

Argument definitions:

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

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

Formal propositions:

WR1: For each instance of **process_plan**, there shall be at least one instance of a **product_definition** entity that relates to it by means of the **sheet_metal_action_assignment** entity.

5.2.4.31 product_definition_formation_requires_date_or_date_and_time

The **product_definition_formation_requires_date_or_date_and_time** rule specifies that each instance of **product_definition_formation** shall be related to at exactly one instance of the **date** entity by means of the **sheet_metal_date_assignment** entity, or exactly one instance of the **date_and_time** entity by means of the **sheet_metal_date_and_time_assignment** entity. This rule specifies the need for every **product_definition_formation** to have exactly one associated date, and optionally exactly one associated time.

EXPRESS specification:

```
*)
RULE product_definition_formation_requires_date_or_date_and_time FOR
  (product_definition_formation, sheet_metal_date_assignment,
   sheet_metal_date_and_time_assignment);
WHERE
  WR1: SIZEOF (QUERY (pdf <* product_definition_formation |
       NOT (SIZEOF (QUERY (smda <* sheet_metal_date_assignment |
    pdf IN smda.items)) +
       (SIZEOF (QUERY (smdata <* sheet_metal_date_and_time_assignment |
    pdf IN smdata.items))) =1 ) )) = 0;
END_RULE;
(*
```

Argument definitions:

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

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

sheet_metal_date_and_time_assignment: identifies the set of all instances of **sheet_metal_date_and_time_assignment** entities

Formal propositions:

WR1: For each instance of **product_definition_formation** exactly one related instance of the **sheet_metal_date_assignment** entity or exactly one instance of the related **sheet_metal_date_and_time_assignment** entity shall exist .

5.2.4.32 product_definition_formation_requires_product_definition

The **product_definition_formation_requires_product_definition** rule specifies that each instance of **product_definition_formation** shall be referenced by at least one instance of **product_definition**. This rule enforces the requirement for every **product_definition_formation** to have one or more definitions.

EXPRESS specification:

```
*)
RULE product_definition_formation_requires_product_definition FOR
```

```
  (product_definition_formation, product_definition);
WHERE
  WR1: SIZEOF (QUERY (pdf <* product_definition_formation |
       NOT (SIZEOF (QUERY (pd <* product_definition |
       pdf :=: pd.formation )) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

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

Formal propositions:

WR1: For each instance of **product_definition_formation**, there shall be one or more instances of **product_definition** that contains a **formation** attribute value equal to that instance of **product_definition_formation**.

5.2.4.33 product_requires_product_definition_formation

The **product_requires_product_definition_formation** rule specifies that each instance of **product** shall be referenced by at least one instance of **product_definition_formation**. This rule enforces the requirement for every product to have one or more versions.

EXPRESS specification:

```
*)
RULE product_requires_product_definition_formation FOR
  (product, product_definition_formation);
WHERE
  WR1: SIZEOF (QUERY (p <* product |
       NOT (SIZEOF (QUERY (pdf <* product_definition_formation |
       p :=: pdf.of_product )) >= 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

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

Formal propositions:

WR1: For each instance of **product**, there shall be one or more instances of **product_definition_formation**

that contains an **of_product** attribute value equal to that instance of **product**.

5.2.4.34 product_requires_product_type

The **product_requires_product_type** rule specifies that all **product**s shall be associated with exactly one instance of the **product_type** entity.

EXPRESS specification:

```
*)
RULE product_requires_product_type FOR (product, product_type);
WHERE
  WR1: SIZEOF (QUERY (p <* product |
       NOT (SIZEOF (QUERY (pt <* product_type |
       SIZEOF (p IN
       pt\product_related_product_category.products) = 1 ))
       = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

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

Formal propositions:

WR1: For each instance of **product**, there shall be exactly one instance of **product_type**.

5.2.4.35 restrict_assembly_component_usage_substitute

The **restrict_assembly_component_usage_substitute** rule specifies that the two **assembly_component_usage** instances that are related shall be of the type **quantified_assembly_component_usage** and the **product_type**s in the **base** and **substitute assembly_component_usage**s shall have the same name.

EXPRESS specification:

```
*)
RULE restrict_assembly_component_usage_substitute FOR
  (assembly_component_usage_substitute, product_type);
WHERE
  WR1: SIZEOF (QUERY (acus <* assembly_component_usage_substitute |
   NOT (('SHEET_METAL.QUANTIFIED_ASSEMBLY_COMPONENT_USAGE'
   IN TYPEOF (acus.base)) AND
   ('SHEET_METAL.QUANTIFIED_ASSEMBLY_COMPONENT_USAGE'
   IN TYPEOF (acus.substitute))))) = 0;
```

```
  WR2: SIZEOF (QUERY (acus <* assembly_component_usage_substitute |
  product_type_of_product
  (acus.base.relating_product_definition.formation.of_product,
      product_type)
  .name = product_type_of_product
  (acus.substitute.relating_product_definition.formation.of_product,
  product_type)
  .name )));
END_RULE;
(*
```

Argument definitions:

assembly_component_usage_substitute: identifies the set of all instances of **assembly_component_usage_-substitute** entities.

product_type: identifies the set of all instances of the **product_type** entity.

Formal propositions:

WR1: For each instance of the **assembly_component_usage_substitute** entity, the **base** and **substitute** **assembly_component_usage**s shall be of the type **quantified_assembly_component_usage**.

WR2: For each instance of the **assembly_component_usage_substitute** entity, the **base** and **substitute** **assembly_component_usage**s shall have the same **product_type.name**.

5.2.4.36 restrict_date_role

The **restrict_date_role** rule specifies the permitted roles for **date** entities. This rule enforces the requirement for the roles of **date** entities to be "creation date", "revision date", "review date", "production year date", "completion date", "order date", "preliminary review date", "start date" or "request date".

EXPRESS specification:

```
*)
RULE restrict_date_role FOR
  (date_role);
WHERE
  WR1: SIZEOF (QUERY (dr <* date_role |
       NOT (dr.name IN ['creation date', 'revision date', 'review date',
       'production year date', 'completion date', 'order date',
       'preliminary review date', 'start date', 'request date']))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of the **date_role**, the **name** attribute shall have a value of ""creation date", "revision date", "review date", "completion date", "order date", "preliminary review date", “production year date”, "start date", or "request date".

Attribute value definitions:

creation date: identifies the date when either a definition of a design or process plan comes into existence.

revision date: identifies the date when the definition of a design is revised.

review date: identifies the date that a process plan has gone through a review.

completion date: identifies the date when a piece of work is scheduled to be completed.

order date: identifies the date when work is ordered.

preliminary review date: identifies the date when a piece of work has undergone or is scheduled to undergo a review that is designed to monitor progress on work that is not yet complete.

production year date: identifies the year in which a product goes into production.

start date: identifies the date when work is scheduled to begin.

request date: identifies the date that work was requested.

5.2.4.37 restrict_date_time_role

The **restrict_date_time_role** rule specifies the permitted roles for **date_and_time** entities. This rule enforces the requirement for the roles of **date_and_time** entities to be "creation date", "revision date", "review date", "production year date", "completion date", "order date", "preliminary review date", "start date" or "request date".

EXPRESS specification:

```
*)
RULE restrict_date_time_role FOR
  (date_time_role);
WHERE
  WR1: SIZEOF (QUERY (dtr <* date_time_role |
       NOT (dtr.name IN ['creation date', 'revision date',
```

```
   'review date', 'completion date', 'order date',
   'preliminary review date', 'start date', 'request date'])) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of the **date_time_role**, the **name** attribute shall have a value of "creation date", "revision date", "review date", "completion date", "order date", "preliminary review date", "start date", or "request date".

Attribute value definitions:

creation date: identifies the date when either a definition of a design or process plan comes into existence.

revision date: identifies the date when the definition of a design is revised.

review date: identifies the date that a process plan has gone through a review.

completion date: identifies the date when a piece of work is scheduled to be completed.

order date: identifies the date when work is ordered.

preliminary review date: identifies the date when a piece of work has undergone or is scheduled to undergo a review that is designed to monitor progress on work that is not yet complete.

start date: identifies the date when work is scheduled to begin.

request date: identifies the date that work was requested.

5.2.4.38 restrict_organization_role

The **restrict_organization_role** rule specifies the permitted roles for **organization** entities. This rule enforces the requirement for the roles of **organization** entities to be "design supplier", "effector", "implementor", "order source", "plant", and "supplier".

EXPRESS specification:

```
*)
RULE restrict_organization_role FOR
  (organization_role);
WHERE
  WR1: SIZEOF (QUERY (o_r <* organization_role |
       NOT (o_r.name IN ['design supplier', 'effector', 'implementor',
    'order source', 'plant', 'supplier']))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of the **organization_role**, the **name** attribute shall have a value of "design supplier", "implementor", "supplier", "order source", or "effector".

Attribute value definitions:

design supplier: identifies the organization responsible for supplying a design.

implementor: identifies the organization responsible for implementing a change order.

supplier: identifies the organization responsible for supplying work based on a contract or internal work order.

order source: identifies the internal organization that created a work order.

effector: identifies the organization responsible for making sure that a work order is carried out.

5.2.4.39 restrict_person_role

The **restrict_person_role** rule specifies the permitted roles for **person** entities. This rule enforces the requirement for the roles of **person** entities to be "designer", "implementor", "purchaser" or "work requestor".

EXPRESS specification:

```
*)
RULE restrict_person_role FOR
  (person_role);
WHERE
  WR1: SIZEOF (QUERY (pr <* person_role |
```

```
      NOT (pr.name IN ['designer', 'implementor', 'purchaser',
  'work requestor']))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of the **person_role**, the **name** attribute shall have a value of "designer", "implementor", "purchaser" or "work requestor".

Attribute value definitions:

designer: identifies the person responsible for creating a design.

implementor: identifies the person responsible for implementing a change order.

purchaser: identifies the person responsible for procuring services of a supplier.

work requestor: identifies the person that makes a request for work to be done.

5.2.4.40 restrict_person_and_organization_role

The **restrict_person_and_organization_role** rule specifies the permitted roles for **person_and_organization** entities. This rule enforces the requirement for the roles of **person_and_organization** entities to be "data owner", "planner" or "implementor".

EXPRESS specification:

```
*)
RULE restrict_person_and_organization_role FOR
  (person_and_organization_role);
WHERE
  WR1: SIZEOF (QUERY (paor <* person_and_organization_role |
       NOT (paor.name IN ['data owner', 'planner',
       'implementor']))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of the **person_and_organization_role**, the **name** attribute shall have a value of "data owner", "planner" or "implementor".

Attribute value definitions:

data owner: identifies the person within an organization who owns design or process planning information.

planner: identifies the person within an organization who is responsible for creating a process plan.

implementor: identifies the person within an organization who is responsible for implementing a change order.

5.2.4.41 restrict_product_definition_context

The **restrict_product_definition_context** rule specifies the permitted values for the **life_cycle_stage** of a **product_definition_context**.

EXPRESS specification:

```
*)
RULE restrict_product_definition_context FOR
  (product_definition_context);
WHERE
  WR1: SIZEOF (QUERY (pdc <* product_definition_context |
       NOT ((pdc.life_cycle_stage = 'design') OR
       (pdc.life_cycle_stage = 'in process') OR
       (pdc.life_cycle_stage = 'as physically modelled') OR
       (pdc.life_cycle_stage = 'as design constrained')))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: For each instance of the **product_definition_context**, the **life_cycle_stage** attribute shall have a value of "design","in process","as physically modelled", or "as design constrained".

Attribute value definitions:

design: design of an item in the final form it will take before any assembly operations

in process: design of an item in any form it takes prior to the final pre-assembled form

as physically modelled: design of an item as represented by a physical object

as design constrained: design of an item after accounting for constraints imposed by designs of other items

5.2.4.42 restrict_quantified_assembly_component_usage

The **restrict_quantified_assembly_component_usage** rule specifies that the two **product_definition**s that are related by the **quantified_assembly_component_usage** entity shall be of the same **product_type.name**.

NOTE - A part can only substutite for another part, a die component can only substitute for another die component, et cetera.

EXPRESS specification:

```
*)
RULE restrict_quantified_assembly_component_usage FOR
  (quantified_assembly_component_usage, product_type);
WHERE
  WR1: SIZEOF (QUERY (qacu <* quantified_assembly_component_usage |
       product_type_of_product
 (qacu.relating_product_definition.formation.of_product, product_type).name =
    product_type_of_product
 (qacu.related_product_definition.formation.of_product, product_type).name ));
END_RULE;
(*
```

Argument definitions:

quantified_assembly_component_usage: identifies the set of all instances of **quantified_assembly_-component_usage** entities.

product_type: identifies the set of all instances of the **product_type** entity.

Formal propositions:

WR1: For each instance of the **quantified_assembly_component_usage** entity, the **product_type**s corresponding to the two related **product_definition**s shall have the same name.

5.2.4.43 same_type_for_relationships

The **same_type_for_relationships** rule specifies that two **product_definition**s taking part in an assembly component, mating or symmetrical relationship by means of the **product_definition_relationship** entity shall be of the same **product_type.name**.

EXPRESS specification:

```
*)
RULE same_type_for_relationships FOR
  (product_definition_relationship, product_type);
WHERE
  WR1: SIZEOF (QUERY (pdr <* product_definition_relationship |
       NOT ((pdr.name <> 'assembly component'
       AND pdr.name <> 'mating' AND pdr.name <> 'symmetrical')
       OR (pdr.name IN ['assembly component', 'mating', 'symmetrical'])
       AND (product_type_of_product
       (pdr.relating_product_definition.formation.of_product,
       product_type).name =
   product_type_of_product
       (pdr.related_product_definition.formation.of_product,
       product_type).name ))))) = 0;
END_RULE;
(*
```

Argument definitions:

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

product_type: identifies the set of all instances of the **product_type** entity.

Formal propositions:

WR1: For each instance of the **product_definition_relationship** entity that represents a mating or symmetrical relationship, the **product_type**s corresponding to the two related **product_definition**s shall have the same name.

5.2.4.44 subtype_mandatory_for_shape_representation

The **subtype_mandatory_for_shape_representation** rule specifies that each instance of **shape_representation** shall be an instance of one of its subtypes: **advanced_brep_shape_representation**, **faceted_brep_shape_representation**, **csg_shape_representation**, **geometrically_bounded_surface_shape_-representation**, **edge_based_wireframe_shape_representation**, **shell_based_wireframe_shape_-representation**, **geometrically_bounded_wireframe_shape_representation**, or **manifold_surface_shape_-representation**. This rule enforces the ARM requirement that shape_representations be complete categorizations of their subtypes.

EXPRESS specification:

```
*)
RULE subtype_mandatory_for_shape_representation FOR (shape_representation);
WHERE
```

```
  WR1: SIZEOF ( QUERY (sr <* shape_representation |
   NOT ( SIZEOF ( ['SHEET_METAL.ADVANCED_BREP_SHAPE_REPRESENTATION',
    'SHEET_METAL.FACETED_BREP_SHAPE_REPRESENTATION',
    'SHEET_METAL.CSG_SHAPE_REPRESENTATION',
    'SHEET_METAL.GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION',
    'SHEET_METAL.EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION',
    'SHEET_METAL.SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION',
    'SHEET_METAL.GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_REPRESENTATION',
    'SHEET_METAL.MANIFOLD_SURFACE_SHAPE_REPRESENTATION'] )
    * ( TYPEOF ('sr') ) = 1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: Each instance of **shape_representation** shall be an instance of either **advanced_brep_shape_-representation**, **faceted_brep_shape_representation**, **csg_shape_representation**, **geometrically_bounded_-surface_shape_representation**, **edge_based_wireframe_shape_representation**, **shell_based_wireframe_-shape_representation**, **geometrically_bounded_wireframe_shape_representation**, or **manifold_surface_-shape_representation**.

5.2.4.45 subtype_mandatory_work_order

The **subtype_mandatory_work_order** rule specifies that each instance of **work_order** shall be an instance of **change_order** or **start_order**. This rule enforces the requirement that work orders must be for starting a new item or changing an existing one.

EXPRESS specification:

```
*)
RULE subtype_mandatory_work_order FOR (work_order);
WHERE
  WR1: SIZEOF ( QUERY (wo <* work_order |
    (NOT ('SHEET_METAL.CHANGE_ORDER' IN TYPEOF (wo)) AND
        NOT ('SHEET_METAL.START_ORDER' IN TYPEOF (wo))) )) = 0;
END_RULE;
(*
```

Argument definitions:

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

Formal propositions:

WR1: Each instance of **work_order** shall be an instance of either **change_order** or **start_order**.

5.2.4.46 work_request_requires_date_or_date_and_time

The **work_request_requires_date_or_date_and_time** rule specifies that each instance of **versioned_action_request** shall be related to exactly one instance of the **date** entity by means of the **sheet_metal_date_assignment**, or exactly one instance of the **date_and_time** entity by means of the **sheet_metal_date_and_time_assignment**. This rule specifies the need for every **versioned_action_request** to have exactly one request date, and optionally exactly one request time, specified.

EXPRESS specification:

```
*)
RULE work_request_requires_date_or_date_and_time FOR
  (versioned_action_request, sheet_metal_date_assignment,
   sheet_metal_date_and_time_assignment);
WHERE
  WR1: SIZEOF (QUERY (varq <* versioned_action_request |
    NOT (SIZEOF (QUERY (smda <* sheet_metal_date_assignment |
    varq IN smda.items)) +
    SIZEOF (QUERY (smdata <* sheet_metal_date_and_time_assignment |
    varq IN smdata.items)) =1 ) )) = 0;
END_RULE;
(*
```

Argument definitions:

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

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

sheet_metal_date_and_time_assignment: identifies the set of all instances of **sheet_metal_date_and_time_assignment** entities

Formal propositions:

WR1: For each instance of **versioned_action_request**, exactly one related instance of the **sheet_metal_date_assignment** entity or exactly one related instance of the **sheet_metal_date_and_time_assignment** entity shall exist .

5.2.4.47 work_request_requires_person

The **work_request_requires_person** rule specifies that each instance of **versioned_action_request** shall be related to one or more instances of the **person** entity by means of the **sheet_metal_person_assignment** entity. This rule specifies the need for every **versioned_action_request** to have an individual(s) who is the work

requestor. The meaning of work requestor is found in the **person_assignment**.

EXPRESS specification:

```
*)
RULE work_request_requires_person FOR
  (versioned_action_request, sheet_metal_person_assignment);
WHERE
  WR1: SIZEOF (QUERY (varq <* versioned_action_request |
       NOT (SIZEOF (QUERY (smpa <* sheet_metal_person_assignment |
       varq IN smpa.items)) >=1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

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

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

Formal propositions:

WR1: For each instance of **versioned_action_request**, one or more related instances of the **sheet_metal_-person_assignment** entity shall exist.

5.2.5 Sheet Metal function definitions

5.2.5.1 product_type_of_product

The **product_type_of_product** function determines the **product_type** instance associated with a given instance of the **product** entity.

EXPRESS specification:

```
*)
FUNCTION product_type_of_product
         (prod : product ; types : SET [1:?] OF product_type )
         : product_type;

  REPEAT i := 1 TO HIINDEX (types);
    IF prod IN types[i]\product_related_product_category.products
      THEN RETURN (types[i]);
    END_IF;
  END_REPEAT;

END_FUNCTION;
(*
```

Argument definitions:

prod: identifies the input **product** instance for which the **product_type** will be determined.

types: identifies the set of all instances of **product_type** entities.

5.2.5.2 sheet_metal_date_correlation

The **sheet_metal_date_correlation** boolean function returns true if the **name** attribute value of the **date_role** entity is coordinated with the type of entity selected in the **items** of the **sheet_metal_date_assignment** entity.

EXAMPLE 45 - If the role of a **date** is 'creation' then all of the **items** in the **sheet_metal_date_assignment** shall be **product_definition** or **process_plan** entities.

EXPRESS specification:

```
*)
FUNCTION sheet_metal_date_correlation
  (e : sheet_metal_date_assignment ) : BOOLEAN;
  LOCAL
    dt_role : label;
  END_LOCAL;
  dt_role := e\date_assignment.role.name;
  CASE dt_role OF
    'creation'              : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                SIZEOF(['SHEET_METAL.' +
                                'PRODUCT_DEFINITION',
                                'SHEET_METAL.PROCESS_PLAN'] *
                                TYPEOF (x)) = 1))
                                THEN RETURN (FALSE);
                              END_IF;
    'revision'              : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'SHEET_METAL.' +
                                'PRODUCT_DEFINITION'
                                IN TYPEOF (x)))
                                THEN RETURN (FALSE);
                              END_IF;
    'review'                : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'SHEET_METAL.' +
                                'PROCESS_PLAN'
                                IN TYPEOF (x)))
                                THEN RETURN (FALSE);
                              END_IF;
    'production year'       : IF SIZEOF (e.items) <>
                                SIZEOF (QUERY (x <* e.items |
                                'SHEET_METAL.' +
                                'PRODUCT_CONCEPT'
                                IN TYPEOF (x)))
```

```
                                  THEN RETURN (FALSE);
                                END_IF;
    'completion'              : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'SHEET_METAL.' +
                                  'EXECUTED_ACTION'
                                  IN TYPEOF (x)))
                                  THEN RETURN (FALSE);
                                END_IF;
    'order'                   : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'SHEET_METAL.' +
                                  'EXECUTED_ACTION'
                                  IN TYPEOF (x)))
                                  THEN RETURN (FALSE);
                                END_IF;
    'preliminary review'      : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'SHEET_METAL.' +
                                  'EXECUTED_ACTION'
                                  IN TYPEOF (x)))
                                  THEN RETURN (FALSE);
                                END_IF;
    'start'                   : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'SHEET_METAL.' +
                                  'EXECUTED_ACTION'
                                  IN TYPEOF (x)))
                                  THEN RETURN (FALSE);
                                END_IF;
    'request'                 : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'SHEET_METAL.' +
                                  'VERSIONED_ACTION_REQUEST'
                                  IN TYPEOF (x)))
                                  THEN RETURN (FALSE);
                                END_IF;
    OTHERWISE  : RETURN (TRUE);
  END_CASE;
END_FUNCTION;
(*
```

Argument definitions:

e: identifies the input **sheet_metal_date_assignment** to be checked.

5.2.5.3 sheet_metal_document_correlation

The **sheet_metal_document_correlation** boolean function returns true if the **product_data_type** attribute value of the **document_type** entity is coordinated with the type of entity selected in the **items** of the **sheet_-metal_document_reference** entity.

EXAMPLE 46 - If the type of a **document** is 'die layout' then all of the **items** in the **sheet_metal_document_reference** shall be **product_definition** entities.

EXPRESS specification:

```
*)
FUNCTION sheet_metal_document_correlation
  (e : sheet_metal_document_reference ) : BOOLEAN;
  LOCAL
    doc_type : label;
  END_LOCAL;
  doc_type := e\document_reference.assigned_document.kind.product_data_type;
  CASE doc_type OF
    'die layout'             : IF SIZEOF (e.items) <>
                                 SIZEOF (QUERY (x <* e.items |
                                 'SHEET_METAL.' +
                                 'PRODUCT_DEFINITION'
                                 IN TYPEOF (x)))
                                 THEN RETURN (FALSE);
                               END_IF;
    'die structure'          : IF SIZEOF (e.items) <>
                                 SIZEOF (QUERY (x <* e.items |
                                 'SHEET_METAL.' +
                                 'PRODUCT_DEFINITION'
                                 IN TYPEOF (x)))
                                 THEN RETURN (FALSE);
                               END_IF;
    'pattern casting'        : IF SIZEOF (e.items) <>
                                 SIZEOF (QUERY (x <* e.items |
                                 'SHEET_METAL.' +
                                 'PRODUCT_DEFINITION'
                                 IN TYPEOF (x)))
                                 THEN RETURN (FALSE);
                               END_IF;
    'press'                  : IF SIZEOF (e.items) <>
                                 SIZEOF (QUERY (x <* e.items |
                                 'SHEET_METAL.' +
                                 'PRODUCT_DEFINITION'
                                 IN TYPEOF (x)))
                                 THEN RETURN (FALSE);
                               END_IF;
    'generating system information' : IF SIZEOF (e.items) <>
                                 SIZEOF (QUERY (x <* e.items |
                                 'SHEET_METAL.' +
                                 'PROCESS_PLAN'
                                 IN TYPEOF (x)))
                                 THEN RETURN (FALSE);
                               END_IF;
    'applicable standards'   : IF SIZEOF (e.items) <>
                                 SIZEOF (QUERY (x <* e.items |
                                 'SHEET_METAL.' +
                                 'EXECUTED_ACTION'
                                 IN TYPEOF (x)))
                                 THEN RETURN (FALSE);
                               END_IF;
```

```
    OTHERWISE  : RETURN (TRUE);
  END_CASE;
END_FUNCTION;
(*
```

Argument definitions:

e: identifies the input **sheet_metal_document_reference** to be checked.

5.2.5.4 sheet_metal_organization_correlation

The **sheet_metal_organization_correlation** boolean function returns true if the **name** attribute value of the **organization_role** entity is coordinated with the type of entity selected in the **items** of the **sheet_metal_-organization_assignment** entity.

EXAMPLE 47 - If the role of an **organization** is 'design supplier' then all of the **items** in the **sheet_metal_-organization_assignment** shall be **product_definition** entities.

EXPRESS specification:

```
*)
FUNCTION sheet_metal_organization_correlation
  (e : sheet_metal_organization_assignment ) : BOOLEAN;
  LOCAL
    o_role : label;
  END_LOCAL;
  o_role := e\organization_assignment.role.name;
  CASE o_role OF
    'design supplier'        : IF SIZEOF (e.items) <>
                                 SIZEOF (QUERY (x <* e.items |
                                 'SHEET_METAL.' +
                                 'PRODUCT_DEFINITION'
                                 IN TYPEOF (x)))
                                 THEN RETURN (FALSE);
                               END_IF;
    'implementor'            : IF SIZEOF (e.items) <>
                                 SIZEOF (QUERY (x <* e.items |
                                 'SHEET_METAL.' +
                                 'EXECUTED_ACTION'
                                 IN TYPEOF (x)))
                                 THEN RETURN (FALSE);
                               END_IF;
    'supplier'               : IF SIZEOF (e.items) <>
                                 SIZEOF (QUERY (x <* e.items |
                                 SIZEOF (['SHEET_METAL.' +
                                 'CONTRACT', 'SHEET_METAL.' +
                                 'EXECUTED_ACTION'] *
                                 TYPEOF (x)) = 1 ))
                                 THEN RETURN (FALSE);
                               END_IF;
    'order source'           : IF SIZEOF (e.items) <>
                                 SIZEOF (QUERY (x <* e.items |
```

```
                                  'SHEET_METAL.' +
                                  'EXECUTED_ACTION'
                                  IN TYPEOF (x)))
                                  THEN RETURN (FALSE);
                                END_IF;
    'effector'                : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'SHEET_METAL.' +
                                  'EXECUTED_ACTION'
                                  IN TYPEOF (x)))
                                  THEN RETURN (FALSE);
                                END_IF;
    OTHERWISE  : RETURN (TRUE);
  END_CASE;
END_FUNCTION;
(*
```

Argument definitions:

e: identifies the input **sheet_metal_organization_assignment** to be checked.

5.2.5.5 sheet_metal_person_correlation

The **sheet_metal_person_correlation** boolean function returns true if the **name** attribute value of the **person_-role** entity is coordinated with the type of entity selected in the **items** of the **sheet_metal_person_assignment** entity.

EXAMPLE 48 - If the role of a **person** is 'designer' then all of the **items** in the **sheet_metal_person_assignment** shall be **product_definition** entities.

EXPRESS specification:

```
*)
FUNCTION sheet_metal_person_correlation
  (e : sheet_metal_person_assignment ) : BOOLEAN;
  LOCAL
    p_role : label;
  END_LOCAL;
  p_role := e\person_assignment.role.name;
  CASE p_role OF
    'designer'                : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'SHEET_METAL.' +
                                  'PRODUCT_DEFINITION'
                                  IN TYPEOF (x)))
                                  THEN RETURN (FALSE);
                                END_IF;
    'implementor'             : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'SHEET_METAL.' +
                                  'EXECUTED_ACTION'
```

```
                                  IN TYPEOF (x)))
                                  THEN RETURN (FALSE);
                              END_IF;
    'purchaser'               : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'SHEET_METAL.' +
                                  'CONTRACT'
                                  IN TYPEOF (x)))
                                  THEN RETURN (FALSE);
                              END_IF;
    'work requestor'          : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'SHEET_METAL.' +
                                  'VERSIONED_ACTION_REQUEST'
                                  IN TYPEOF (x)))
                                  THEN RETURN (FALSE);
                              END_IF;
    OTHERWISE  : RETURN (TRUE);
  END_CASE;
END_FUNCTION;
(*
```

Argument definitions:

e: identifies the input **sheet_metal_person_assignment** to be checked.

5.2.5.6 sheet_metal_person_and_organization_correlation

The **sheet_metal_person_and_organization_correlation** boolean function returns true if the **name** attribute value of the **person_and_organization_role** entity is coordinated with the type of entity selected in the **items** of the **sheet_metal_person_and_organization_assignment** entity.

EXAMPLE 49 - If the role of a **person_and_organization** is 'data owner' then all of the **items** in the **sheet_metal_-person_and_organization_assignment** shall either be **product_definition** or **process_plan** entities.

EXPRESS specification:

```
*)
FUNCTION sheet_metal_person_and_organization_correlation
  (e : sheet_metal_person_and_organization_assignment ) : BOOLEAN;
  LOCAL
    p_org_role : label;
  END_LOCAL;
  p_org_role := e\person_and_organization_assignment.role.name;
  CASE p_org_role OF
    'data owner'              : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  SIZEOF (['SHEET_METAL.' +
                                  'PRODUCT_DEFINITION',
                                  'SHEET_METAL.PROCESS_PLAN'] *
                                  TYPEOF (x)) = 1))
```

```
                                  THEN RETURN (FALSE);
                                END_IF;
    'planner'                 : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'SHEET_METAL.' +
                                  'PROCESS_PLAN'
                                  IN TYPEOF (x)))
                                  THEN RETURN (FALSE);
                                END_IF;
    'implementor'             : IF SIZEOF (e.items) <>
                                  SIZEOF (QUERY (x <* e.items |
                                  'SHEET_METAL.' +
                                  'EXECUTED_ACTION'
                                  IN TYPEOF (x)))
                                  THEN RETURN (FALSE);
                                END_IF;
    OTHERWISE  : RETURN (TRUE);
  END_CASE;
END_FUNCTION;
(*
```

Argument definitions:

e: identifies the input **sheet_metal_person_and_organization_assignment** to be checked.

```
*)
END_SCHEMA;  -- sheet_metal
(*
```

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;

— ISO 10303-22.

Requirements with respect to implementation methods are specified in annex C.

The Protocol Information Conformance Statement (PICS) proforma lists the options or the combinations of options that may be included in the implementation. The PICS proforma is provided in annex D.

NOTE - ISO/TR 10303-307:—[1)] defines the abstract test suite to be used in the assessment of conformance. ISO 10303-32:—[1)] describes the conformance assessment process.

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. Fourteen conformance classes are defined. Conformance to this part of ISO 10303 requires, as a minimum, conformance to class 1. Options are defined by classes 2 through 14 and may be selected by an implementation. Conformance to a particular conformance class requires that all AIM entities, types and associated constraints defined as part of that class be supported. Support for a particular conformance class requires support of all the options specified in this class.

The conformance classes are characterized as follows:

— Class 1: Product management (PM) and identification information without shape;

— Class 2: Class 1 and sheet metal part process plan data without shape;

— Class 3: Class 1 and shapes represented by topologically bounded wireframe models;

— Class 4: Class 1 and shapes represented by geometrically bounded wireframe and surface models;

— Class 5: Class 1 and shapes represented by manifold surface models with topology;

— Class 6: Class 1 and shapes represented by faceted boundary representation;

[1)] To be published.

— Class 7: Class 1 and shapes represented by advanced boundary representation;

— Class 8: Class 1 and shapes represented by constructive solid geometry;

— Class 9: Class 2 and shapes represented by topologically bounded wireframe models;

— Class 10: Class 2 and shapes represented by geometrically bounded wireframe and surface models;

— Class 11: Class 2 and shapes represented by manifold surface models with topology;

— Class 12: Class 2 and shapes represented by faceted boundary representation;

— Class 13: Class 2 and shapes represented by advanced boundary representation;

— Class 14: Class 2 and shapes represented by constructive solid geometry.

Class 1 is a prerequisite for classes 2 through 14. If an implementation conforms to any of classes 2 through 14, then it shall also conform to class 1. Class 2 is a prerequisite for classes 9 through 14. If an implementation conforms to any of classes 9 through 14, then it shall also conform to class 2.

Conformance classes 3 through 14 are defined in terms of required AIM shape representation subtypes listed in Table 14. Each shape representation subtype specifies in its local domain rules all of the AIM constructs that define the corresponding conformance class.

Table 14 - Conformance options

Shape representation subtype	Class 3 or 9	Class 4 or 10	Class 5 or 11	Class 6 or 12	Class 7 or 13	Class 8 or 14
advanced_ brep_shape_representation					X	
csg_shape_representation						X
edge_ based_ wireframe_shape_representation	X					
faceted_brep_shape_representation				X		
geometrically _bounded_ surface_shape_representation		X				
geometrically_ bounded_ - wireframe_shape_representation		X				

manifold_ - surface_shape_representation			X			
shell_based_wireframe	X					

6.1 Conformance class membership

An implementation of any of the 14 conformance classes of this part of ISO 10303 shall support entities and related constructs marked with an X in the table 15.

Table 15 - Conformance class membership

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
action	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
action_assigned_item (select type)	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
action_assignment	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
action_direcrive	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
action_method	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
action_method_relationship	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
action_method_with_-associated_documents	49	X	X	X	X	X	X	X	X	X	X	X	X	X	X
action_property	49	X	X	X	X	X	X	X	X	X	X	X	X	X	X
action_property_-representation	49	X	X	X	X	X	X	X	X	X	X	X	X	X	X
action_relationship	41		X	X	X	X	X	X	X	X	X	X	X	X	X
action_request_solution	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
action_resource	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
action_resource_relationship	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X

action_resource_requirement	49		X							X	X	X	X	X	X
action_resource_-requirement_relationship	49	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
action_resource_type	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
advanced_-brep_shape_representation	514							X	X					X	X
advanced_face	511				X	X		X	X		X	X		X	X
ahead_or_behind (enumeration type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
angle_relator (enumeration type)	47			X	X	X	X	X	X	X	X	X	X	X	X
angular_location	47			X	X	X	X	X	X	X	X	X	X	X	X
application_context	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
application_context_element	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
application_protocol_-definition	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
approval	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
approval_assigned_item (select type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
approval_assignment	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
approval_date_time	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
approval_person_organization	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
approval_role	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
approval_status	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
assembly_component_usage	44	X	X	X	X	X	X	X	X	X	X	X	X	X	X
assembly_component_usage_-substitute	44	X	X	X	X	X	X	X	X	X	X	X	X	X	X
axis1_placement	42				X	X		X	X		X	X		X	X
axis2_placement (select type)	42			X	X	X	X	X	X	X	X	X	X	X	X
axis2_placement_2d	42				X	X					X	X			
axis2_placement_3d	42			X	X	X	X	X	X	X	X	X	X	X	X
b_spline_curve	42			X	X	X		X	X	X	X	X		X	X
b_spline_curve_form (enumeration type)	42			X	X	X		X	X	X	X	X		X	X
b_spline_curve_with_knots	42			X	X	X		X	X	X	X	X		X	X
b_spline_surface	42				X	X		X	X		X	X		X	X
b_spline_surface_form (enumeration type)	42				X	X		X	X		X	X		X	X
b_spline_surface_with_knots	42				X	X		X	X		X	X		X	X
bezier_curve	42			X	X	X		X	X	X	X	X		X	X
bezier_surface	42				X	X		X	X		X	X		X	X
block	42								X						X
boolean_operand (select type)	42								X						X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
boolean_operator (enumeration type)	42								X						X
boolean_result	42								X						X
boundary_curve	42				X						X				
bounded_curve	42			X	X	X		X	X	X	X	X		X	X
bounded_surface	42				X	X		X	X		X	X		X	X
box_domain	42								X						X
boxed_half_space	42								X						X
brep_with_voids	42						X	X	X				X	X	X
calendar_date	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
cartesian_point	42			X	X	X	X	X	X	X	X	X	X	X	X
cartesian_transformation_-operator	42			X	X	X				X	X	X			
cartesian_transformation_-operator_3d	42			X	X	X				X	X	X			
change_order	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
characterized_action_-definition (select type)	49		X							X	X	X	X	X	X
characterized_definition (select type)	45	X	X	X	X	X	X	X	X	X	X	X	X	X	X
characterized_product_-definition (select type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
characterized_resource_-definition (select type)	49		X							X	X	X	X	X	X
circle	42			X	X	X		X	X	X	X	X		X	X
closed_shell	42					X	X	X	X		X	X	X	X	X
composite_curve	42				X						X				
composite_curve_on_surface	42				X						X				
composite_curve_segment	42				X						X				
configuration_design	44	X	X	X	X	X	X	X	X	X	X	X	X	X	X
configuration_item	44	X	X	X	X	X	X	X	X	X	X	X	X	X	X
conic	42			X	X	X		X		X	X	X		X	
conical_surface	42				X	X		X	X		X	X		X	X
connected_edge_set	42			X						X					
connected_face_set	42					X	X	X	X			X	X	X	X
context_dependent_unit	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
contract	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
contract_assigned_item (select type)	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
contract_assignment	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
contract_type	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
conversion_based_unit	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
coordinated_universal_time_-offset	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
count_measure (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
csg_primitive (select type)	42								X						X
csg_select (select type)	42								X						X
csg_shape_representation	515								X						X
csg_solid	42								X						X
curve	42			X	X	X		X		X	X	X		X	
curve_bounded_surface	42				X				X		X				X
curve_on_surface (select type)	42				X	X					X	X			
curve_replica	42			X	X	X				X	X	X			
cylindrical_surface	42				X	X		X	X		X	X		X	X
date	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
date_and_time	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
date_and_time_assigned_item (select type)	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
date_and_time_assignment	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
date_assigned_item (select type)	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
date_assignment	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
date_role	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
date_time_role	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
date_time_select (select type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
dated_effectivity	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
datum	47			X	X	X	X	X	X	X	X	X	X	X	X
datum_reference	47			X	X	X	X	X	X	X	X	X	X	X	X
datum_target	47			X	X	X	X	X	X	X	X	X	X	X	X
day_in_month_number (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
day_in_week_number (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
day_in_year_number (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
definitional_representation	43				X	X					X	X			
degenerate_pcurve	42				X	X					X	X			
degenerate_toroidal_surface	42				X	X		X	X		X	X		X	X
derived_unit	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
derived_unit_element	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
descriptive_representation_item	45	X	X	X	X	X	X	X	X	X	X	X	X	X	X
die_definition_constraint_relationship	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
dimension_count (type)	42	X	X	X	X	X	X	X	X	X	X	X	X	X	
dimensional_characteristic (select type)	47	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
dimensional_characteristic_representation	47			X	X	X	X	X	X	X	X	X	X	X	X
dimensional_exponents	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
dimensional_location	47			X	X	X	X	X	X	X	X	X	X	X	X
dimensional_location_with_path	47			X	X	X	X	X	X	X	X	X	X	X	X
dimensional_size	47			X	X	X	X	X	X	X	X	X	X	X	X
directed_action	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
direction	42			X	X	X	X		X	X	X	X	X		X
document	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
document_reference	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
document_referenced_item (select type)	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
document_type	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
edge	42			X		X		X	X	X		X		X	X
edge_based_wireframe_model	42			X						X					
edge_based_wireframe_representation	501			X						X					
edge_curve	42			X		X		X	X	X		X		X	X
edge_loop	42			X		X		X	X	X		X		X	X
effectivity	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
elementary_surface	42				X	X	X	X	X		X	X	X	X	X
ellipse	42			X	X	X		X	X	X	X	X		X	X
evaluated_degenerate_pcurve	42				X	X					X	X			
executed_action	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
extruded_area_solid	42								X						X
face	42					X	X	X	X			X	X	X	X
face_bound	42					X	X	X	X			X	X	X	X
face_outer_bound	42					X	X	X	X			X	X	X	X
face_surface	42					X	X	X	X			X	X	X	X
faceted_brep	42						X		X				X		X
faceted_brep_model	512						X		X				X		X
functionally_defined_-transformation	43			X	X	X	X	X	X	X	X	X	X	X	X
geometric_curve_set	42				X						X				
geometric_representation_-context	42	X	X	X	X	X	X	X	X	X	X	X	X	X	
geometric_representation_-item	42			X	X	X	X	X	X	X	X	X	X	X	X
geometric_set	42				X						X				
geometric_set_select (select type)	42			X	X					X	X				

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
geometric_tolerance	47			X	X	X	X	X	X	X	X	X	X	X	X
geometric_tolerance_with_-datum_reference	47			X	X	X	X	X	X	X	X	X	X	X	X
geometrically_bounded_-surface_representation	507				X						X				
geometrically_bounded_-wireframe_representation	510				X						X				
global_uncertainty_assigned_-context	43			X	X	X	X	X	X	X	X	X	X	X	X
half_space_solid	42								X						X
hour_in_day (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
hyperbola	42			X	X	X		X	X	X	X	X		X	X
identifier (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
input_item_die_relationship	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
intersection_curve	42				X	X					X	X			
item_defined_transformation	43			X	X	X	X	X	X	X	X	X	X	X	X
knot_type (enumeration type)	42			X	X	X		X	X	X	X	X		X	X
label (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
length_measure (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
length_measure_with_unit	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
length_unit	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
limit_condition (enumeration type)	47	X	X	X	X	X	X	X	X	X	X	X	X	X	X
limits_and_fits	47			X	X	X	X	X	X	X	X	X	X	X	X
line	42			X	X	X		X	X	X	X	X		X	X
local_time	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
loop	42			X		X	X	X	X	X		X	X	X	X
make_from_usage_option	44	X	X	X	X	X	X	X	X	X	X	X	X	X	X
make_from_usage_option_-group	44	X	X	X	X	X	X	X	X	X	X	X	X	X	X
manifold_solid_brep	42						X	X	X				X	X	X
manifold_surface_shape_-representation	509					X						X			
mapped_item	43			X			X	X	X	X			X	X	X
mass_measure (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
material_property	45	X	X	X	X	X	X	X	X	X	X	X	X	X	X
measure_representation_item	47			X	X	X	X	X	X	X	X	X	X	X	X
measure_value (select type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
measure_with_unit	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
minute_in_hour (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
modified_geometric_tolerance	47			X	X	X	X	X	X	X	X	X	X	X	X
month_in_year_number (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
named_unit	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
offset_curve_3d	42			X	X	X				X	X	X			
offset_surface	42				X	X					X	X			
open_shell	42					X						X			
ordinal_date	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
organization	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
organization_assigned_item (select type)	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
organization_assignment	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
organization_role	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
oriented_closed_shell	42						X	X	X				X	X	X
oriented_edge	42			X		X		X	X	X		X		X	X
oriented_face	42					X	X	X	X			X	X	X	X
oriented_open_shell	42								X						X
oriented_path	42								X						X
outer_boundary_curve	42				X						X				
parabola	42			X	X	X		X	X	X	X	X		X	X
parameter_value (type)	42			X	X				X	X	X				X
parametric_representation_-context	43				X	X					X	X			

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
path	42			X		X		X	X	X		X		X	X
pcurve	42				X	X					X	X			
pcurve_or_surface (select type)	42				X	X					X	X			
person	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
person_and_organization	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
person_and_organization_-assigned_item (select type)	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
person_and_organization_-assignment	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
person_and_organization_role	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
person_assigned_item (select type)	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
person_assignment	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
person_organization_select (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
person_role	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
physically_modelled_-product_definition	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
placement	42			X	X	X	X	X	X	X	X	X	X	X	X
plane	42					X	X	X	X			X	X	X	X
plane_angle_measure (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
plane_angle_measure_with_-unit (entity)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
plus_minus_tolerance	47			X	X	X	X	X	X	X	X	X	X	X	X
point	42			X	X	X	X	X	X	X	X	X	X	X	X
point_on_curve	42			X	X	X				X	X	X			
point_on_surface	42				X	X					X	X			
point_replica	42			X	X					X	X				
poly_loop	42						X		X				X		X
polyline	42			X	X	X		X	X	X	X	X		X	X
positive_length_measure (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
positive_plane_angle_-measure (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
preferred_surface_curve_-representation (enumeration type)	42				X	X					X	X			
process_plan	207		X							X	X	X	X	X	X
process_product_association	49	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_category	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_category_relationship	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_concept	44	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
product_concept_context	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_context	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition_context	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition_effectivity	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition_formation	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition_-formation_with_specified_-source	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition_process	49	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition_-relationship	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition_shape	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition_substitute	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition_usage	44	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_definition_with_-associated_documents	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_related_product_-category	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
product_type	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
promissory_usage_occurrence	44	X	X	X	X	X	X	X	X	X	X	X	X	X	X
property_definition	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
property_definition_-representation	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
quantified_assembly_-component_usage	44	X	X	X	X	X	X	X	X	X	X	X	X	X	X
quasi_uniform_curve	42			X	X	X		X	X	X	X	X		X	X
quasi_uniform_surface	42				X	X		X	X		X	X		X	X
rational_b_spline_curve	42			X	X	X		X	X	X	X	X		X	X
rational_b_spline_surface	42				X	X		X	X		X	X		X	X
rectangular_composite_-surface	42				X						X				
rectangular_trimmed_surface	42				X						X				
reparametrised_composite_-curve_segment	42				X						X				
replacement_relationship	49	X	X	X	X	X	X	X	X	X	X	X	X	X	X
representation	43			X	X	X	X	X	X	X	X	X	X	X	X
representation_context	43			X	X	X	X	X	X	X	X	X	X	X	X
representation_item	43			X	X	X	X	X	X	X	X	X	X	X	X
representation_map	43			X	X	X	X	X	X	X	X	X	X	X	X
representation_relationship	43			X	X	X	X	X	X	X	X	X	X	X	X
representation_relationship_-with_transformation	43			X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
requirement_for_action_-resource	41		X							X	X	X	X	X	X
resource_property	49		X	X	X	X	X	X	X	X	X	X	X	X	X
resource_property_-representation	49		X	X	X	X	X	X	X	X	X	X	X	X	X
resource_requirement_type	49		X							X	X	X	X	X	X
revolved_area_solid	42								X						X
right_angular_wedge	42								X						X
right_circular_cone	42								X						X
right_circular_cylinder	42								X						X
seam_curve	42				X	X					X	X			
second_in_minute	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
security_classification	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
security_classification_-assigned_item (select type)	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
security_classification_-assignment	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
security_classification_level	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
sequenced_product_-definition_relationship	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
sequential_method	49		X							X	X	X	X	X	X
serial_action_method	49		X							X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
serial_numbered_effectivity	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
shape_aspect	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
shape_aspect_relationship	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
shape_definition (select type)	41			X	X	X	X	X	X	X	X	X	X	X	X
shape_definition_-representation	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
shape_dimension_-representation	47			X	X	X	X	X	X	X	X	X	X	X	X
shape_representation	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
shape_representation_-relationship	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
sheet_metal_action_-assignment	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
sheet_metal_approval_-assignment	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
sheet_metal_contract_-assignment	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
sheet_metal_date_and_time_-assignment	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
sheet_metal_date_assignment	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
sheet_metal_document_-reference	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
sheet_metal_organization_-assignment	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
sheet_metal_person_and_-organization_assignment	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
sheet_metal_person_-assignment	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
sheet_metal_security_-classification_assignment	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
shell (select type)	42			X		X				X		X			
shell_based_surface_model	42					X						X			
shell_based_wireframe_-model	42			X						X					
shell_based_wireframe_-representation	502			X						X					
si_prefix (enumeration type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
si_unit	41	X	X												
si_unit_name (enumeration type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
solid_model	42						X	X	X				X	X	X
solid_replica	42								X						X
source (enumeration type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
sphere	42								X						X
spherical_surface	42				X	X		X	X		X	X		X	X
start_order	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
supported_item (select type)	41		X							X	X	X	X	X	X
surface	42				X	X	X	X	X		X	X	X	X	X
surface_curve	42				X	X					X	X			
surface_of_linear_extrusion	42				X	X		X	X		X	X		X	X
surface_of_revolution	42				X	X		X	X		X	X		X	X
surface_patch	42				X						X				
surface_replica	42				X	X					X	X			
swept_area_solid	42								X						X
swept_surface	42				X	X		X	X		X	X		X	X
text (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
tolerance_method_definition (select type)	47	X	X	X	X	X	X	X	X	X	X	X	X	X	X
tolerance_value	47			X	X	X	X	X	X	X	X	X	X	X	X
tolerance_zone	47			X	X	X	X	X	X	X	X	X	X	X	X
tolerance_zone_definition	47			X	X	X	X	X	X	X	X	X	X	X	X
tolerance_zone_form	47			X	X	X	X	X	X	X	X	X	X	X	X
topological_representation_item	42			X		X	X	X	X	X		X	X	X	X
toroidal_surface	42				X	X		X	X		X	X		X	X
torus	42								X						X

Table 15 - Conformance class membership (continued)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
transformation (select type)	43			X	X	X	X	X	X	X	X	X	X	X	X
transition_code (enumeration type)	42				X						X				
trimmed_curve	42				X						X				
trimming_preference (enumeration type)	42				X						X				
trimming_select (select type)	42			X	X					X	X				
uncertainty_measure_with_-unit	43			X	X	X	X	X	X	X	X	X	X	X	X
uniform_curve	42			X	X	X		X	X	X	X	X		X	X
uniform_surface	42				X	X		X	X		X	X		X	X
unit (select type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
vector	42			X	X	X		X	X	X	X	X		X	X
vector_or_direction (select type)	42								X						X

Table 15 - Conformance class membership (concluded)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
versioned_action_request	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
vertex	42			X		X		X	X	X		X		X	X

Table 15 - Conformance class membership (concluded)

AIM element	Source part	Conformance classes in which AIM element is found													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
vertex_loop	42			X		X		X	X	X		X		X	X
vertex_point	42			X		X		X	X	X		X		X	X
vertex_shell	42			X		X				X		X			
week_in_year_number (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X
week_of_year_and_day_date	41	X	X												
wire_shell	42			X						X					
work_order	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
work_order_relationship	207	X	X	X	X	X	X	X	X	X	X	X	X	X	X
year_number (type)	41	X	X	X	X	X	X	X	X	X	X	X	X	X	X

NOTE - ISO/TR 10303-307:—[1)] defines the abstract test suite to be used in the assessment of conformance. ISO 10303-32:—[1)] describes the conformance assessment process.

[1)] To be published.

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 sheet_metal;

  TYPE action_assigned_item = SELECT
    (executed_action,
     process_plan,
     product_definition,
     product_definition_formation);
  END_TYPE; -- action_assigned_item

  TYPE ahead_or_behind = ENUMERATION OF
    (ahead,
     behind);
  END_TYPE; -- ahead_or_behind

  TYPE angle_relator = ENUMERATION OF
    (equal,
     large,
     small);
  END_TYPE; -- angle_relator

  TYPE approval_assigned_item = SELECT
    (executed_action,
     process_plan,
     product_definition,
     product_definition_formation,
     sheet_metal_action_assignment);
  END_TYPE; -- approval_assigned_item

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

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

```
TYPE b_spline_surface_form = ENUMERATION OF
  (plane_surf,
   conical_surf,
   cylindrical_surf,
   generalised_cone,
   quadric_surf,
   ruled_surf,
   spherical_surf,
   surf_of_linear_extrusion,
   surf_of_revolution,
   toroidal_surf,
   unspecified);
END_TYPE; -- b_spline_surface_form

TYPE boolean_operand = SELECT
  (boolean_result,
   csg_primitive,
   half_space_solid,
   solid_model);
END_TYPE; -- boolean_operand

TYPE boolean_operator = ENUMERATION OF
  (difference,
   intersection,
   union);
END_TYPE; -- boolean_operator

TYPE characterized_action_definition = SELECT
  (action,
   action_method,
   action_method_relationship,
   action_relationship);
END_TYPE; -- characterized_action_definition

TYPE characterized_definition = SELECT
  (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 characterized_resource_definition = SELECT
  (action_resource,
   action_resource_relationship,
   action_resource_requirement,
   action_resource_requirement_relationship);
END_TYPE; -- characterized_resource_definition

TYPE contract_assigned_item = SELECT
  (change_order,
   executed_action,
   product_definition_formation_with_specified_source,
   start_order);
```

```
END_TYPE; -- contract_assigned_item

TYPE count_measure = NUMBER;
END_TYPE; -- count_measure

TYPE csg_primitive = SELECT
  (block,
   right_angular_wedge,
   right_circular_cone,
   right_circular_cylinder,
   sphere,
   torus);
END_TYPE; -- csg_primitive

TYPE csg_select = SELECT
  (boolean_result,
   csg_primitive);
END_TYPE; -- csg_select

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

TYPE date_and_time_assigned_item = SELECT
  (executed_action,
   process_plan,
   product_concept,
   product_definition,
   product_definition_formation,
   sheet_metal_action_assignment,
   versioned_action_request);
END_TYPE; -- date_and_time_assigned_item

TYPE date_assigned_item = SELECT
  (executed_action,
   process_plan,
   product_concept,
   product_definition,
   product_definition_formation,
   sheet_metal_action_assignment,
   versioned_action_request);
END_TYPE; -- date_assigned_item

TYPE date_time_select = SELECT
  (date,
   date_and_time,
   local_time);
END_TYPE; -- date_time_select

TYPE day_in_month_number = INTEGER;
END_TYPE; -- day_in_month_number

TYPE day_in_week_number = INTEGER;
WHERE
```

```
  WR1: (1 <= SELF) AND (SELF <= 7);
END_TYPE; -- day_in_week_number

TYPE day_in_year_number = INTEGER;
END_TYPE; -- day_in_year_number

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

TYPE dimensional_characteristic = SELECT
  (dimensional_location,
   dimensional_size);
END_TYPE; -- dimensional_characteristic

TYPE document_referenced_item = SELECT
  (action_resource_requirement,
   executed_action,
   process_plan,
   product_definition,
   sheet_metal_action_assignment);
END_TYPE; -- document_referenced_item

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

TYPE hour_in_day = INTEGER;
WHERE
  WR1: (0 <= SELF) AND (SELF < 24);
END_TYPE; -- hour_in_day

TYPE identifier = STRING;
END_TYPE; -- identifier

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

TYPE label = STRING;
END_TYPE; -- label

TYPE length_measure = REAL;
END_TYPE; -- length_measure

TYPE limit_condition = ENUMERATION OF
  (least_material_condition,
   maximum_material_condition,
   regardless_of_feature_size);
END_TYPE; -- limit_condition
```

```
TYPE list_of_reversible_topology_item = LIST [0:?] OF
            reversible_topology_item;
END_TYPE; -- list_of_reversible_topology_item

TYPE mass_measure = REAL;
END_TYPE; -- mass_measure

TYPE measure_value = SELECT
  (count_measure,
   length_measure,
   mass_measure,
   parameter_value,
   plane_angle_measure,
   positive_length_measure,
   positive_plane_angle_measure);
END_TYPE; -- measure_value

TYPE minute_in_hour = INTEGER;
WHERE
  WR1: (0 <= SELF) AND (SELF <= 59);
END_TYPE; -- minute_in_hour

TYPE month_in_year_number = INTEGER;
WHERE
  WR1: (1 <= SELF) AND (SELF <= 12);
END_TYPE; -- month_in_year_number

TYPE organization_assigned_item = SELECT
  (change_order,
   contract,
   executed_action,
   product_definition,
   start_order);
END_TYPE; -- organization_assigned_item

TYPE parameter_value = REAL;
END_TYPE; -- parameter_value

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

TYPE person_and_organization_assigned_item = SELECT
  (change_order,
   process_plan,
   product_definition);
END_TYPE; -- person_and_organization_assigned_item

TYPE person_assigned_item = SELECT
  (change_order,
   contract,
   product_definition,
   shape_aspect,
   versioned_action_request);
END_TYPE; -- person_assigned_item
```

```
TYPE person_organization_select = SELECT
  (organization,
   person,
   person_and_organization);
END_TYPE; -- person_organization_select

TYPE plane_angle_measure = REAL;
END_TYPE; -- plane_angle_measure

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

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

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

TYPE reversible_topology = SELECT
  (list_of_reversible_topology_item,
   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 second_in_minute = REAL;
WHERE
  WR1: (0 <= SELF) AND (SELF < 60);
END_TYPE; -- second_in_minute

TYPE security_classification_assigned_item = SELECT
  (process_plan,
   product_definition);
END_TYPE; -- security_classification_assigned_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
  (closed_shell,
   open_shell,
   vertex_shell,
   wire_shell);
END_TYPE; -- shell

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

TYPE si_unit_name = ENUMERATION OF
  (ampere,
   becquerel,
   candela,
   coulomb,
   degree_celsius,
   farad,
   gram,
   gray,
   henry,
   hertz,
   joule,
   kelvin,
   lumen,
   lux,
   metre,
   mole,
   newton,
   ohm,
   pascal,
   radian,
   second,
   siemens,
   sievert,
```

```
    steradian,
    tesla,
    volt,
    watt,
    weber);
END_TYPE; -- si_unit_name

TYPE source = ENUMERATION OF
  (bought,
   made,
   not_known);
END_TYPE; -- source

TYPE supported_item = SELECT
  (action_directive,
   action,
   action_method);
END_TYPE; -- supported_item

TYPE text = STRING;
END_TYPE; -- text

TYPE tolerance_method_definition = SELECT
  (limits_and_fits,
   tolerance_value);
END_TYPE; -- tolerance_method_definition

TYPE transformation = SELECT
  (functionally_defined_transformation,
   item_defined_transformation);
END_TYPE; -- transformation

TYPE transition_code = ENUMERATION OF
  (cont_same_gradient,
   cont_same_gradient_same_curvature,
   continuous,
   discontinuous);
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
  (derived_unit,
   named_unit);
END_TYPE; -- unit

TYPE vector_or_direction = SELECT
```

```
    (direction,
     vector);
  END_TYPE; -- vector_or_direction

  TYPE week_in_year_number = INTEGER;
  WHERE
    WR1: (1 <= SELF) AND (SELF <= 52);
  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_directive;
      name        : label;
      description : text;
      analysis    : text;
      comment     : text;
      requests    : SET [1:?] OF versioned_action_request;
  END_ENTITY; -- action_directive

  ENTITY action_method;
      name        : label;
      description : text;
      consequence : text;
      purpose     : text;
  END_ENTITY; -- action_method

  ENTITY action_method_relationship;
      name            : label;
      description     : text;
      relating_method : action_method;
      related_method  : action_method;
  END_ENTITY; -- action_method_relationship

  ENTITY action_method_with_associated_documents
    SUBTYPE OF (action_method);
     documents : SET [1:?] OF document;
  END_ENTITY; -- action_method_with_associated_documents

  ENTITY action_property;
      name        : label;
      description : text;
      definition  : characterized_action_definition;
  END_ENTITY; -- action_property
```

```
ENTITY action_property_representation;
    name           : label;
    description    : text;
    property       : action_property;
    representation : representation;
END_ENTITY; -- action_property_representation

ENTITY action_relationship;
    name            : label;
    description     : text;
    relating_action : action;
    related_action  : action;
END_ENTITY; -- action_relationship

ENTITY action_request_solution;
    method  : action_method;
    request : versioned_action_request;
END_ENTITY; -- action_request_solution

ENTITY action_resource;
    name        : label;
    description : text;
    usage       : SET [1:?] OF supported_item;
    kind        : action_resource_type;
END_ENTITY; -- action_resource

ENTITY action_resource_relationship;
name      : label;
description     : text;
relating_resource : action_resource;
related_resource      : action_resource;
END_ENTITY; -- action_resource_relationship

ENTITY action_resource_requirement;
    name        : label;
    description : text;
    kind        : resource_requirement_type;
    operations  : SET [1:?] OF characterized_action_definition;
END_ENTITY; -- action_resource_requirement

ENTITY action_resource_requirement_relationship;
name      : label;
description     : text;
relating_action_resource_requirement : action_resource_requirement;
related_action_resource_requirement      : action_resource_requirement;
  WHERE
    WR1: relating_action_resource_requirement :<>:
           related_action_resource_requirement;
END_ENTITY; -- action_resource_requirement_relationship

ENTITY action_resource_type;
    name : label;
END_ENTITY; -- action_resource_type

ENTITY advanced_brep_shape_representation
  SUBTYPE OF (shape_representation);
```

```
  WHERE
    WR1: SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
              'SHEET_METAL.MANIFOLD_SOLID_BREP','SHEET_METAL.FACETED_BREP',
              'SHEET_METAL.MAPPED_ITEM','SHEET_METAL.AXIS2_PLACEMENT_3D']
              * TYPEOF(it)) = 1)) )) = 0;
    WR2: SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
              'SHEET_METAL.MANIFOLD_SOLID_BREP','SHEET_METAL.MAPPED_ITEM']
              * TYPEOF(it)) = 1) )) > 0;
    WR3: SIZEOF(QUERY ( msb <* QUERY ( it <* SELF.items | (
              'SHEET_METAL.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) ) | (NOT (
              SIZEOF(QUERY ( csh <* msb_shells(msb,'AIC_ADVANCED_BREP') |
              (NOT (SIZEOF(QUERY ( fcs <* csh\connected_face_set.cfs_faces
               | (NOT ('SHEET_METAL.ADVANCED_FACE' IN TYPEOF(fcs))) )) =
              0)))) = 0)) )) = 0;
    WR4: SIZEOF(QUERY ( msb <* QUERY ( it <* SELF.items | (
              'SHEET_METAL.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) ) | (
              'SHEET_METAL.ORIENTED_CLOSED_SHELL' IN TYPEOF(msb\
              manifold_solid_brep.outer)) )) = 0;
    WR5: SIZEOF(QUERY ( brv <* QUERY ( it <* SELF.items | (
              'SHEET_METAL.BREP_WITH_VOIDS' IN TYPEOF(it)) ) | (NOT (
              SIZEOF(QUERY ( csh <* brv\brep_with_voids.voids | csh\
              oriented_closed_shell.orientation )) = 0)) )) = 0;
    WR6: SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
              'SHEET_METAL.MAPPED_ITEM' IN TYPEOF(it)) ) | (NOT (
              'SHEET_METAL.ADVANCED_BREP_SHAPE_REPRESENTATION' IN TYPEOF(
              mi\mapped_item.mapping_source.mapped_representation))) )) =
              0;
END_ENTITY; -- advanced_brep_shape_representation

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

```
     WR5 : SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* SELF.bounds | (
               'SHEET_METAL.EDGE_LOOP' IN TYPEOF(bnds.bound)) ) | (
               'SHEET_METAL.ORIENTED_PATH' IN TYPEOF(elp_fbnds.bound)) ))
               = 0;
     WR6 : (NOT ('SHEET_METAL.SWEPT_SURFACE' IN TYPEOF(face_geometry)))
               OR (SIZEOF(['SHEET_METAL.LINE','SHEET_METAL.CONIC',
               'SHEET_METAL.POLYLINE','SHEET_METAL.B_SPLINE_CURVE'] *
               TYPEOF(face_geometry\swept_surface.swept_curve)) = 1);
     WR7 : SIZEOF(QUERY ( vlp_fbnds <* QUERY ( bnds <* SELF.bounds | (
               'SHEET_METAL.VERTEX_LOOP' IN TYPEOF(bnds.bound)) ) | (NOT (
               ('SHEET_METAL.VERTEX_POINT' IN TYPEOF(vlp_fbnds\face_bound.
               bound\vertex_loop.loop_vertex)) AND (
               'SHEET_METAL.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([
               'SHEET_METAL.EDGE_LOOP','SHEET_METAL.VERTEX_LOOP'] *
               TYPEOF(bnd.bound)) = 1)) )) = 0;
     WR9 : SIZEOF(QUERY ( elp_fbnds <* QUERY ( bnds <* SELF.bounds | (
               'SHEET_METAL.EDGE_LOOP' IN TYPEOF(bnds.bound)) ) | (NOT (
               SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.edge_list | ((
               'SHEET_METAL.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 (
               'SHEET_METAL.PCURVE' IN TYPEOF(sc_ag))) )) = 0))) )) = 0)) ))
               = 0;
     WR10: ((NOT ('SHEET_METAL.SWEPT_SURFACE' IN TYPEOF(face_geometry)))
               OR (NOT ('SHEET_METAL.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 | (
               'SHEET_METAL.EDGE_LOOP' IN TYPEOF(bnds.bound)) ) | (NOT (
               SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.edge_list | ((
               'SHEET_METAL.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 angular_location
  SUBTYPE OF (dimensional_location);
    angle_selection : angle_relator;
END_ENTITY; -- angular_location

ENTITY application_context;
    application : text;
  INVERSE
    context_elements : SET [1:?] OF application_context_element FOR
                          frame_of_reference;
END_ENTITY; -- application_context

ENTITY application_context_element
  SUPERTYPE OF (ONEOF (product_context,product_definition_context,
      product_concept_context));
    name               : label;
```

```
    frame_of_reference : application_context;
END_ENTITY; -- application_context_element

ENTITY application_protocol_definition;
    status                                  : label;
    application_interpreted_model_schema_name : label;
    application_protocol_year               : year_number;
    application                             : application_context;
END_ENTITY; -- application_protocol_definition

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

ENTITY assembly_component_usage_substitute;
    name       : label;
    definition : text;
    base       : assembly_component_usage;
    substitute : assembly_component_usage;
  UNIQUE
    ur1 : base, substitute;
  WHERE
    WR1: base.relating_product_definition :=: substitute.
              relating_product_definition;
    WR2: base :<>: substitute;
```

```
END_ENTITY; -- assembly_component_usage_substitute

ENTITY axis1_placement
  SUBTYPE OF (placement);
    axis : OPTIONAL direction;
  DERIVE
    z : direction := NVL(normalise(axis),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: ('SHEET_METAL.UNIFORM_CURVE' IN TYPEOF(SELF)) OR (
              'SHEET_METAL.QUASI_UNIFORM_CURVE' IN TYPEOF(SELF)) OR (
              'SHEET_METAL.BEZIER_CURVE' IN TYPEOF(SELF)) OR (
              'SHEET_METAL.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: ('SHEET_METAL.UNIFORM_SURFACE' IN TYPEOF(SELF)) OR (
              'SHEET_METAL.QUASI_UNIFORM_SURFACE' IN TYPEOF(SELF)) OR (
              'SHEET_METAL.BEZIER_SURFACE' IN TYPEOF(SELF)) OR (
              'SHEET_METAL.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        : length_measure;
    y        : length_measure;
    z        : 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, composite_curve_segment))
  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 box_domain;
    corner  : cartesian_point;
    xlength : length_measure;
    ylength : length_measure;
    zlength : length_measure;
  WHERE
    WR1: SIZEOF(QUERY ( item <* USEDIN(SELF,'') | (NOT (
              'SHEET_METAL.BOXED_HALF_SPACE' IN TYPEOF(item))) )) = 0;
END_ENTITY; -- box_domain
```

```
ENTITY boxed_half_space
  SUBTYPE OF (half_space_solid);
    enclosure : box_domain;
END_ENTITY; -- boxed_half_space

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

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_order
  SUBTYPE OF (work_order);
END_ENTITY; -- change_order

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

```
ENTITY closed_shell
  SUBTYPE OF (connected_face_set);
END_ENTITY; -- closed_shell

ENTITY 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
    SUPERTYPE OF (reparametrised_composite_curve_segment)
    SUBTYPE OF (bounded_curve);
    transition   : transition_code;
    same_sense   : BOOLEAN;
    parent_curve : curve;
  INVERSE
    using_curves : BAG [1:?] OF composite_curve FOR segments;
  WHERE
    WR1: 'SHEET_METAL_BOUNDED_CURVE' IN TYPEOF(parent_curve);
END_ENTITY; -- composite_curve_segment

ENTITY configuration_design;
    configuration : configuration_item;
    design        : product_definition_formation;
  UNIQUE
    ur1 : configuration, design;
END_ENTITY; -- configuration_design

ENTITY configuration_item;
    id           : identifier;
    name         : label;
    description  : OPTIONAL text;
    item_concept : product_concept;
    purpose      : OPTIONAL label;
  UNIQUE
    ur1 : id;
```

```
END_ENTITY; -- configuration_item

ENTITY conic
  SUPERTYPE OF (ONEOF (circle,ellipse,hyperbola,parabola))
  SUBTYPE OF (curve);
    position : axis2_placement;
END_ENTITY; -- conic

ENTITY conical_surface
  SUBTYPE OF (elementary_surface);
    radius     : length_measure;
    semi_angle : plane_angle_measure;
  WHERE
    WR1: radius >= 0;
END_ENTITY; -- conical_surface

ENTITY connected_edge_set
  SUBTYPE OF (topological_representation_item);
    ces_edges : SET [1:?] OF edge;
END_ENTITY; -- connected_edge_set

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

ENTITY context_dependent_unit
  SUBTYPE OF (named_unit);
    name : label;
END_ENTITY; -- context_dependent_unit

ENTITY contract;
    name    : label;
    purpose : text;
    kind    : contract_type;
END_ENTITY; -- contract

ENTITY contract_assignment
  ABSTRACT SUPERTYPE;
    assigned_contract : contract;
END_ENTITY; -- contract_assignment

ENTITY contract_type;
    description : label;
END_ENTITY; -- contract_type

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

ENTITY coordinated_universal_time_offset;
    hour_offset   : hour_in_day;
    minute_offset : OPTIONAL minute_in_hour;
```

```
    sense         : ahead_or_behind;
END_ENTITY; -- coordinated_universal_time_offset

ENTITY csg_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    WR1: SELF.context_of_items\geometric_representation_context.
              coordinate_space_dimension = 3;
    WR2: SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
              'SHEET_METAL.CSG_SOLID','SHEET_METAL.EXTRUDED_AREA_SOLID',
              'SHEET_METAL.REVOLVED_AREA_SOLID',
              'SHEET_METAL.SOLID_REPLICA','SHEET_METAL.MAPPED_ITEM',
              'SHEET_METAL.AXIS2_PLACEMENT_3D'] * TYPEOF(it)) = 1)) )) = 0;
    WR3: SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
              'SHEET_METAL.CSG_SOLID','SHEET_METAL.EXTRUDED_AREA_SOLID',
              'REVOLVED_AREA_SOLID','SHEET_METAL.SOLID_REPLICA',
              'SHEET_METAL.MAPPED_ITEM'] * TYPEOF(it)) = 1)) ) >= 1;
    WR4: SIZEOF(QUERY ( it <* SELF.items | (('SHEET_METAL.MAPPED_ITEM'
              IN TYPEOF(it)) AND (NOT (SIZEOF([
              'SHEET_METAL.CSG_SHAPE_REPRESENTATION',
              'SHEET_METAL.FACETED_BREP_SHAPE_REPRESENTATION',
              'SHEET_METAL.ADVANCED_BREP_SHAPE_REPRESENTATION'] * TYPEOF(
              it\mapped_item.mapping_source.mapped_representation)) = 1))) ))
              = 0;
END_ENTITY; -- csg_shape_representation

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 ('SHEET_METAL.OUTER_BOUNDARY_CURVE' IN
              TYPEOF(boundaries)));
    WR2: (NOT implicit_outer) OR ('SHEET_METAL.BOUNDED_SURFACE' IN
              TYPEOF(basis_surface));
    WR3: SIZEOF(QUERY ( temp <* boundaries | (
              'SHEET_METAL.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_and_time;
    date_component : date;
    time_component : local_time;
END_ENTITY; -- date_and_time

ENTITY date_and_time_assignment
  ABSTRACT SUPERTYPE;
    assigned_date_and_time : date_and_time;
    role                   : date_time_role;
END_ENTITY; -- date_and_time_assignment

ENTITY date_assignment
  ABSTRACT SUPERTYPE;
    assigned_date : date;
    role          : date_role;
END_ENTITY; -- date_assignment

ENTITY date_role;
    name : label;
END_ENTITY; -- date_role

ENTITY date_time_role;
    name : label;
END_ENTITY; -- date_time_role

ENTITY dated_effectivity
  SUBTYPE OF (effectivity);
    effectivity_start_date : date_and_time;
    effectivity_end_date   : OPTIONAL date_and_time;
END_ENTITY; -- dated_effectivity

ENTITY datum
  SUBTYPE OF (shape_aspect);
    identification : label;
  INVERSE
    established_by_relationships : SET [1:?] OF
                                     shape_aspect_relationship FOR
                                   related_shape_aspect;
```

```
  WHERE
    WR1: SIZEOF(QUERY ( x <* SELF.established_by_relationships | (
              SIZEOF(TYPEOF(x.relating_shape_aspect) * [
              'SHEET_METAL.DATUM_FEATURE',
              'SHEET_METAL.DATUM_TARGET']) <> 1) )) = 0;
    WR2: SIZEOF(QUERY ( x <* SELF.established_by_relationships | (
              SIZEOF(TYPEOF(x.relating_shape_aspect) * [
              'SHEET_METAL.DATUM_FEATURE',
              'SHEET_METAL.DATUM_TARGET_FEATURE']) <> (TYPEOF(SELF.
              established_by_relationships[1].relating_shape_aspect) * [
              'SHEET_METAL.DATUM_FEATURE',
              'SHEET_METAL.DATUM_TARGET_FEATURE'])) )) = 0;
END_ENTITY; -- datum

ENTITY datum_reference;
    precedence       : INTEGER;
    referenced_datum : datum;
  WHERE
    WR1: precedence > 0;
END_ENTITY; -- datum_reference

ENTITY datum_target
  SUBTYPE OF (shape_aspect);
  target_id             : identifier;
  INVERSE
  target_basis_relationship : shape_aspect_relationship FOR
                              relating_shape_aspect;
WHERE
   WR1: SIZEOF (QUERY (sar<* bag_to_set (USEDIN (SELF,
      'PRODUCT_PROPERTY_DEFINITION_SCHEMA.SHAPE_ASPECT_RELATIONSHIP.' +
        'RELATING_SHAPE_ASPECT'))
      | NOT ('SHAPE_ASPECT_DEFINITION_SCHEMA.DATUM' IN TYPEOF
      (sar.related_shape_aspect)))=0;
   WR2: SELF.product_definitional = TRUE;
END_ENTITY; -- datum_target

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

ENTITY degenerate_pcurve
  SUBTYPE OF (point);
    basis_surface      : surface;
    reference_to_curve : definitional_representation;
  WHERE
    WR1: SIZEOF(reference_to_curve\representation.items) = 1;
    WR2: 'SHEET_METAL.CURVE' IN TYPEOF(reference_to_curve\representation
              .items[1]);
    WR3: reference_to_curve\representation.items[1]\
              geometric_representation_item.dim = 2;
END_ENTITY; -- degenerate_pcurve

ENTITY degenerate_toroidal_surface
```

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

ENTITY derived_unit;
    elements : SET [1:?] OF derived_unit_element;
  WHERE
    WR1: (SIZEOF(elements) > 1) OR ((SIZEOF(elements) = 1) AND (elements
              [1].exponent <> 1));
END_ENTITY; -- derived_unit

ENTITY derived_unit_element;
    unit     : named_unit;
    exponent : REAL;
END_ENTITY; -- derived_unit_element

ENTITY descriptive_representation_item
  SUBTYPE OF (representation_item);
    description : text;
END_ENTITY; -- descriptive_representation_item

ENTITY die_definition_constraint_relationship
  SUBTYPE OF (product_definition_relationship);
  WHERE
    WR1: SELF\product_definition_relationship.related_product_definition
              .frame_of_reference.life_cycle_stage =
              'as design constrained';
    WR2: product_type_of_product(SELF.relating_product_definition.
              formation.of_product,product_type).name IN ['die'];
    WR3: product_type_of_product(SELF.related_product_definition.
              formation.of_product,product_type).name IN ['die'];
END_ENTITY; -- die_definition_constraint_relationship

ENTITY dimensional_characteristic_representation;
    dimension      : dimensional_characteristic;
    representation : shape_dimension_representation;
END_ENTITY; -- dimensional_characteristic_representation

ENTITY dimensional_exponents;
    length_exponent                     : REAL;
    mass_exponent                       : REAL;
    time_exponent                       : REAL;
    electric_current_exponent           : REAL;
    thermodynamic_temperature_exponent : REAL;
    amount_of_substance_exponent        : REAL;
    luminous_intensity_exponent         : REAL;
END_ENTITY; -- dimensional_exponents

ENTITY dimensional_location
  SUPERTYPE OF (ONEOF (angular_location,dimensional_location_with_path))
  SUBTYPE OF (shape_aspect_relationship);
END_ENTITY; -- dimensional_location

ENTITY dimensional_location_with_path
```

```
  SUBTYPE OF (dimensional_location);
    path : shape_aspect;
END_ENTITY; -- dimensional_location_with_path

ENTITY dimensional_size;
    applies_to : shape_aspect;
    name       : label;
  WHERE
    WR1: applies_to.product_definitional = TRUE;
END_ENTITY; -- dimensional_size

ENTITY directed_action
  SUBTYPE OF (executed_action);
    directive : action_directive;
END_ENTITY; -- directed_action

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

ENTITY 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 edge
  SUPERTYPE OF (ONEOF (edge_curve,oriented_edge))
  SUBTYPE OF (topological_representation_item);
    edge_start : vertex;
    edge_end   : vertex;
END_ENTITY; -- edge

ENTITY edge_based_wireframe_model
  SUBTYPE OF (geometric_representation_item);
    ebwm_boundary : SET [1:?] OF connected_edge_set;
END_ENTITY; -- edge_based_wireframe_model

ENTITY edge_based_wireframe_shape_representation
  SUBTYPE OF (shape_representation);
```

```
WHERE
  WR1 : SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
            'SHEET_METAL.EDGE_BASED_WIREFRAME_MODEL',
            'SHEET_METAL.MAPPED_ITEM','SHEET_METAL.AXIS2_PLACEMENT_3D']
            * TYPEOF(it)) = 1)) )) = 0;
  WR2 : SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
            'SHEET_METAL.EDGE_BASED_WIREFRAME_MODEL',
            'SHEET_METAL.MAPPED_ITEM'] * TYPEOF(it)) = 1) )) >= 1;
  WR3 : SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.EDGE_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
             | (NOT (SIZEOF(QUERY ( eb <* ebwm\
            edge_based_wireframe_model.ebwm_boundary | (NOT (SIZEOF(
            QUERY ( edges <* eb.ces_edges | (NOT (
            'SHEET_METAL.EDGE_CURVE' IN TYPEOF(edges))) )) =
            0)) )) = 0)) )) = 0;
  WR4 : SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.EDGE_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
             | (NOT (SIZEOF(QUERY ( eb <* ebwm\
            edge_based_wireframe_model.ebwm_boundary | (NOT (SIZEOF(
            QUERY ( pline_edges <* QUERY ( edges <* eb.ces_edges | (
            'SHEET_METAL.POYLINE' IN TYPEOF(edges\edge_curve.
            edge_geometry)) ) | (NOT (SIZEOF(pline_edges\edge_curve.
            edge_geometry\polyline.points) > 2)) )) = 0)) )) = 0)) )) =
             0;
  WR5 : SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.EDGE_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
             | (NOT (SIZEOF(QUERY ( eb <* ebwm\
            edge_based_wireframe_model.ebwm_boundary | (NOT (SIZEOF(
            QUERY ( edges <* eb.ces_edges | (NOT ((
            'SHEET_METAL.VERTEX_POINT' IN TYPEOF(edges.edge_start)) AND
             ('SHEET_METAL.VERTEX_POINT' IN TYPEOF(edges.edge_end)))) ))
            = 0)) )) = 0)) )) = 0;
  WR6 : SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.EDGE_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
             | (NOT (SIZEOF(QUERY ( eb <* ebwm\
            edge_based_wireframe_model.ebwm_boundary | (NOT (SIZEOF(
            QUERY ( edges <* eb.ces_edges | (NOT
            valid_wireframe_edge_curve(edges\edge_curve.edge_geometry,
            'AIC_EDGE_BASED_WIREFRAME')) )) = 0)) )) = 0)) )) = 0;
  WR7 : SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.EDGE_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
             | (NOT (SIZEOF(QUERY ( eb <* ebwm\
            edge_based_wireframe_model.ebwm_boundary | (NOT (SIZEOF(
            QUERY ( edges <* eb.ces_edges | (NOT (
            valid_wireframe_vertex_point(edges.edge_start\vertex_point.
            vertex_geometry,'AIC_EDGE_BASED_WIREFRAME') AND
            valid_wireframe_vertex_point(edges.edge_end\vertex_point.
            vertex_geometry,'AIC_EDGE_BASED_WIREFRAME'))) )) = 0)) )) =
             0)) )) = 0;
  WR8 : SIZEOF(QUERY ( ebwm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.EDGE_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
             | (NOT (SIZEOF(QUERY ( eb <* ebwm\
            edge_based_wireframe_model.ebwm_boundary | (NOT (SIZEOF(
            QUERY ( con_edges <* QUERY ( edges <* eb.ces_edges | (
            'SHEET_METAL.CONIC' IN TYPEOF(edges\edge_curve.
            edge_geometry)) ) | (NOT ('SHEET_METAL.AXIS2_PLACEMENT_3D'
```

```
              IN TYPEOF(con_edges\edge_curve.edge_geometry\conic.position)))
               )) = 0)) )) = 0)) )) = 0;
    WR9 : SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
              'SHEET_METAL.MAPPED_ITEM' IN TYPEOF(it)) ) | (NOT ((
              'SHEET_METAL.' +
              'EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION') IN TYPEOF(mi\
              mapped_item.mapping_source.mapped_representation))) )) = 0;
    WR10: SELF.context_of_items\geometric_representation_context.
              coordinate_space_dimension = 3;
END_ENTITY; -- edge_based_wireframe_shape_representation

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

ENTITY edge_loop
  SUBTYPE OF (loop, path);
  DERIVE
    ne : INTEGER := SIZEOF(SELF\path.edge_list);
  WHERE
    WR1: SELF\path.edge_list[1].edge_start :=: SELF\path.edge_list[ne].
            edge_end;
END_ENTITY; -- edge_loop

ENTITY effectivity
  SUPERTYPE OF (ONEOF (serial_numbered_effectivity,dated_effectivity));
    id : identifier;
END_ENTITY; -- effectivity

ENTITY elementary_surface
  SUPERTYPE OF (ONEOF (plane,cylindrical_surface,conical_surface,
      spherical_surface,toroidal_surface))
  SUBTYPE OF (surface);
    position : axis2_placement_3d;
END_ENTITY; -- elementary_surface

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

ENTITY evaluated_degenerate_pcurve
  SUBTYPE OF (degenerate_pcurve);
    equivalent_point : cartesian_point;
END_ENTITY; -- evaluated_degenerate_pcurve

ENTITY executed_action
  SUBTYPE OF (action);
END_ENTITY; -- executed_action

ENTITY 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 | ('SHEET_METAL.FACE_OUTER_BOUND'
         IN TYPEOF(temp)) )) <= 1;
END_ENTITY; -- face

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

ENTITY face_outer_bound
  SUBTYPE OF (face_bound);
END_ENTITY; -- face_outer_bound

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

ENTITY faceted_brep
  SUBTYPE OF (manifold_solid_brep);
END_ENTITY; -- faceted_brep

ENTITY faceted_brep_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    WR1: SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
         'SHEET_METAL.FACETED_BREP','SHEET_METAL.MAPPED_ITEM',
         'SHEET_METAL.AXIS2_PLACEMENT_3D'] * TYPEOF(it)) = 1)) )) = 0;
    WR2: SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
         'SHEET_METAL.FACETED_BREP','SHEET_METAL.MAPPED_ITEM'] *
         TYPEOF(it)) = 1) )) > 0;
    WR3: SIZEOF(QUERY ( fbrep <* QUERY ( it <* SELF.items | (
         'SHEET_METAL.FACETED_BREP' IN TYPEOF(it)) ) | (NOT (SIZEOF(
         QUERY ( csh <* msb_shells(fbrep,'AIC_FACETED_BREP') | (NOT (
         SIZEOF(QUERY ( fcs <* csh.cfs_faces | (NOT ((
         'SHEET_METAL.FACE_SURFACE' IN TYPEOF(fcs)) AND (
         'SHEET_METAL.PLANE' IN TYPEOF(fcs\face_surface.face_geometry))
         AND ('SHEET_METAL.CARTESIAN_POINT' IN TYPEOF(fcs\
         face_surface.face_geometry\elementary_surface.position.
         location)))) )) = 0)) )) = 0)) )) = 0;
    WR4: SIZEOF(QUERY ( fbrep <* QUERY ( it <* SELF.items | (
         'SHEET_METAL.FACETED_BREP' IN TYPEOF(it)) ) | (NOT (SIZEOF(
```

```
            QUERY ( csh <* msb_shells(fbrep,'AIC_FACETED_BREP') | (NOT (
            SIZEOF(QUERY ( fcs <* csh.cfs_faces | (NOT (SIZEOF(
            QUERY ( bnds <* fcs.bounds | ('SHEET_METAL.FACE_OUTER_BOUND'
            IN TYPEOF(bnds)) )) = 1)) )) = 0)) )) = 0)) )) = 0;
    WR5: SIZEOF(QUERY ( msb <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) ) | (
            'SHEET_METAL.ORIENTED_CLOSED_SHELL' IN TYPEOF(msb\
            manifold_solid_brep.outer)) )) = 0;
    WR6: SIZEOF(QUERY ( brv <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.BREP_WITH_VOIDS' IN TYPEOF(it)) ) | (NOT (
            SIZEOF(QUERY ( csh <* brv\brep_with_voids.voids | csh\
            oriented_closed_shell.orientation )) = 0)) )) = 0;
    WR7: SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.MAPPED_ITEM' IN TYPEOF(it)) ) | (NOT (
            'SHEET_METAL.FACETED_BREP_SHAPE_REPRESENTATION' IN TYPEOF(mi
            \mapped_item.mapping_source.mapped_representation))) )) = 0;
END_ENTITY; -- faceted_brep_shape_representation

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 | (
            'SHEET_METAL.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,
      right_angular_wedge,half_space_solid,shell_based_surface_model,
      shell_based_wireframe_model,edge_based_wireframe_model,
      geometric_set))
  SUBTYPE OF (representation_item);
  DERIVE
    dim : dimension_count := dimension_of(SELF);
  WHERE
    WR1: SIZEOF(QUERY ( using_rep <* using_representations(SELF) | (NOT
            ('SHEET_METAL.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 geometric_tolerance;
    name                     : label;
    description              : text;
    magnitude                : measure_with_unit;
    toleranced_shape_aspect : shape_aspect;
  WHERE
    WR1: magnitude.value_component >= 0;
END_ENTITY; -- geometric_tolerance

ENTITY geometric_tolerance_with_datum_reference
  SUBTYPE OF (geometric_tolerance);
    datum_system : SET [1:?] OF datum_reference;
END_ENTITY; -- geometric_tolerance_with_datum_reference

ENTITY geometrically_bounded_surface_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    WR1: SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
              'SHEET_METAL.GEOMETRIC_SET','SHEET_METAL.MAPPED_ITEM',
              'SHEET_METAL.AXIS2_PLACEMENT_3D'] * TYPEOF(it)) = 1)) )) = 0;
    WR2: SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
              'SHEET_METAL.GEOMETRIC_SET','SHEET_METAL.MAPPED_ITEM'] *
              TYPEOF(it)) = 1) )) > 0;
    WR3: SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
              'SHEET_METAL.MAPPED_ITEM' IN TYPEOF(it)) ) | (NOT ((
              'SHEET_METAL.' +
              'GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION') IN
              TYPEOF(mi\mapped_item.mapping_source.mapped_representation)))
              )) = 0;
    WR4: SIZEOF(QUERY ( gs <* QUERY ( it <* SELF.items | (
              'SHEET_METAL.GEOMETRIC_SET' IN TYPEOF(it)) ) | (NOT (SIZEOF(
              QUERY ( pnt <* QUERY ( gsel <* gs\geometric_set.elements | (
              'SHEET_METAL.POINT' IN TYPEOF(gsel)) ) | (NOT
              gbsf_check_point(pnt)) )) = 0)) )) = 0;
    WR5: SIZEOF(QUERY ( gs <* QUERY ( it <* SELF.items | (
              'SHEET_METAL.GEOMETRIC_SET' IN TYPEOF(it)) ) | (NOT (SIZEOF(
              QUERY ( cv <* QUERY ( gsel <* gs\geometric_set.elements | (
              'SHEET_METAL.CURVE' IN TYPEOF(gsel)) ) | (NOT
              gbsf_check_curve(cv)) )) = 0)) )) = 0;
    WR6: SIZEOF(QUERY ( gs <* QUERY ( it <* SELF.items | (
              'SHEET_METAL.GEOMETRIC_SET' IN TYPEOF(it)) ) | (NOT (SIZEOF(
              QUERY ( sf <* QUERY ( gsel <* gs\geometric_set.elements | (
              'SHEET_METAL.SURFACE' IN TYPEOF(gsel)) ) | (NOT
              gbsf_check_surface(sf)) )) = 0)) )) = 0;
END_ENTITY; -- geometrically_bounded_surface_shape_representation

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

```
            'SHEET_METAL.GEOMETRIC_CURVE_SET','SHEET_METAL.MAPPED_ITEM'])
            = 1) )) >= 1;
    WR3: SIZEOF(QUERY ( gcs <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.GEOMETRIC_CURVE_SET' IN TYPEOF(it)) ) | (NOT (
            SIZEOF(QUERY ( crv <* QUERY ( elem <* gcs\geometric_set.
            elements | ('SHEET_METAL.CURVE' IN TYPEOF(elem)) ) | (NOT
            valid_geometrically_bounded_wf_curve(crv,
            'AIC_GEOMETRICALLY_BOUNDED_WIREFRAME')) )) = 0)) )) = 0;
    WR4: SIZEOF(QUERY ( gcs <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.GEOMETRIC_CURVE_SET' IN TYPEOF(it)) ) | (NOT (
            SIZEOF(QUERY ( pnts <* QUERY ( elem <* gcs\geometric_set.
            elements | ('SHEET_METAL.POINT' IN TYPEOF(elem)) ) | (NOT
            valid_geometrically_bounded_wf_point(pnts,
            'AIC_GEOMETRICALLY_BOUNDED_WIREFRAME')) )) = 0)) )) = 0;
    WR5: SIZEOF(QUERY ( gcs <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.GEOMETRIC_CURVE_SET' IN TYPEOF(it)) ) | (NOT (
            SIZEOF(QUERY ( cnc <* QUERY ( elem <* gcs\geometric_set.
            elements | ('SHEET_METAL.CONIC' IN TYPEOF(elem)) ) | (NOT (
            'SHEET_METAL.AXIS2_PLACEMENT_3D' IN TYPEOF(cnc\conic.
            position))) )) = 0)) )) = 0;
    WR6: SIZEOF(QUERY ( gcs <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.GEOMETRIC_CURVE_SET' IN TYPEOF(it)) ) | (NOT (
            SIZEOF(QUERY ( pline <* QUERY ( elem <* gcs\geometric_set.
            elements | ('SHEET_METAL.POLYLINE' IN TYPEOF(elem)) ) | (
            NOT (SIZEOF(pline\polyline.points) > 2)) )) = 0)) )) = 0;
    WR7: SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.MAPPED_ITEM' IN TYPEOF(it)) ) | (NOT ((
            'SHEET_METAL.' +
            'GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_REPRESENTATION') IN
            TYPEOF(mi\mapped_item.mapping_source.mapped_representation)))
            )) = 0;
    WR8: SELF.context_of_items\geometric_representation_context.
            coordinate_space_dimension = 3;
END_ENTITY; -- geometrically_bounded_wireframe_shape_representation

ENTITY global_uncertainty_assigned_context
  SUBTYPE OF (representation_context);
    uncertainty : SET [1:?] OF uncertainty_measure_with_unit;
END_ENTITY; -- global_uncertainty_assigned_context

ENTITY 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 input_item_die_relationship
  SUBTYPE OF (product_definition_relationship);
  WHERE
    WR1: product_type_of_product(SELF.relating_product_definition.
```

```
            formation.of_product,product_type).name = 'die';
    WR2: product_type_of_product(SELF.related_product_definition.
            formation.of_product,product_type).name IN ['material',
            'part'];
END_ENTITY; -- input_item_die_relationship

ENTITY intersection_curve
  SUBTYPE OF (surface_curve);
  WHERE
    WR1: SIZEOF(SELF\surface_curve.associated_geometry) = 2;
    WR2: associated_surface(SELF\surface_curve.associated_geometry[1])
            <> associated_surface(SELF\surface_curve.associated_geometry
            [2]);
END_ENTITY; -- intersection_curve

ENTITY item_defined_transformation;
    name             : label;
    description      : text;
    transform_item_1 : representation_item;
    transform_item_2 : representation_item;
END_ENTITY; -- item_defined_transformation

ENTITY length_measure_with_unit
  SUBTYPE OF (measure_with_unit);
  WHERE
    WR1: 'SHEET_METAL.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 limits_and_fits;
    form_variance : label;
    zone_variance : label;
    grade         : label;
    source        : text;
END_ENTITY; -- limits_and_fits

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

```
ENTITY local_time;
    hour_component   : hour_in_day;
    minute_component : OPTIONAL minute_in_hour;
    second_component : OPTIONAL second_in_minute;
    zone             : coordinated_universal_time_offset;
  WHERE
    WR1: valid_time(SELF);
END_ENTITY; -- local_time

ENTITY loop
  SUPERTYPE OF (ONEOF (vertex_loop,edge_loop,poly_loop))
  SUBTYPE OF (topological_representation_item);
END_ENTITY; -- loop

ENTITY make_from_usage_option
  SUBTYPE OF (product_definition_usage);
    ranking           : INTEGER;
    ranking_rationale : text;
    quantity          : measure_with_unit;
  WHERE
    WR1: ranking > 0;
END_ENTITY; -- make_from_usage_option

ENTITY make_from_usage_option_group;
  members  : SET [2:?] OF make_from_usage_option;
WHERE
  WR1: SIZEOF (QUERY (example <* members |
   example.related_product_definition
       :=: members [1].related_product_definition)) =SIZEOF(members);
END_ENTITY; -- make_from_usage_option_group

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

ENTITY manifold_surface_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    WR1 : SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
              'SHEET_METAL.SHELL_BASED_SURFACE_MODEL',
              'SHEET_METAL.MAPPED_ITEM','SHEET_METAL.AXIS2_PLACEMENT_3D']
              * TYPEOF(it)) = 1)) )) = 0;
    WR2 : SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
              'SHEET_METAL.SHELL_BASED_SURFACE_MODEL',
              'SHEET_METAL.MAPPED_ITEM'] * TYPEOF(it)) = 1) )) > 0;
    WR3 : SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
              'SHEET_METAL.MAPPED_ITEM' IN TYPEOF(it)) ) | (NOT (
              'SHEET_METAL.MANIFOLD_SURFACE_SHAPE_REPRESENTATION' IN
             TYPEOF(mi\mapped_item.mapping_source.mapped_representation))) ))
              = 0;
    WR4 : SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
              'SHEET_METAL.SHELL_BASED_SURFACE_MODEL' IN TYPEOF(it)) ) |
              (NOT (SIZEOF(QUERY ( sh <* sbsm\shell_based_surface_model.
              sbsm_boundary | (NOT (SIZEOF(['SHEET_METAL.OPEN_SHELL',
              'SHEET_METAL.CLOSED_SHELL'] * TYPEOF(sh)) = 1)) )) = 0)) ))
```

```
            = 0;
WR5 : SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.SHELL_BASED_SURFACE_MODEL' IN TYPEOF(it)) ) |
            (NOT (SIZEOF(QUERY ( cfs <* sbsm\shell_based_surface_model.
            sbsm_boundary | (NOT (SIZEOF(QUERY ( fa <* cfs\
            connected_face_set.cfs_faces | (NOT (SIZEOF([
            'SHEET_METAL.FACE_SURFACE','SHEET_METAL.ORIENTED_FACE'] *
            TYPEOF(fa)) = 1)) )) = 0)) )) = 0)) )) = 0;
WR6 : SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.SHELL_BASED_SURFACE_MODEL' IN TYPEOF(it)) ) |
            (NOT (SIZEOF(QUERY ( cfs <* sbsm\shell_based_surface_model.
            sbsm_boundary | (NOT (SIZEOF(QUERY ( f_sf <* QUERY ( fa <*
            cfs\connected_face_set.cfs_faces | (
            'SHEET_METAL.FACE_SURFACE' IN TYPEOF(fa)) ) | (NOT ((
            'SHEET_METAL.ADVANCED_FACE' IN TYPEOF(f_sf)) OR (SIZEOF([
            'SHEET_METAL.OFFSET_SURFACE','SHEET_METAL.SURFACE_REPLICA']
            * TYPEOF(f_sf\face_surface.face_geometry)) = 1))) )) = 0)) ))
            = 0)) )) = 0;
WR7 : SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.SHELL_BASED_SURFACE_MODEL' IN TYPEOF(it)) ) |
            (NOT (SIZEOF(QUERY ( cfs <* sbsm\shell_based_surface_model.
            sbsm_boundary | (NOT (SIZEOF(QUERY ( fa <* cfs\
            connected_face_set.cfs_faces | (NOT ((
            'SHEET_METAL.ADVANCED_FACE' IN TYPEOF(fa)) OR
            basis_surface_check(fa\face_surface.face_geometry))) ) =
            0)))) = 0)) )) = 0;
WR8 : SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.SHELL_BASED_SURFACE_MODEL' IN TYPEOF(it)) ) |
            (NOT (SIZEOF(QUERY ( cfs <* sbsm\shell_based_surface_model.
            sbsm_boundary | (NOT (SIZEOF(QUERY ( fa <* cfs\
            connected_face_set.cfs_faces | (NOT ((
            'SHEET_METAL.ADVANCED_FACE' IN TYPEOF(fa)) OR (SIZEOF(
            QUERY ( bnds <* fa.bounds | (NOT (SIZEOF([
            'SHEET_METAL.EDGE_LOOP','SHEET_METAL.VERTEX_LOOP'] *
            TYPEOF(bnds.bound)) = 1)) )) = 0))) )) = 0)) )) = 0)) )) = 0;
WR9 : SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.SHELL_BASED_SURFACE_MODEL' IN TYPEOF(it)) ) |
            (NOT (SIZEOF(QUERY ( cfs <* sbsm\shell_based_surface_model.
            sbsm_boundary | (NOT (SIZEOF(QUERY ( fa <* cfs\
            connected_face_set.cfs_faces | (NOT ((
            'SHEET_METAL.ADVANCED_FACE' IN TYPEOF(fa)) OR (SIZEOF(
            QUERY ( elp_fbnds <* QUERY ( bnds <* fa.bounds | (
            'SHEET_METAL.EDGE_LOOP' IN TYPEOF(bnds)) ) | (NOT (SIZEOF(
            QUERY ( oe <* elp_fbnds.bound\path.edge_list | (NOT (
            'SHEET_METAL.EDGE_CURVE' IN TYPEOF(oe.edge_element))) ) =
            0)) )) = 0))) )) = 0)) )) = 0)) )) = 0;
WR10: SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.SHELL_BASED_SURFACE_MODEL' IN TYPEOF(it)) ) |
            (NOT (SIZEOF(QUERY ( cfs <* sbsm\shell_based_surface_model.
            sbsm_boundary | (NOT (SIZEOF(QUERY ( fa <* cfs\
            connected_face_set.cfs_faces | (NOT ((
            'SHEET_METAL.ADVANCED_FACE' IN TYPEOF(fa)) OR (SIZEOF(
            QUERY ( elp_fbnds <* QUERY ( bnds <* fa.bounds | (
            'SHEET_METAL.EDGE LOOP' IN TYPEOF(bnds.bound)) ) | (NOT (
            SIZEOF(QUERY ( oe_cv <* QUERY ( oe <* elp_fbnds.bound\path.
            edge_list | ('SHEET_METAL.EDGE_CURVE' IN TYPEOF(oe.
```

```
            edge_element)) ) | (NOT (SIZEOF([
            'SHEET_METAL.CURVE_REPLICA','SHEET_METAL.OFFSET_CURVE_3D',
            'SHEET_METAL.PCURVE'] * TYPEOF(oe_cv.edge_element\
            edge_curve.edge_geometry)) = 1)) )) = 0)) )) = 0))) )) =
            0)))) = 0)) )) = 0;
    WR11: SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.SHELL_BASED_SURFACE_MODEL' IN TYPEOF(it)) ) |
            (NOT (SIZEOF(QUERY ( cfs <* sbsm\shell_based_surface_model.
            sbsm_boundary | (NOT (SIZEOF(QUERY ( fa <* cfs\
            connected_face_set.cfs_faces | (NOT ((
            'SHEET_METAL.ADVANCED_FACE' IN TYPEOF(fa)) OR (SIZEOF(
            QUERY ( elp_fbnds <* QUERY ( bnds <* fa.bounds | (
            'SHEET_METAL.EDGE_LOOP' IN TYPEOF(bnds.bound)) ) | (NOT (
            SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.edge_list | (NOT
            basis_curve_check(oe.edge_element\edge_curve.edge_geometry))))
            = 0)) )) = 0))) )) = 0)) )) = 0)) )) = 0;
    WR12: SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.SHELL_BASED_SURFACE_MODEL' IN TYPEOF(it)) ) |
            (NOT (SIZEOF(QUERY ( cfs <* sbsm\shell_based_surface_model.
            sbsm_boundary | (NOT (SIZEOF(QUERY ( fa <* cfs\
            connected_face_set.cfs_faces | (NOT ((
            'SHEET_METAL.ADVANCED_FACE' IN TYPEOF(fa)) OR (SIZEOF(
            QUERY ( elp_fbnds <* QUERY ( bnds <* fa.bounds | (
            'SHEET_METAL.EDGE_LOOP' IN TYPEOF(bnds)) ) | (NOT (SIZEOF(
            QUERY ( oe <* elp_fbnds.bound\path.edge_list | (NOT ((
            'SHEET_METAL.VERTEX_POINT' IN TYPEOF(oe.edge_element.
            edge_start)) AND ('SHEET_METAL.VERTEX_POINT' IN TYPEOF(oe.
            edge_element.edge_end))))) = 0)) )) = 0))) )) = 0)) )) =
            0)) )) = 0;
    WR13: SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.SHELL_BASED_SURFACE_MODEL' IN TYPEOF(it)) ) |
            (NOT (SIZEOF(QUERY ( cfs <* sbsm\shell_based_surface_model.
            sbsm_boundary | (NOT (SIZEOF(QUERY ( fa <* cfs\
            connected_face_set.cfs_faces | (NOT ((
            'SHEET_METAL.ADVANCED_FACE' IN TYPEOF(fa)) OR (SIZEOF(
            QUERY ( elp_fbnds <* QUERY ( bnds <* fa.bounds | (
            'SHEET_METAL.EDGE_LOOP' IN TYPEOF(bnds.bound)) ) | (NOT (
            SIZEOF(QUERY ( oe <* elp_fbnds.bound\path.edge_list | (NOT
            ((SIZEOF(['SHEET_METAL.CARTESIAN_POINT',
            'SHEET_METAL.DEGENERATE_PCURVE',
            'SHEET_METAL.POINT_ON_CURVE','SHEET_METAL.POINT_ON_SURFACE']
            * TYPEOF(oe.edge_element.edge_start\vertex_point.
            vertex_geometry)) = 1) AND (SIZEOF([
            'SHEET_METAL.CARTESIAN_POINT',
            'SHEET_METAL.DEGENERATE_PCURVE',
            'SHEET_METAL.POINT_ON_CURVE','SHEET_METAL.POINT_ON_SURFACE']
            * TYPEOF(oe.edge_element.edge_end\vertex_point.
            vertex_geometry)) = 1))) )) = 0)) )) = 0))) )) = 0)) )) =
            0)))) = 0;
    WR14: SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.SHELL_BASED_SURFACE_MODEL' IN TYPEOF(it)) ) |
            (NOT (SIZEOF(QUERY ( cfs <* sbsm\shell_based_surface_model.
            sbsm_boundary | (NOT (SIZEOF(QUERY ( fa <* cfs\
            connected_face_set.cfs_faces | (NOT ((
            'SHEET_METAL.ADVANCED_FACE' IN TYPEOF(fa)) OR (SIZEOF(
            QUERY ( vlp_fbnds <* QUERY ( bnds <* fa.bounds | (
```

```
                'SHEET_METAL.VERTEX_LOOP' IN TYPEOF(bnds.bound)) ) | (NOT (
                'SHEET_METAL.VERTEX_POINT' IN TYPEOF(vlp_fbnds.bound\
                vertex_loop.loop_vertex))) )) = 0))) )) = 0)) )) = 0)) )) =
                 0;
    WR15: SIZEOF(QUERY ( sbsm <* QUERY ( it <* SELF.items | (
                'SHEET_METAL.SHELL_BASED_SURFACE_MODEL' IN TYPEOF(it)) ) |
                (NOT (SIZEOF(QUERY ( cfs <* sbsm\shell_based_surface_model.
                sbsm_boundary | (NOT (SIZEOF(QUERY ( fa <* cfs\
                connected_face_set.cfs_faces | (NOT ((
                'SHEET_METAL.ADVANCED_FACE' IN TYPEOF(fa)) OR (SIZEOF(
                QUERY ( vlp_fbnds <* QUERY ( bnds <* fa.bounds | (
                'SHEET_METAL.VERTEX_LOOP' IN TYPEOF(bnds.bound)) ) | (NOT (
                SIZEOF(['SHEET_METAL.CARTESIAN_POINT',
                'SHEET_METAL.DEGENERATE_PCURVE',
                'SHEET_METAL.POINT_ON_CURVE','SHEET_METAL.POINT_ON_SURFACE']
                * TYPEOF(vlp_fbnds.bound\vertex_loop.loop_vertex\
                vertex_point.vertex_geometry)) = 1)) )) = 0))) )) = 0)) ))
                = 0)) )) = 0;
END_ENTITY; -- manifold_surface_shape_representation

ENTITY mapped_item
  SUBTYPE OF (representation_item);
    mapping_source : representation_map;
    mapping_target : representation_item;
  WHERE
    WR1: acyclic_mapped_representation(using_representations(SELF),[SELF]);
END_ENTITY; -- mapped_item

ENTITY material_property
  SUBTYPE OF (property_definition);
  UNIQUE
    ur1 : name, definition;
  WHERE
    WR1: ('SHEET_METAL.CHARACTERIZED_OBJECT' IN (TYPEOF(SELF\
              property_definition.definition) OR SIZEOF(bag_to_set(USEDIN(
              SELF,'SHEET_METAL.' +
              'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) -
              QUERY ( temp <* bag_to_set(USEDIN(SELF,
              'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.' +
              'PROPERTY_DEFINITION_REPRESENTATION.DEFINITION')) | ((
              'MATERIAL_PROPERTY_REPRESENTATION_SCHEMA.' +
              'MATERIAL_PROPERTY_REPRESENTATION') IN TYPEOF(temp)) )))) =
              0;
END_ENTITY; -- material_property

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 modified_geometric_tolerance
  SUBTYPE OF (geometric_tolerance);
    modifier : limit_condition;
END_ENTITY; -- modified_geometric_tolerance

ENTITY named_unit
  SUPERTYPE OF (ONEOF (si_unit,conversion_based_unit,
      context_dependent_unit) ANDOR ONEOF (length_unit,plane_angle_unit));
    dimensions : dimensional_exponents;
END_ENTITY; -- named_unit

ENTITY offset_curve_3d
  SUBTYPE OF (curve);
    basis_curve    : curve;
    distance       : length_measure;
    self_intersect : LOGICAL;
    ref_direction  : direction;
  WHERE
    WR1: (basis_curve.dim = 3) AND (ref_direction.dim = 3);
END_ENTITY; -- offset_curve_3d

ENTITY offset_surface
  SUBTYPE OF (surface);
    basis_surface  : surface;
    distance       : length_measure;
    self_intersect : LOGICAL;
END_ENTITY; -- offset_surface

ENTITY open_shell
  SUBTYPE OF (connected_face_set);
END_ENTITY; -- open_shell

ENTITY ordinal_date
  SUBTYPE OF (date);
    day_component : day_in_year_number;
  WHERE
    WR1: ((NOT leap_year(SELF.year_component)) AND (1 <= day_component)
              AND (day_component <= 365)) OR (leap_year(SELF.
              year_component) AND (1 <= day_component) AND (day_component
              <= 366));
END_ENTITY; -- ordinal_date

ENTITY organization;
    id          : OPTIONAL identifier;
    name        : label;
    description : text;
END_ENTITY; -- organization

ENTITY organization_assignment
  ABSTRACT SUPERTYPE;
    assigned_organization : organization;
    role                  : organization_role;
END_ENTITY; -- organization_assignment
```

```
ENTITY organization_role;
    name : label;
END_ENTITY; -- organization_role

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 ('SHEET_METAL.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 ('SHEET_METAL.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 ('SHEET_METAL.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 ('SHEET_METAL.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 ('SHEET_METAL.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: 'SHEET_METAL.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 physically_modelled_product_definition
  SUBTYPE OF (product_definition);
  WHERE
    WR1: SELF\product_definition.frame_of_reference.life_cycle_stage =
              'as physically modelled';
    WR2: SIZEOF(USEDIN(SELF,'SHEET_METAL.' +
              'PRODUCT_DEFINITION_RELATIONSHIP.' +
              'RELATED_PRODUCT_DEFINITION')) > 0;
END_ENTITY; -- physically_modelled_product_definition

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: 'SHEET_METAL.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 plus_minus_tolerance;
    range                : tolerance_method_definition;
    toleranced_dimension : dimensional_characteristic;
  UNIQUE
    ur1 : toleranced_dimension;
END_ENTITY; -- plus_minus_tolerance

ENTITY point
  SUPERTYPE OF (ONEOF (cartesian_point,point_on_curve,point_on_surface,
      point_replica,degenerate_pcurve))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- point

ENTITY point_on_curve
  SUBTYPE OF (point);
    basis_curve     : curve;
    point_parameter : parameter_value;
END_ENTITY; -- point_on_curve

ENTITY point_on_surface
  SUBTYPE OF (point);
    basis_surface     : surface;
    point_parameter_u : parameter_value;
    point_parameter_v : parameter_value;
END_ENTITY; -- point_on_surface

ENTITY point_replica
  SUBTYPE OF (point);
    parent_pt      : point;
    transformation : cartesian_transformation_operator;
  WHERE
    WR1: transformation.dim = parent_pt.dim;
    WR2: acyclic_point_replica(SELF,parent_pt);
END_ENTITY; -- point_replica

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

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

ENTITY process_plan
  SUBTYPE OF (action);
END_ENTITY; -- process_plan

ENTITY process_product_association;
    name            : label;
    description     : text;
    process         : product_definition_process;
    defined_product : characterized_product_definition;
END_ENTITY; -- process_product_association

ENTITY product;
    id                  : identifier;
    name                : label;
    description         : text;
    frame_of_reference : SET [1:?] OF product_context;
  UNIQUE
    ur1 : id;
END_ENTITY; -- product

ENTITY product_category;
    name        : label;
    description : OPTIONAL text;
END_ENTITY; -- product_category

ENTITY product_category_relationship;
    name         : label;
    description  : text;
    category     : product_category;
    sub_category : product_category;
  WHERE
    WR1: acyclic_product_category_relationship(SELF,[SELF.sub_category]);
END_ENTITY; -- product_category_relationship

ENTITY product_concept;
    id             : identifier;
    name           : label;
    description    : text;
    market_context : product_concept_context;
  UNIQUE
    ur1 : id;
END_ENTITY; -- product_concept

ENTITY product_concept_context
  SUBTYPE OF (application_context_element);
    market_segment_type : label;
END_ENTITY; -- product_concept_context

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

```
END_ENTITY; -- product_context

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

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

ENTITY product_definition_effectivity
  SUBTYPE OF (effectivity);
    usage : product_definition_relationship;
  UNIQUE
    ur1 : usage, id;
END_ENTITY; -- product_definition_effectivity

ENTITY product_definition_formation;
    id          : identifier;
    description : text;
    of_product  : product;
  UNIQUE
    ur1 : id, of_product;
END_ENTITY; -- product_definition_formation

ENTITY product_definition_formation_with_specified_source
  SUBTYPE OF (product_definition_formation);
    make_or_buy : source;
END_ENTITY; -- product_definition_formation_with_specified_source

ENTITY product_definition_process
  SUBTYPE OF (action);
    identification : identifier;
  INVERSE
    product_definitions : SET [1:?] OF process_product_association FOR
                                 process;
END_ENTITY; -- product_definition_process

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: 'SHEET_METAL.CHARACTERIZED_PRODUCT_DEFINITION' IN TYPEOF(SELF\
```

```
            property_definition.definition);
END_ENTITY; -- product_definition_shape

ENTITY product_definition_substitute;
    description           : text;
    context_relationship  : product_definition_relationship;
    substitute_definition : product_definition;
  WHERE
    WR1: context_relationship.related_product_definition :<>:
            substitute_definition;
END_ENTITY; -- product_definition_substitute

ENTITY product_definition_usage
  SUPERTYPE OF (ONEOF (make_from_usage_option,assembly_component_usage))
  SUBTYPE OF (product_definition_relationship);
  UNIQUE
    ur1 : id, relating_product_definition, related_product_definition;
  WHERE
    WR1: acyclic_product_definition_relationship(SELF,[SELF\
            product_definition_relationship.related_product_definition],
            'SHEET_METAL.PRODUCT_DEFINITION_USAGE.' +
            'RELATED_PRODUCT_DEFINITION');
END_ENTITY; -- product_definition_usage

ENTITY product_definition_with_associated_documents
  SUBTYPE OF (product_definition);
    documentation_ids : SET [1:?] OF document;
END_ENTITY; -- product_definition_with_associated_documents

ENTITY product_related_product_category
  SUBTYPE OF (product_category);
    products : SET [1:?] OF product;
END_ENTITY; -- product_related_product_category

ENTITY product_type
  SUBTYPE OF (product_related_product_category);
  WHERE
    WR1: SIZEOF(USEDIN(SELF,'SHEET_METAL.PRODUCT_CATEGORY_RELATIONSHIP.'
            + 'RELATING_PRODUCT_CATEGORY') + USEDIN(SELF,
            'SHEET_METAL.PRODUCT_CATEGORY_RELATIONSHIP.' +
            'RELATED_PRODUCT_CATEGORY')) = 0;
    WR2: SELF.name IN ['die','part','material','unspecified'];
END_ENTITY; -- product_type

ENTITY promissory_usage_occurrence
  SUBTYPE OF (assembly_component_usage);
END_ENTITY; -- promissory_usage_occurrence

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

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

ENTITY quantified_assembly_component_usage
  SUBTYPE OF (assembly_component_usage);
    quantity : measure_with_unit;
END_ENTITY; -- quantified_assembly_component_usage

ENTITY quasi_uniform_curve
  SUBTYPE OF (b_spline_curve);
END_ENTITY; -- quasi_uniform_curve

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

ENTITY 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 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: (('SHEET_METAL.ELEMENTARY_SURFACE' IN TYPEOF(basis_surface))
               AND (NOT ('SHEET_METAL.PLANE' IN TYPEOF(basis_surface)))) OR
                ('SHEET_METAL.SURFACE_OF_REVOLUTION' IN TYPEOF(
               basis_surface)) OR (usense = (u2 > u1));
      WR4: ('SHEET_METAL.SPHERICAL_SURFACE' IN TYPEOF(basis_surface)) OR (
               'SHEET_METAL.TOROIDAL_SURFACE' IN TYPEOF(basis_surface)) OR
               (vsense = (v2 > v1));
  END_ENTITY; -- rectangular_trimmed_surface

  ENTITY reparametrised_composite_curve_segment
    SUBTYPE OF (composite_curve_segment);
      param_length : parameter_value;
    WHERE
      WR1: param_length > 0;
  END_ENTITY; -- reparametrised_composite_curve_segment

  ENTITY replacement_relationship
    SUBTYPE OF (action_relationship);
    WHERE
      WR1: acyclic_action_relationship(SELF,[SELF\action_relationship.
               related_action],'SHEET_METAL.REPLACEMENT_RELATIONSHIP');
  END_ENTITY; -- replacement_relationship

  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 requirement_for_action_resource
  SUBTYPE OF (action_resource_requirement);
    resources : SET [1:?] OF action_resource;
END_ENTITY; -- requirement_for_action_resource

ENTITY resource_property;
    name        : label;
    description : text;
    resource    : characterized_resource_definition;
END_ENTITY; -- resource_property

ENTITY resource_property_representation;
    name           : label;
    description    : text;
    property       : resource_property;
    representation : representation;
END_ENTITY; -- resource_property_representation

ENTITY resource_requirement_type;
    name        : label;
    description : text;
END_ENTITY; -- resource_requirement_type

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_angular_wedge
  SUBTYPE OF (geometric_representation_item);
```

```
    position : axis2_placement_3d;
    x        : positive_length_measure;
    y        : positive_length_measure;
    z        : positive_length_measure;
    ltx      : length_measure;
  WHERE
    WR1: (0 <= ltx) AND (ltx < x);
END_ENTITY; -- right_angular_wedge

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 seam_curve
  SUBTYPE OF (surface_curve);
  WHERE
    WR1: SIZEOF(SELF\surface_curve.associated_geometry) = 2;
    WR2: associated_surface(SELF\surface_curve.associated_geometry[1]) =
              associated_surface(SELF\surface_curve.associated_geometry[2]);
    WR3: 'SHEET_METAL.PCURVE' IN TYPEOF(SELF\surface_curve.
             associated_geometry[1]);
    WR4: 'SHEET_METAL.PCURVE' IN TYPEOF(SELF\surface_curve.
             associated_geometry[2]);
END_ENTITY; -- seam_curve

ENTITY security_classification;
    name           : label;
    purpose        : text;
    security_level : security_classification_level;
END_ENTITY; -- security_classification

ENTITY security_classification_assignment
  ABSTRACT SUPERTYPE;
    assigned_security_classification : security_classification;
END_ENTITY; -- security_classification_assignment

ENTITY security_classification_level;
    name : label;
END_ENTITY; -- security_classification_level

ENTITY sequenced_product_definition_relationship
  SUBTYPE OF (product_definition_relationship);
  WHERE
```

```
    WR1: (('SHEET_METAL.' +
             'PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS') IN (TYPEOF(
             SELF\product_definition_relationship.
             relating_product_definition) AND ('SHEET_METAL.' +
             'PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS'))) IN TYPEOF(
             SELF\product_definition_relationship.
             related_product_definition);
END_ENTITY; -- sequenced_product_definition_relationship

ENTITY sequential_method
  SUBTYPE OF (serial_action_method);
    sequence_position : count_measure;
END_ENTITY; -- sequential_method

ENTITY serial_action_method
  SUBTYPE OF (action_method_relationship);
END_ENTITY; -- serial_action_method

ENTITY serial_numbered_effectivity
  SUBTYPE OF (effectivity);
    effectivity_start_id : identifier;
    effectivity_end_id   : OPTIONAL identifier;
END_ENTITY; -- serial_numbered_effectivity

ENTITY shape_aspect;
    name                 : label;
    description          : text;
    of_shape             : product_definition_shape;
    product_definitional : LOGICAL;
END_ENTITY; -- shape_aspect

ENTITY shape_aspect_relationship;
    name                   : label;
    description            : text;
    relating_shape_aspect  : shape_aspect;
    related_shape_aspect   : shape_aspect;
END_ENTITY; -- shape_aspect_relationship

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

ENTITY shape_dimension_representation
  SUBTYPE OF (shape_representation);
  WHERE
    WR1: SIZEOF(QUERY ( temp <* SELF.items | ((NOT
             'SHEET_METAL.MEASURE_REPRESENTATION_ITEM') IN TYPEOF(temp)) ))
             = 0;
```

```
    WR2: SIZEOF(SELF.items) <= 2;
    WR3: SIZEOF(QUERY ( pos_mri <* QUERY ( real_mri <* SELF.items | (
             'REAL' IN TYPEOF(real_mri\measure_with_unit.value_component)) )
              | (NOT (pos_mri\measure_with_unit.value_component > 0)) ))
             = 0;
END_ENTITY; -- shape_dimension_representation

ENTITY shape_representation
  SUBTYPE OF (representation);
END_ENTITY; -- shape_representation

ENTITY shape_representation_relationship
  SUBTYPE OF (representation_relationship);
  WHERE
    WR1: 'SHEET_METAL.SHAPE_REPRESENTATION' IN (TYPEOF(SELF\
             representation_relationship.rep_1) + TYPEOF(SELF\
             representation_relationship.rep_2));
END_ENTITY; -- shape_representation_relationship

ENTITY sheet_metal_action_assignment
  SUBTYPE OF (action_assignment);
    items : SET [1:?] OF action_assigned_item;
  WHERE
    WR1: 'SHEET_METAL.EXECUTED_ACTION' IN TYPEOF(SELF\action_assignment.
             assigned_action);
END_ENTITY; -- sheet_metal_action_assignment

ENTITY sheet_metal_approval_assignment
  SUBTYPE OF (approval_assignment);
    items : SET [1:?] OF approval_assigned_item;
END_ENTITY; -- sheet_metal_approval_assignment

ENTITY sheet_metal_contract_assignment
  SUBTYPE OF (contract_assignment);
    items : SET [1:?] OF contract_assigned_item;
END_ENTITY; -- sheet_metal_contract_assignment

ENTITY sheet_metal_date_and_time_assignment
  SUBTYPE OF (date_and_time_assignment);
    items : SET [1:?] OF date_and_time_assigned_item;
  WHERE
    WR1: sheet_metal_date_correlation(SELF);
END_ENTITY; -- sheet_metal_date_and_time_assignment

ENTITY sheet_metal_date_assignment
  SUBTYPE OF (date_assignment);
    items : SET [1:?] OF date_assigned_item;
  WHERE
    WR1: sheet_metal_date_correlation(SELF);
END_ENTITY; -- sheet_metal_date_assignment

ENTITY sheet_metal_document_reference
  SUBTYPE OF (document_reference);
    items : SET [1:?] OF document_referenced_item;
  WHERE
    WR1: sheet_metal_document_correlation(SELF);
```

```
END_ENTITY; -- sheet_metal_document_reference

ENTITY sheet_metal_organization_assignment
  SUBTYPE OF (organization_assignment);
    items : SET [1:?] OF organization_assigned_item;
  WHERE
    WR1: sheet_metal_organization_correlation(SELF);
END_ENTITY; -- sheet_metal_organization_assignment

ENTITY sheet_metal_person_and_organization_assignment
  SUBTYPE OF (person_and_organization_assignment);
    items : SET [1:?] OF person_and_organization_assigned_item;
  WHERE
    WR1: sheet_metal_person_and_organization_correlation(SELF);
END_ENTITY; -- sheet_metal_person_and_organization_assignment

ENTITY sheet_metal_person_assignment
  SUBTYPE OF (person_assignment);
    items : SET [1:?] OF person_assigned_item;
  WHERE
    WR1: sheet_metal_person_correlation(SELF);
END_ENTITY; -- sheet_metal_person_assignment

ENTITY sheet_metal_security_classification_assignment
  SUBTYPE OF (security_classification_assignment);
    items : SET [1:?] OF security_classification_assigned_item;
END_ENTITY; -- sheet_metal_security_classification_assignment


ENTITY shell_based_surface_model
  SUBTYPE OF (geometric_representation_item);
    sbsm_boundary : SET [1:?] OF shell;
  WHERE
    WR1: constraints_geometry_shell_based_surface_model(SELF);
END_ENTITY; -- shell_based_surface_model

ENTITY shell_based_wireframe_model
  SUBTYPE OF (geometric_representation_item);
    sbwm_boundary : SET [1:?] OF shell;
  WHERE
    WR1: constraints_geometry_shell_based_wireframe_model(SELF);
END_ENTITY; -- shell_based_wireframe_model

ENTITY shell_based_wireframe_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
    WR1 : SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
              'SHEET_METAL.SHELL_BASED_WIREFRAME_MODEL',
              'SHEET_METAL.MAPPED_ITEM','SHEET_METAL.AXIS2_PLACEMENT_3D']
              * TYPEOF(it)) = 1)) )) = 0;
    WR2 : SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
              'SHEET_METAL.SHELL_BASED_WIREFRAME_MODEL',
              'SHEET_METAL.MAPPED_ITEM'] * TYPEOF(it)) = 1) )) >= 1;
    WR3 : SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
              'SHEET_METAL.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
               | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <* sbwm\
```

```
            shell_based_wireframe_model.sbwm_boundary | (
            'SHEET_METAL.WIRE_SHELL' IN TYPEOF(sb)) ) | (NOT (SIZEOF(
            QUERY ( eloop <* QUERY ( wsb <* ws\wire_shell.
            wire_shell_extent | ('SHEET_METAL.EDGE_LOOP' IN TYPEOF(wsb)) )
             | (NOT (SIZEOF(QUERY ( el <* eloop\path.edge_list | (NOT (
            'SHEET_METAL.EDGE_CURVE' IN TYPEOF(el.edge_element))) )) =
            0)) )) = 0)) )) = 0)) )) = 0;
    WR4 : SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
             | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <* sbwm\
            shell_based_wireframe_model.sbwm_boundary | (
            'SHEET_METAL.WIRE_SHELL' IN TYPEOF(sb)) ) | (NOT (SIZEOF(
            QUERY ( eloop <* QUERY ( wsb <* ws\wire_shell.
            wire_shell_extent | ('SHEET_METAL.EDGE_LOOP' IN TYPEOF(wsb)) )
             | (NOT (SIZEOF(QUERY ( pline_el <* QUERY ( el <* eloop\
            path.edge_list | ('SHEET_METAL.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 | (
            'SHEET_METAL.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
             | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <* sbwm\
            shell_based_wireframe_model.sbwm_boundary | (
            'SHEET_METAL.WIRE_SHELL' IN TYPEOF(sb)) ) | (NOT (SIZEOF(
            QUERY ( eloop <* QUERY ( wsb <* ws\wire_shell.
            wire_shell_extent | ('SHEET_METAL.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,'AIC_SHELL_BASED_WIREFRAME')) )) = 0)) )) =
            0)))) = 0)) )) = 0;
    WR6 : SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
             | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <* sbwm\
            shell_based_wireframe_model.sbwm_boundary | (
            'SHEET_METAL.WIRE_SHELL' IN TYPEOF(sb)) ) | (NOT (SIZEOF(
            QUERY ( eloop <* QUERY ( wsb <* ws\wire_shell.
            wire_shell_extent | ('SHEET_METAL.EDGE_LOOP' IN TYPEOF(wsb)) )
             | (NOT (SIZEOF(QUERY ( el <* eloop\path.edge_list | (NOT (
            ('SHEET_METAL.VERTEX_POINT' IN TYPEOF(el.edge_element.
            edge_start)) AND ('SHEET_METAL.VERTEX_POINT' IN TYPEOF(el.
            edge_element.edge_end)))) )) = 0)) )) = 0)) )) = 0)) )) = 0;
    WR7 : SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
             | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <* sbwm\
            shell_based_wireframe_model.sbwm_boundary | (
            'SHEET_METAL.WIRE_SHELL' IN TYPEOF(sb)) ) | (NOT (SIZEOF(
            QUERY ( eloop <* QUERY ( wsb <* ws\wire_shell.
            wire_shell_extent | ('SHEET_METAL.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,'AIC_SHELL_BASED_WIREFRAME')
            AND valid_wireframe_vertex_point(el.edge_element.edge_end\
            vertex_point.vertex_geometry,'AIC_SHELL_BASED_WIREFRAME'))) ))
            = 0)) )) = 0)) )) = 0)) )) = 0;
    WR8 : SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
            'SHEET_METAL.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
```

```
                 | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <* sbwm\
                shell_based_wireframe_model.sbwm_boundary | (
                'SHEET_METAL.WIRE_SHELL' IN TYPEOF(sb)) ) | (NOT (SIZEOF(
                QUERY ( eloop <* QUERY ( wsb <* ws\wire_shell.
                wire_shell_extent | ('SHEET_METAL.EDGE_LOOP' IN TYPEOF(wsb)) )
                 | (NOT (SIZEOF(QUERY ( con_edges <* QUERY ( el <* eloop\
                path.edge_list | ('SHEET_METAL.CONIC' IN TYPEOF(el.
                edge_element\edge_curve.edge_geometry)) ) | (NOT (
                'SHEET_METAL.AXIS2_PLACEMENT_3D' IN TYPEOF(con_edges.
                edge_element\edge_curve.edge_geometry\conic.position)) ) )
                = 0)) )) = 0)) )) = 0)) )) = 0;
    WR9 : SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
                'SHEET_METAL.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
                 | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <* sbwm\
                shell_based_wireframe_model.sbwm_boundary | (
                'SHEET_METAL.WIRE_SHELL' IN TYPEOF(sb)) ) | (NOT (SIZEOF(
                QUERY ( vloop <* QUERY ( wsb <* ws\wire_shell.
                wire_shell_extent | ('SHEET_METAL.VERTEX_LOOP' IN TYPEOF(
                wsb)) ) | (NOT ('SHEET_METAL.VERTEX_POINT' IN TYPEOF(vloop\
                vertex_loop.loop_vertex))) )) = 0)) )) = 0)) )) = 0;
    WR10: SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
                'SHEET_METAL.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
                 | (NOT (SIZEOF(QUERY ( ws <* QUERY ( sb <* sbwm\
                shell_based_wireframe_model.sbwm_boundary | (
                'SHEET_METAL.WIRE_SHELL' IN TYPEOF(sb)) ) | (NOT (SIZEOF(
                QUERY ( vloop <* QUERY ( wsb <* ws\wire_shell.
                wire_shell_extent | ('SHEET_METAL.VERTEX_LOOP' IN TYPEOF(
                wsb)) ) | (NOT valid_wireframe_vertex_point(vloop\
                vertex_loop.loop_vertex\vertex_point.vertex_geometry,
                'AIC_SHELL_BASED_WIREFRAME')) )) = 0)) )) = 0)) )) = 0;
    WR11: SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
                'SHEET_METAL.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
                 | (NOT (SIZEOF(QUERY ( vs <* QUERY ( sb <* sbwm\
                shell_based_wireframe_model.sbwm_boundary | (
                'SHEET_METAL.VERTEX_SHELL' IN TYPEOF(sb)) ) | (NOT (
                'SHEET_METAL.VERTEX_POINT' IN TYPEOF(vs\vertex_shell.
                vertex_shell_extent.loop_vertex))) )) = 0)) )) = 0;
    WR12: SIZEOF(QUERY ( sbwm <* QUERY ( it <* SELF.items | (
                'SHEET_METAL.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF(it)) )
                 | (NOT (SIZEOF(QUERY ( vs <* QUERY ( sb <* sbwm\
                shell_based_wireframe_model.sbwm_boundary | (
                'SHEET_METAL.VERTEX_SHELL' IN TYPEOF(sb)) ) | (NOT
                valid_wireframe_vertex_point(vs\vertex_shell.
                vertex_shell_extent.loop_vertex\vertex_point.
                vertex_geometry,'AIC_SHELL_BASED_WIREFRAME')) )) = 0)) )) =
                 0;
    WR13: SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items | (
                'SHEET_METAL.MAPPED_ITEM' IN TYPEOF(it)) ) | (NOT ((
                'SHEET_METAL.' +
                'SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION') IN TYPEOF(mi\
                mapped_item.mapping_source.mapped_representation))) )) = 0;
    WR14: SELF.context_of_items\geometric_representation_context.
                coordinate_space_dimension = 3;
END_ENTITY; -- shell_based_wireframe_shape_representation

ENTITY si_unit
```

```
  SUBTYPE OF (named_unit);
    prefix : OPTIONAL si_prefix;
    name   : si_unit_name;
  DERIVE
    SELF\named_unit.dimensions : dimensional_exponents :=
                                    dimensions_for_si_unit(SELF.name);
END_ENTITY; -- si_unit

ENTITY solid_model
  SUPERTYPE OF (ONEOF (csg_solid,manifold_solid_brep,swept_area_solid,
      solid_replica))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- solid_model

ENTITY solid_replica
  SUBTYPE OF (solid_model);
    parent_solid   : solid_model;
    transformation : cartesian_transformation_operator_3d;
  WHERE
    WR1: acyclic_solid_replica(SELF,parent_solid);
END_ENTITY; -- solid_replica

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 start_order
  SUBTYPE OF (work_order);
END_ENTITY; -- start_order

ENTITY surface
  SUPERTYPE OF (ONEOF (elementary_surface,swept_surface,bounded_surface,
      offset_surface,surface_replica))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- surface

ENTITY surface_curve
  SUPERTYPE OF (ONEOF (intersection_curve,seam_curve))
  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: ('SHEET_METAL.PCURVE' IN TYPEOF(associated_geometry[1])) OR (
           master_representation <> pcurve_s1);
    WR3: ('SHEET_METAL.PCURVE' IN TYPEOF(associated_geometry[2])) OR (
```

```
              master_representation <> pcurve_s2);
    WR4: NOT ('SHEET_METAL.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 ('SHEET_METAL.CURVE_BOUNDED_SURFACE' IN TYPEOF(
              parent_surface));
END_ENTITY; -- surface_patch

ENTITY surface_replica
  SUBTYPE OF (surface);
    parent_surface : surface;
    transformation : cartesian_transformation_operator_3d;
  WHERE
    WR1: acyclic_surface_replica(SELF,parent_surface);
END_ENTITY; -- surface_replica

ENTITY swept_area_solid
  SUPERTYPE OF (ONEOF (revolved_area_solid,extruded_area_solid))
  SUBTYPE OF (solid_model);
    swept_area : curve_bounded_surface;
  WHERE
    WR1: 'SHEET_METAL.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 tolerance_value;
    lower_bound : measure_with_unit;
    upper_bound : measure_with_unit;
```

```
  WHERE
    WR1: upper_bound.value_component > lower_bound.value_component;
    WR2: upper_bound.unit_component = lower_bound.unit_component;
END_ENTITY; -- tolerance_value

ENTITY tolerance_zone
  SUBTYPE OF (shape_aspect);
    defining_tolerance : SET [1:?] OF geometric_tolerance;
    form               : tolerance_zone_form;
END_ENTITY; -- tolerance_zone

ENTITY tolerance_zone_definition;
    zone       : tolerance_zone;
    boundaries : SET [1:?] OF shape_aspect;
END_ENTITY; -- tolerance_zone_definition

ENTITY tolerance_zone_form;
    name : label;
END_ENTITY; -- tolerance_zone_form

ENTITY topological_representation_item
  SUPERTYPE OF (ONEOF (vertex,edge,face_bound,face,vertex_shell,
      wire_shell,connected_edge_set,connected_face_set,loop ANDOR path))
  SUBTYPE OF (representation_item);
END_ENTITY; -- topological_representation_item

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

ENTITY 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 uncertainty_measure_with_unit
  SUBTYPE OF (measure_with_unit);
    name        : label;
```

```
    description : text;
  WHERE
    WR1: (NOT ('NUMBER' IN TYPEOF(SELF\measure_with_unit.value_component)))
           OR (SELF\measure_with_unit.value_component >= 0);
END_ENTITY; -- uncertainty_measure_with_unit

ENTITY uniform_curve
  SUBTYPE OF (b_spline_curve);
END_ENTITY; -- uniform_curve

ENTITY uniform_surface
  SUBTYPE OF (b_spline_surface);
END_ENTITY; -- uniform_surface

ENTITY vector
  SUBTYPE OF (geometric_representation_item);
    orientation : direction;
    magnitude   : length_measure;
  WHERE
    WR1: magnitude >= 0;
END_ENTITY; -- vector

ENTITY versioned_action_request;
    id          : identifier;
    version     : label;
    purpose     : text;
    description : text;
END_ENTITY; -- versioned_action_request

ENTITY vertex
  SUBTYPE OF (topological_representation_item);
END_ENTITY; -- vertex

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

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

ENTITY vertex_shell
  SUBTYPE OF (topological_representation_item);
    vertex_shell_extent : vertex_loop;
END_ENTITY; -- vertex_shell

ENTITY week_of_year_and_day_date
  SUBTYPE OF (date);
    week_component : week_in_year_number;
    day_component  : OPTIONAL day_in_week_number;
END_ENTITY; -- week_of_year_and_day_date

ENTITY wire_shell
  SUBTYPE OF (topological_representation_item);
```

```
    wire_shell_extent : SET [1:?] OF loop;

  WHERE
    WR1: NOT mixed_loop_type_set(wire_shell_extent);
END_ENTITY; -- wire_shell

ENTITY work_order
  SUPERTYPE OF (ONEOF (change_order,start_order))
  SUBTYPE OF (executed_action);
  UNIQUE
    ur1 : name;
END_ENTITY; -- work_order

ENTITY work_order_relationship
  SUBTYPE OF (action_relationship);
  WHERE
    WR1: (('CHANGE_ORDER' IN (TYPEOF(SELF\action_relationship.
              relating_action) OR 'START_ORDER')) IN TYPEOF(SELF\
              action_relationship.relating_action)) AND (('CHANGE_ORDER'
              IN (TYPEOF(SELF\action_relationship.related_action) OR
              'START_ORDER')) IN TYPEOF(SELF\action_relationship.
              related_action));
END_ENTITY; -- work_order_relationship

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 =
       'sheet_metal'))) = 1 ))) = 0;
END_RULE; -- application_context_requires_ap_definition

RULE approval_requires_approval_date_time FOR (approval,
              approval_date_time);
 WHERE
   WR1: SIZEOF(QUERY ( a <* approval | (NOT (SIZEOF(QUERY ( apo <*
             approval_date_time | (a :=: apo.dated_approval) )) = 1)) )) = 0;
END_RULE; -- approval_requires_approval_date_time

RULE approval_requires_approval_person_organization FOR (approval,
            approval_person_organization);
WHERE
  WR1: SIZEOF(QUERY ( a <* approval | (NOT (SIZEOF(QUERY ( apo <*
            approval_person_organization | (a :=: apo.authorized_approval) ))
            >= 1)) )) = 0;
END_RULE; -- approval_requires_approval_person_organization


RULE cardinality_action_method_to_process_plan FOR (action_method,
           process_plan);
WHERE
  WR1: SIZEOF(QUERY ( am <* action_method | ((am.name =
           'part process plan template') AND (NOT (SIZEOF(QUERY ( pp <*
```

```
            process_plan | (am :=: pp.chosen_method) )) = 1))) )) = 0;
END_RULE; -- cardinality_action_method_to_process_plan

RULE cardinality_product_definition_process_to_action_resource_requirement
FOR (product_definition_process, action_resource_requirement);
WHERE
  WR1: SIZEOF(QUERY ( pdp <* product_definition_process |
    (NOT (SIZEOF(QUERY ( arr <* action_resource_requirement |
     (pdp :=: arr.operations) )) <= 1)) )) = 0;
END_RULE;
-- cardinality_product_definition_process_to_action_resource_requirement

RULE cardinality_sm_action_assignment_to_process_plan FOR (
            sheet_metal_action_assignment, process_plan);
WHERE
  WR1: SIZEOF(QUERY ( pp <* process_plan | (NOT (SIZEOF(QUERY ( smaa <*
            sheet_metal_action_assignment | (pp IN smaa.items) )) <= 1)) ))
            = 0;
END_RULE; -- cardinality_sm_action_assignment_to_process_plan

RULE cardinality_sm_action_assignment_to_product_definition FOR (
            sheet_metal_action_assignment, product_definition);
WHERE
  WR1: SIZEOF(QUERY ( pd <* product_definition | (NOT (SIZEOF(
            QUERY ( smaa <* sheet_metal_action_assignment |
     (pd IN smaa.items) )) <= 1)) )) = 0;
END_RULE; -- cardinality_sm_action_assignment_to_product_definition

RULE cardinality_sm_action_assignment_to_product_definition_formation FOR (
            sheet_metal_action_assignment, product_definition_formation);
WHERE
  WR1: SIZEOF(QUERY ( pdf <* product_definition_formation | (NOT (
            SIZEOF(QUERY ( smaa <* sheet_metal_action_assignment | (pdf IN
             smaa.items) )) <= 1)) )) = 0;
END_RULE; --cardinality_sm_action_assignment_to_product_definition_formation

RULE cardinality_sm_organization_assignment_to_work_order FOR (
            sheet_metal_organization_assignment, work_order);
WHERE
  WR1: SIZEOF(QUERY ( wo <* work_order | (NOT (SIZEOF(QUERY ( smoa <*
            sheet_metal_organization_assignment | (('effector' = smoa.role)
            AND (wo IN smoa.items)) )) >= 1)) )) = 0;
END_RULE; -- cardinality_sm_organization_assignment_to_work_order

RULE cardinality_start_order_to_sm_action_assignment FOR (
            sheet_metal_action_assignment, start_order);
WHERE
  WR1: SIZEOF(QUERY ( so <* start_order | (NOT (SIZEOF(QUERY ( smaa <*
            sheet_metal_action_assignment | (so :=: smaa.assigned_action) ))
            >= 1)) )) = 0;
END_RULE; -- cardinality_start_order_to_sm_action_assignment

RULE cardinality_work_order_to_sm_contract_assig_or_sm_org_assig FOR (
            work_order, sheet_metal_contract_assignment,
            sheet_metal_organization_assignment);
```

```
WHERE
  WR1: SIZEOF(QUERY ( wo <* work_order | ((NOT (SIZEOF(QUERY ( smca <*
           sheet_metal_contract_assignment | (wo IN smca.items) )) = 1))
           XOR (SIZEOF(QUERY ( smoa <*
           sheet_metal_organization_assignment | (wo IN smoa.items) )) =
           1)) )) = 0;
END_RULE; -- cardinality_work_order_to_sm_contract_assig_or_sm_org_assig

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 dependent_instantiation_for_action FOR (action);
WHERE
  WR1: SIZEOF(QUERY ( a <* action | (NOT (SIZEOF(USEDIN(a,'')) >= 1)) ))
           = 0;
END_RULE; -- dependent_instantiation_for_action

RULE dependent_instantiation_for_action_request_solution FOR (
           action_request_solution);
WHERE
  WR1: SIZEOF(QUERY ( ars <* action_request_solution | (NOT (SIZEOF(
           USEDIN(ars,'')) >= 1)) )) = 0;
END_RULE; -- dependent_instantiation_for_action_request_solution

RULE dependent_instantiation_for_application_context FOR (
           application_context);
WHERE
  WR1: SIZEOF(QUERY ( ac <* application_context | (NOT (SIZEOF(USEDIN(ac,
           '')) >= 1)) )) = 0;
END_RULE; -- dependent_instantiation_for_application_context

RULE dependent_instantiation_for_configuration_design FOR (
           configuration_design);
WHERE
  WR1: SIZEOF(QUERY ( cd <* configuration_design | (NOT (SIZEOF(USEDIN(cd,
           '')) >= 1)) )) = 0;
END_RULE; -- dependent_instantiation_for_configuration_design

RULE dependent_instantiation_for_contract FOR (contract);
WHERE
  WR1: SIZEOF(QUERY ( c <* contract | (NOT (SIZEOF(USEDIN(c,'')) >= 1)) ))
           = 0;
END_RULE; -- dependent_instantiation_for_contract

RULE dependent_instantiation_for_date_role FOR (date_role);
WHERE
```

```
  WR1: SIZEOF(QUERY ( dr <* date_role | (NOT(SIZEOF(USEDIN(dr,'')) >= 1)) ))
          = 0;
END_RULE; -- dependent_instantiation_for_date_role

RULE dependent_instantiation_for_date_time_role FOR (date_time_role);
WHERE
  WR1: SIZEOF(QUERY ( dtr <* date_time_role | (NOT (SIZEOF(USEDIN(dtr,''))
          >= 1)) )) = 0;
END_RULE; -- dependent_instantiation_for_date_time_role

RULE dependent_instantiation_for_derived_unit FOR (derived_unit);
WHERE
  WR1: SIZEOF(QUERY ( du <* derived_unit | (NOT (SIZEOF(USEDIN(du,'')) >=
          1)) )) = 0;
END_RULE; -- dependent_instantiation_for_derived_unit

RULE dependent_instantiation_for_document_reference FOR (
          document_reference);
WHERE
  WR1: SIZEOF(QUERY ( dr <* document_reference | (NOT (SIZEOF(USEDIN(dr,''))
          >= 1)) )) = 0;
END_RULE; -- dependent_instantiation_for_document_reference

RULE dependent_instantiation_for_functionally_defined_transformation FOR (
          functionally_defined_transformation);
WHERE
  WR1: SIZEOF(QUERY ( fdt <* functionally_defined_transformation | (NOT (
          SIZEOF(USEDIN(fdt,'')) >= 1)) )) = 0;
END_RULE; -- dependent_instantiation_for_functionally_defined_transformation

RULE dependent_instantiation_for_item_defined_transformation FOR (
          item_defined_transformation);
WHERE
  WR1: SIZEOF(QUERY ( idt <* item_defined_transformation | (NOT (SIZEOF(
          USEDIN(idt,'')) >= 1)) )) = 0;
END_RULE; -- dependent_instantiation_for_item_defined_transformation

RULE dependent_instantiation_for_organization_role FOR (
          organization_role);
WHERE
  WR1: SIZEOF(QUERY ( o_r <* organization_role | (NOT (SIZEOF (USEDIN
                    (o_r,'')) >= 1)) )) = 0;
END_RULE; -- dependent_instantiation_for_organization_role

RULE dependent_instantiation_for_person_and_organization_role FOR (
          person_and_organization_role);
WHERE
  WR1: SIZEOF(QUERY ( paor <* person_and_organization_role | (NOT (SIZEOF(
          USEDIN(paor,'')) >= 1)) )) = 0;
END_RULE; -- dependent_instantiation_for_person_and_organization_role

RULE dependent_instantiation_for_person_role FOR (person_role);
WHERE
  WR1: SIZEOF(QUERY(pr <* person_role | (NOT(SIZEOF(USEDIN(pr,'')) >= 1)) ))
          = 0;
END_RULE; -- dependent_instantiation_for_person_role
```

```
RULE dependent_instantiation_for_tolerance_value FOR (tolerance_value);
WHERE
  WR1: SIZEOF(QUERY ( tr <* tolerance_value | (NOT (SIZEOF(USEDIN(tr,''))
           >= 1)) )) = 0;
END_RULE; -- dependent_instantiation_for_tolerance_value

RULE executed_action_requires_approval FOR (executed_action,
           sheet_metal_approval_assignment);
WHERE
  WR1: SIZEOF(QUERY ( ea <* executed_action | (NOT (SIZEOF(QUERY ( smaa <*
           sheet_metal_approval_assignment | (ea IN smaa.items) )) >= 1)) ))
           = 0;
END_RULE; -- executed_action_requires_approval

RULE executed_action_requires_date_or_date_and_time FOR (executed_action,
           sheet_metal_date_assignment,
           sheet_metal_date_and_time_assignment);
WHERE
  WR1: SIZEOF(QUERY ( ea <* executed_action | (NOT ((SIZEOF(
           QUERY ( smda <* sheet_metal_date_assignment | (ea IN smda.
           items) )) + SIZEOF(QUERY ( smdata <*
           sheet_metal_date_and_time_assignment | (ea IN smdata.items) )))
           = 1)) )) = 0;
END_RULE; -- executed_action_requires_date_or_date_and_time

RULE executed_action_requires_organization FOR (executed_action,
           sheet_metal_organization_assignment);
WHERE
  WR1: SIZEOF(QUERY ( ea <* executed_action | (NOT (SIZEOF(QUERY ( smoa <*
       sheet_metal_organization_assignment | (ea IN smoa.items) )) >= 1)) ))
           = 0;
END_RULE; -- executed_action_requires_organization

RULE global_uncertainty_assigned_context_constraint FOR (
           global_uncertainty_assigned_context);
WHERE
  WR1: SIZEOF(QUERY ( guac <* global_uncertainty_assigned_context | (
           NOT (SIZEOF(guac.uncertainty) = 2)) )) = 0;
  WR2: SIZEOF(QUERY ( guac <* global_uncertainty_assigned_context | (
           NOT ((SIZEOF(QUERY ( x <* guac.uncertainty | (
           'SHEET_METAL.LENGTH_UNIT' IN TYPEOF(x.unit_component)) )) = 1)
           AND (SIZEOF(QUERY ( x <* guac.uncertainty | (
           'SHEET_METAL.PLANE_ANGLE_UNIT' IN TYPEOF(x.unit_component)) ))
           = 1))) )) = 0;
END_RULE; -- global_uncertainty_assigned_context_constraint

RULE process_plan_requires_product_definition FOR (process_plan,
           sheet_metal_action_assignment);
WHERE
  WR1: SIZEOF(QUERY ( pp <* process_plan | (NOT (SIZEOF(QUERY ( smaa <*
           sheet_metal_action_assignment | ((smaa\action_assignment.
           assigned_action :=: pp\action) AND (SIZEOF(QUERY( i <* smaa.items
            | (('SHEET_METAL.' + 'PRODUCT_DEFINITION') IN TYPEOF(i)) ))
           >= 1)) )) = 1)) )) = 0;
END_RULE; -- process_plan_requires_product_definition
```

```
RULE product_definition_formation_requires_date_or_date_and_time FOR (
          product_definition_formation, sheet_metal_date_assignment,
          sheet_metal_date_and_time_assignment);
WHERE
  WR1: SIZEOF(QUERY ( pdf <* product_definition_formation | (NOT ((
          SIZEOF(QUERY ( smda <* sheet_metal_date_assignment | (pdf IN
          smda.items) )) + SIZEOF(QUERY ( smdata <*
          sheet_metal_date_and_time_assignment | (pdf IN smdata.items) )))
          = 1)) )) = 0;
END_RULE; -- product_definition_formation_requires_date_or_date_and_time

RULE product_definition_formation_requires_product_definition FOR (
          product_definition_formation, product_definition);
WHERE
  WR1: SIZEOF(QUERY ( pdf <* product_definition_formation | (NOT (SIZEOF(
      QUERY ( pd <* product_definition | (pdf :=: pd.formation) )) >= 1)) ))
      = 0;
END_RULE; -- product_definition_formation_requires_product_definition

RULE product_requires_product_definition_formation FOR (product,
          product_definition_formation);
WHERE
  WR1: SIZEOF(QUERY ( p <* product | (NOT (SIZEOF(QUERY ( pdf <*
         product_definition_formation | (p :=: pdf.of_product) )) >= 1)) ))
          = 0;
END_RULE; -- product_requires_product_definition_formation

RULE product_requires_product_type FOR (product, product_type);
WHERE
  WR1: SIZEOF(QUERY ( p <* product | (NOT (SIZEOF(QUERY ( pt <*
          product_type | (SIZEOF(p IN pt\
          product_related_product_category.products) = 1) )) = 1)) )) =
          0;
END_RULE; -- product_requires_product_type

RULE restrict_assembly_component_usage_substitute FOR (
          assembly_component_usage_substitute, product_type);
WHERE
  WR1: SIZEOF(QUERY ( acus <* assembly_component_usage_substitute | (NOT (
          ('SHEET_METAL.QUANTIFIED_ASSEMBLY_COMPONENT_USAGE' IN TYPEOF(
          acus.base)) AND (
          'SHEET_METAL.QUANTIFIED_ASSEMBLY_COMPONENT_USAGE' IN TYPEOF(acus
          .substitute)))) )) = 0;
  WR2: SIZEOF(QUERY ( acus <* assembly_component_usage_substitute | (
          product_type_of_product(acus.base.relating_product_definition.
          formation.of_product,product_type).name =
          product_type_of_product(acus.substitute.
          relating_product_definition.formation.of_product,product_type)
          .name) ));
END_RULE; -- restrict_assembly_component_usage_substitute

RULE restrict_date_role FOR (date_role);
WHERE
  WR1: SIZEOF(QUERY ( dr <* date_role | (NOT (dr.name IN [
          'creation date','revision date','review date',
          'production year date','completion date','order date',
```

```
            'preliminary review date','start date','request date'])) )) =
            0;
  END_RULE; -- restrict_date_role

  RULE restrict_date_time_role FOR (date_time_role);
  WHERE
    WR1: SIZEOF(QUERY ( dtr <* date_time_role | (NOT (dtr.name IN [
            'creation date','revision date','review date',
            'completion date','order date','preliminary review date',
            'start date','request date'])) )) = 0;
  END_RULE; -- restrict_date_time_role

  RULE restrict_organization_role FOR (organization_role);
  WHERE
    WR1: SIZEOF(QUERY ( o_r <* organization_role | (NOT
      (o_r.name IN ['design supplier','effector',
      'implementor', 'order source','plant','supplier'])) )) = 0;
  END_RULE; -- restrict_organization_role

  RULE restrict_person_and_organization_role FOR (
            person_and_organization_role);
  WHERE
    WR1: SIZEOF(QUERY ( paor <* person_and_organization_role | (NOT (
            paor.name IN ['data owner','planner','implementor'])) )) =
            0;
  END_RULE; -- restrict_person_and_organization_role

  RULE restrict_person_role FOR (person_role);
  WHERE
    WR1: SIZEOF(QUERY ( pr <* person_role | (NOT (pr.name IN [
            'designer','implementor','purchaser','work requestor'])) )) =
            0;
  END_RULE; -- restrict_person_role

  RULE restrict_product_definition_context FOR (product_definition_context);
  WHERE
    WR1: SIZEOF(QUERY ( pdc <* product_definition_context | (NOT ((pdc.
            life_cycle_stage = 'design') OR (pdc.life_cycle_stage =
            'in process') OR (pdc.life_cycle_stage =
            'as physically modelled') OR (pdc.life_cycle_stage =
            'as design constrained'))) )) = 0;
  END_RULE; -- restrict_product_definition_context

  RULE restrict_quantified_assembly_component_usage FOR (
            quantified_assembly_component_usage, product_type);
  WHERE
    WR1: SIZEOF(QUERY ( qacu <* quantified_assembly_component_usage | (
            product_type_of_product(qacu.relating_product_definition.
            formation.of_product,product_type).name =
            product_type_of_product(qacu.related_product_definition.formation
            .of_product,product_type).name) ));
  END_RULE; -- restrict_quantified_assembly_component_usage

  RULE same_type_for_relationships FOR (product_definition_relationship,
            product_type);
  WHERE
```

```
  WR1: SIZEOF(QUERY ( pdr <* product_definition_relationship | (NOT ((((
           pdr.name <> ('assembly component' AND pdr.name)) <> ('mating'
           AND pdr.name)) <> 'symmetrical') OR (pdr.name IN ([
           'assembly component','mating','symmetrical'] AND (
           product_type_of_product(pdr.relating_product_definition.
           formation.of_product,product_type).name =
           product_type_of_product(pdr.related_product_definition.
           formation.of_product,product_type).name))))) )) = 0;
END_RULE; -- same_type_for_relationships

RULE subtype_mandatory_for_shape_representation FOR (
           shape_representation);
WHERE
  WR1: SIZEOF(QUERY ( sr <* shape_representation | (NOT ((SIZEOF([
           'SHEET_METAL.ADVANCED_BREP_SHAPE_REPRESENTATION',
           'SHEET_METAL.FACETED_BREP_SHAPE_REPRESENTATION',
           'SHEET_METAL.CSG_SHAPE_REPRESENTATION',
           'SHEET_METAL.GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION',
           'SHEET_METAL.EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION',
           'SHEET_METAL.SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION',
           'SHEET_METAL.GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_REPRESENTATION',
           'SHEET_METAL.MANIFOLD_SURFACE_SHAPE_REPRESENTATION']) *
           TYPEOF('sr')) = 1)) )) = 0;
END_RULE; -- subtype_mandatory_for_shape_representation

RULE subtype_mandatory_work_order FOR (work_order);
WHERE
  WR1: SIZEOF(QUERY ( wo <* work_order | ((NOT (
           'SHEET_METAL.CHANGE_ORDER' IN TYPEOF(wo))) AND (NOT (
           'SHEET_METAL.START_ORDER' IN TYPEOF(wo)))) )) = 0;
END_RULE; -- subtype_mandatory_work_order

RULE work_request_requires_date_or_date_and_time FOR
  (versioned_action_request, sheet_metal_date_assignment,
   sheet_metal_date_and_time_assignment);
WHERE
  WR1: SIZEOF (QUERY (varq <* versioned_action_request |
      NOT (SIZEOF (QUERY (smda <* sheet_metal_date_assignment |
      varq IN smda.items)) +
          SIZEOF (QUERY (smdata <* sheet_metal_date_and_time_assignment |
      varq IN smdata.items)) =1 ) )) = 0;
END_RULE; -- work_request_requires_date_or_date_and_time

RULE work_request_requires_person FOR (versioned_action_request,
           sheet_metal_person_assignment);
WHERE
  WR1: SIZEOF(QUERY ( varq <* versioned_action_request | (NOT (SIZEOF(
       QUERY ( smpa <* sheet_metal_person_assignment |
                (varq IN smpa.items))) >= 1)) )) = 0;
END_RULE; -- work_request_requires_person

FUNCTION acyclic_action_relationship(
             relation: action_relationship;
             relatives: SET [1:?] OF action;
             specific_relation: STRING
    ): LOGICAL;
```

```
    LOCAL
      i               : INTEGER;
      x               : SET [1:?] OF action_relationship;
      local_relatives : SET [1:?] OF action;
    END_LOCAL;
    REPEAT i := 1 TO HIINDEX(relatives) BY 1;
      IF relation.relating_action :=: relatives[i] THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
    x := bag_to_set(USEDIN(relation.relating_action,specific_relation));
    local_relatives := relatives + relation.relating_action;
    IF SIZEOF(x) > 0 THEN
      REPEAT i := 1 TO HIINDEX(x) BY 1;
        IF NOT acyclic_action_relationship(x[i],local_relatives,
            specific_relation) THEN
          RETURN(FALSE);
        END_IF;
      END_REPEAT;
    END_IF;
    RETURN(TRUE);
  END_FUNCTION; -- acyclic_action_relationship

  FUNCTION acyclic_curve_replica(
              rep: curve_replica;
              parent: curve
      ): BOOLEAN;
    IF NOT ('SHEET_METAL.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 | ('SHEET_METAL.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],'')) | (
           'SHEET_METAL.REPRESENTATION_ITEM' IN TYPEOF(z)) );
      IF NOT acyclic_mapped_representation(parent_set,y) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(TRUE);
END_FUNCTION; -- acyclic_mapped_representation

FUNCTION acyclic_point_replica(
             rep: point_replica;
             parent: point
    ): BOOLEAN;
  IF NOT ('SHEET_METAL.POINT_REPLICA' IN TYPEOF(parent)) THEN
    RETURN(TRUE);
  END_IF;
  IF parent :=: rep THEN
    RETURN(FALSE);
  ELSE
    RETURN(acyclic_point_replica(rep,parent\point_replica.parent_pt));
  END_IF;
END_FUNCTION; -- acyclic_point_replica

FUNCTION acyclic_product_category_relationship(
             relation: product_category_relationship;
             children: SET OF product_category
    ): LOGICAL;
  LOCAL
    i              : INTEGER;
    x              : SET OF product_category_relationship;
    local_children : SET OF product_category;
  END_LOCAL;
  REPEAT i := 1 TO HIINDEX(children) BY 1;
    IF relation.category :=: children[i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  x := bag_to_set(USEDIN(relation.category,'SHEET_METAL.' +
       '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 acyclic_solid_replica(
               rep: solid_replica;
               parent: solid_model
      ): BOOLEAN;
    IF NOT ('SHEET_METAL.SOLID_REPLICA' IN TYPEOF(parent)) THEN
      RETURN(TRUE);
    END_IF;
    IF parent :=: rep THEN
      RETURN(FALSE);
    ELSE
      RETURN(acyclic_solid_replica(rep,parent\solid_replica.parent_solid));
    END_IF;
  END_FUNCTION; -- acyclic_solid_replica

  FUNCTION acyclic_surface_replica(
               rep: surface_replica;
               parent: surface
      ): BOOLEAN;
    IF NOT ('SHEET_METAL.SURFACE_REPLICA' IN TYPEOF(parent)) THEN
      RETURN(TRUE);
    END_IF;
    IF parent :=: rep THEN
      RETURN(FALSE);
```

```
    ELSE
      RETURN(acyclic_surface_replica(rep,parent\surface_replica.
          parent_surface));
    END_IF;
  END_FUNCTION; -- acyclic_surface_replica

  FUNCTION associated_surface(
              arg: pcurve_or_surface
      ): surface;
    LOCAL
      surf : surface;
    END_LOCAL;
    IF 'SHEET_METAL.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 basis_curve_check(
            cv: curve
    ): BOOLEAN;
  IF ('SHEET_METAL.B_SPLINE_CURVE' IN TYPEOF(cv)) AND (cv\b_spline_curve
      .self_intersect = FALSE) THEN
    RETURN(TRUE);
  ELSE
    IF SIZEOF(['SHEET_METAL.BOUNDED_CURVE','SHEET_METAL.CONIC',
        'SHEET_METAL.LINE'] * TYPEOF(cv)) = 1 THEN
      RETURN(TRUE);
    ELSE
      IF SIZEOF(['SHEET_METAL.BOUNDED_CURVE','SHEET_METAL.CURVE_REPLICA']
          * TYPEOF(cv)) = 1 THEN
        RETURN(basis_curve_check(cv\curve_replica.parent_curve));
      ELSE
        IF (SIZEOF(['SHEET_METAL.BOUNDED_CURVE',
            'SHEET_METAL.OFFSET_CURVE_3D'] * TYPEOF(cv)) = 1) AND (cv\
          offset_curve_3d.self_intersect = FALSE) THEN
          RETURN(basis_curve_check(cv\offset_curve_3d.basis_curve));
        ELSE
          IF SIZEOF(['SHEET_METAL.BOUNDED_CURVE','SHEET_METAL.PCURVE'] *
              TYPEOF(cv)) = 1 THEN
            RETURN(basis_curve_check(cv\pcurve.reference_to_curve\
                representation.items[1]) AND basis_surface_check(cv\
                pcurve.basis_surface));
          ELSE
            IF SIZEOF(['SHEET_METAL.BOUNDED_CURVE',
                'SHEET_METAL.SURFACE_CURVE'] * TYPEOF(cv)) = 1 THEN
              IF basis_curve_check(cv\surface_curve.curve_3d) THEN
                REPEAT i := 1 TO SIZEOF(cv\surface_curve.
                    associated_geometry) BY 1;
                  IF 'SHEET_METAL.SURFACE' IN TYPEOF(cv\surface_curve.
                      associated_geometry[i]) THEN
                    IF NOT basis_surface_check(cv\surface_curve.
                        associated_geometry[i]) THEN
```

```
                    RETURN(FALSE);
                  END_IF;
                ELSE
                  IF 'SHEET_METAL.PCURVE' IN TYPEOF(cv\surface_curve.
                      associated_geometry[i]) THEN
                    IF NOT basis_curve_check(cv\surface_curve.
                        associated_geometry[i]) THEN
                      RETURN(FALSE);
                    END_IF;
                  END_IF;
                END_IF;
              END_REPEAT;
              RETURN(TRUE);
            END_IF;
          ELSE
            IF 'SHEET_METAL.POLYLINE' IN TYPEOF(cv) THEN
              IF SIZEOF(cv\polyline.points) >= 3 THEN
                RETURN(TRUE);
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION; -- basis_curve_check

FUNCTION basis_surface_check(
              surf: surface
    ): BOOLEAN;
  IF 'SHEET_METAL.ELEMENTARY_SURFACE' IN TYPEOF(surf) THEN
    RETURN(TRUE);
  ELSE
    IF 'SHEET_METAL.SWEPT_SURFACE' IN TYPEOF(surf) THEN
      RETURN(basis_curve_check(surf\swept_surface.swept_curve));
    ELSE
      IF ('SHEET_METAL.OFFSET_SURFACE' IN TYPEOF(surf)) AND (surf\
          offset_surface.self_intersect = FALSE) THEN
        RETURN(basis_surface_check(surf\offset_surface.basis_surface));
      ELSE
        IF 'SHEET_METAL.SURFACE REPLICA' IN TYPEOF(surf) THEN
          RETURN(basis_surface_check(surf\surface_replica.parent_surface));
        ELSE
          IF ('SHEET_METAL.B_SPLINE_SURFACE' IN TYPEOF(surf)) AND (surf\
              b_spline_surface.self_intersect = FALSE) THEN
            RETURN(TRUE);
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION; -- basis_surface_check
```

```
  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 ('SHEET_METAL.PCURVE' IN TYPEOF(c\composite_curve.segments[k]
           .parent_curve))) AND (NOT ('SHEET_METAL.SURFACE_CURVE' IN TYPEOF(
           c\composite_curve.segments[k].parent_curve))) AND (NOT (
           'SHEET_METAL.COMPOSITE_CURVE_ON_SURFACE' IN TYPEOF(c\
```

```
        composite_curve.segments[k].parent_curve))) THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  RETURN(TRUE);
END_FUNCTION; -- constraints_composite_curve_on_surface

FUNCTION constraints_geometry_shell_based_surface_model(
            m: shell_based_surface_model
    ): BOOLEAN;
  LOCAL
    result : BOOLEAN := TRUE;
  END_LOCAL;
  REPEAT j := 1 TO SIZEOF(m.sbsm_boundary) BY 1;
    IF (NOT ('SHEET_METAL.OPEN_SHELL' IN TYPEOF(m.sbsm_boundary[j])))
        AND (NOT ('SHEET_METAL.CLOSED_SHELL' IN TYPEOF(m.sbsm_boundary[j])))
         THEN
      result := FALSE;
      RETURN(result);
    END_IF;
  END_REPEAT;
  RETURN(result);
END_FUNCTION; -- constraints_geometry_shell_based_surface_model

FUNCTION constraints_geometry_shell_based_wireframe_model(
            m: shell_based_wireframe_model
    ): BOOLEAN;
  LOCAL
    result : BOOLEAN := TRUE;
  END_LOCAL;
  REPEAT j := 1 TO SIZEOF(m.sbwm_boundary) BY 1;
    IF (NOT ('SHEET_METAL.WIRE_SHELL' IN TYPEOF(m.sbwm_boundary[j])))
        AND (NOT ('SHEET_METAL.VERTEX_SHELL' IN TYPEOF(m.sbwm_boundary[j])))
         THEN
      result := FALSE;
      RETURN(result);
    END_IF;
  END_REPEAT;
  RETURN(result);
END_FUNCTION; -- constraints_geometry_shell_based_wireframe_model

FUNCTION constraints_param_b_spline(
            degree, up_knots, up_cp: INTEGER;
            knot_mult: LIST OF INTEGER;
            knots: LIST OF parameter_value
    ): BOOLEAN;
  LOCAL
    k      : INTEGER;
    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 (('SHEET_METAL.BSPLINE_SURFACE' IN TYPEOF(s.segments[i][j]
            .parent_surface)) OR ('SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 gbsf_check_curve(
            cv: curve
    ): BOOLEAN;
  IF SIZEOF(['SHEET_METAL.BOUNDED_CURVE','SHEET_METAL.CIRCLE',
      'SHEET_METAL.ELLIPSE'] * TYPEOF(cv)) = 1 THEN
    RETURN(TRUE);
  ELSE
    IF ('SHEET_METAL.B_SPLINE_CURVE' IN TYPEOF(cv)) AND (cv\
        b_spline_curve.self_intersect = FALSE) THEN
      RETURN(TRUE);
    ELSE
      IF ('SHEET_METAL.COMPOSITE_CURVE' IN TYPEOF(cv)) AND (cv\
          composite_curve.self_intersect = FALSE) THEN
        RETURN(SIZEOF(QUERY ( seg <* cv\composite_curve.segments | (NOT
            gbsf_check_curve(seg.parent_curve)) )) = 0);
      ELSE
        IF SIZEOF(['SHEET_METAL.BOUNDED_CURVE',
            'SHEET_METAL.CURVE_REPLICA'] * TYPEOF(cv)) = 1 THEN
          RETURN(gbsf_check_curve(cv\curve_replica.parent_curve));
        ELSE
          IF (SIZEOF(['SHEET_METAL.BOUNDED_CURVE',
              'SHEET_METAL.OFFSET_CURVE_3D'] * TYPEOF(cv)) = 1) AND (cv\
              offset_curve_3d.self_intersect = FALSE) THEN
            RETURN(gbsf_check_curve(cv\offset_curve_3d.basis_curve));
          ELSE
            IF SIZEOF(['SHEET_METAL.BOUNDED_CURVE','SHEET_METAL.PCURVE']
                * TYPEOF(cv)) = 1 THEN
              RETURN(gbsf_check_curve(cv\pcurve.reference_to_curve.items
                  [1]) AND gbsf_check_surface(cv\pcurve.basis_surface));
            ELSE
              IF 'SHEET_METAL.POLYLINE' IN TYPEOF(cv) THEN
                IF (SIZEOF(cv\polyline.points) >= 3) AND (SIZEOF(
                    bag_to_set(USEDIN(cv,'')) - bag_to_set(USEDIN(cv,
                    'GEOMETRY_SCHEMA.INTERSECTION_CURVE.BASIS_CURVE'))) =
                     0) THEN
                  RETURN(TRUE);
                END_IF;
```

```
                ELSE
                  IF SIZEOF(['SHEET_METAL.BOUNDED_CURVE',
                      'SHEET_METAL.SURFACE_CURVE'] * TYPEOF(cv)) = 1 THEN
                    IF gbsf_check_curve(cv\surface_curve.curve_3d) THEN
                      REPEAT i := 1 TO SIZEOF(cv\surface_curve.
                          associated_geometry) BY 1;
                        IF 'SHEET_METAL.SURFACE' IN TYPEOF(cv\
                            surface_curve.associated_geometry[i]) THEN
                          IF NOT gbsf_check_surface(cv\surface_curve.
                              associated_geometry[i]) THEN
                            RETURN(FALSE);
                          END_IF;
                        ELSE
                          IF 'SHEET_METAL.PCURVE' IN TYPEOF(cv\
                              surface_curve.associated_geometry[i]) THEN
                            IF NOT gbsf_check_curve(cv\surface_curve.
                                associated_geometry[i]) THEN
                              RETURN(FALSE);
                            END_IF;
                          END_IF;
                        END_IF;
                      END_REPEAT;
                      RETURN(TRUE);
                    END_IF;
                  ELSE
                    IF 'SHEET_METAL.TRIMMED_CURVE' IN TYPEOF(cv) THEN
                      IF SIZEOF(['SHEET_METAL.BOUNDED_CURVE',
                          'SHEET_METAL.LINE','SHEET_METAL.PARABOLA',
                          'SHEET_METAL.HYPERBOLA'] * TYPEOF(cv\
                          trimmed_curve.basis_curve)) = 1 THEN
                        RETURN(TRUE);
                      ELSE
                        RETURN(gbsf_check_curve(cv\trimmed_curve.
                            basis_curve));
                      END_IF;
                    END_IF;
                  END_IF;
                END_IF;
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
    RETURN(FALSE);
  END_FUNCTION; -- gbsf_check_curve

  FUNCTION gbsf_check_point(
              pnt: point
    ): BOOLEAN;
    IF 'SHEET_METAL.CARTESIAN_POINT' IN TYPEOF(pnt) THEN
      RETURN(TRUE);
    ELSE
      IF 'SHEET_METAL.POINT_ON_CURVE' IN TYPEOF(pnt) THEN
        RETURN(gbsf_check_curve(pnt\point_on_curve.basis_curve));
      ELSE
```

```
      IF 'SHEET_METAL.POINT_ON_SURFACE' IN TYPEOF(pnt) THEN
        RETURN(gbsf_check_surface(pnt\point_on_surface.basis_surface));
      ELSE
        IF 'SHEET_METAL.DEGENERATE_PCURVE' IN TYPEOF(pnt) THEN
          RETURN(gbsf_check_curve(pnt\degenerate_pcurve.
              reference_to_curve.items[1]) AND gbsf_check_surface(pnt\
              degenerate_pcurve.basis_surface));
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION; -- gbsf_check_point

FUNCTION gbsf_check_surface(
            sf: surface
    ): BOOLEAN;
  IF ('SHEET_METAL.B_SPLINE_SURFACE' IN TYPEOF(sf)) AND (sf\
      b_spline_surface.self_intersect = FALSE) THEN
    RETURN(TRUE);
  ELSE
    IF SIZEOF(['SHEET_METAL.SPHERICAL_SURFACE',
        'SHEET_METAL.TOROIDAL_SURFACE'] * TYPEOF(sf)) = 1 THEN
      RETURN(TRUE);
    ELSE
      IF 'SHEET_METAL.CURVE_BOUNDED_SURFACE' IN TYPEOF(sf) THEN
        IF SIZEOF(['SHEET_METAL.CONICAL_SURFACE',
            'SHEET_METAL.CYLINDRICAL_SURFACE','SHEET_METAL.PLANE'] *
            TYPEOF(sf\curve_bounded_surface.basis_surface)) = 1 THEN
          RETURN(SIZEOF(QUERY ( bcurve <* sf\curve_bounded_surface.
              boundaries | (NOT gbsf_check_curve(bcurve)) )) = 0);
        ELSE
          IF gbsf_check_surface(sf\curve_bounded_surface.basis_surface)
              THEN
            RETURN(SIZEOF(QUERY ( bcurve <* sf\curve_bounded_surface.
                boundaries | (NOT gbsf_check_curve(bcurve)) )) = 0);
          END_IF;
        END_IF;
      ELSE
        IF ('SHEET_METAL.OFFSET_SURFACE' IN TYPEOF(sf)) AND (sf\
            offset_surface.self_intersect = FALSE) THEN
          RETURN(gbsf_check_surface(sf\offset_surface.basis_surface));
        ELSE
          IF 'SHEET_METAL.RECTANGULAR_COMPOSITE_SURFACE' IN TYPEOF(sf)
              THEN
            REPEAT i := 1 TO SIZEOF(sf\rectangular_composite_surface.
                segments) BY 1;
              REPEAT j := 1 TO SIZEOF(sf\rectangular_composite_surface.
                  segments[i]) BY 1;
                IF NOT gbsf_check_surface(sf\
                    rectangular_composite_surface.segments[i][j].
                    parent_surface) THEN
                  RETURN(FALSE);
                END_IF;
              END_REPEAT;
            END_REPEAT;
```

```
            RETURN(TRUE);
          ELSE
            IF 'SHEET_METAL.RECTANGULAR_TRIMMED_SURFACE' IN TYPEOF(sf)
                 THEN
              IF SIZEOF(['SHEET_METAL.CONICAL_SURFACE',
                  'SHEET_METAL.CYLINDRICAL_SURFACE','SHEET_METAL.PLANE']
                  * TYPEOF(sf\rectangular_trimmed_surface.basis_surface))
                  = 1 THEN
                RETURN(TRUE);
              ELSE
                RETURN(gbsf_check_surface(sf\rectangular_trimmed_surface
                    .basis_surface));
              END_IF;
            ELSE
              IF 'SHEET_METAL.SURFACE_REPLICA' IN TYPEOF(sf) THEN
                RETURN(gbsf_check_surface(sf\surface_replica.
                     parent_surface));
              ELSE
                IF 'SHEET_METAL.SWEPT_SURFACE' IN TYPEOF(sf) THEN
                  RETURN(gbsf_check_curve(sf\swept_surface.swept_curve));
                END_IF;
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
    RETURN(FALSE);
  END_FUNCTION; -- gbsf_check_surface

  FUNCTION get_basis_surface(
              c: curve_on_surface
      ): SET [0:2] OF surface;
    LOCAL
      surfs : SET [0:2] OF surface;
      n     : INTEGER;
    END_LOCAL;
    surfs := [];
    IF 'SHEET_METAL.PCURVE' IN TYPEOF(c) THEN
      surfs := [c\pcurve.basis_surface];
    ELSE
      IF 'SHEET_METAL.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 'SHEET_METAL.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,'SHEET_METAL.REPRESENTATION.ITEMS') * cntxt.
      representations_in_context) > 0 THEN
    RETURN(TRUE);
  ELSE
    y := QUERY ( z <* USEDIN(item,'') | (
       'SHEET_METAL.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 := 'SHEET_METAL.POLY_LOOP' IN TYPEOF(l[1]);
    REPEAT i := 2 TO SIZEOF(l) BY 1;
      IF ('SHEET_METAL.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;
              schema_name: STRING
      ): SET [1:?] OF closed_shell;
    IF (schema_name + '.BREP_WITH_VOIDS') IN TYPEOF(brep) 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 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 product_type_of_product(
            prod: product;
            types: SET [1:?] OF product_type
    ): product_type;
  REPEAT i := 1 TO HIINDEX(types) BY 1;
    IF prod IN types[i]\product_related_product_category.products THEN
      RETURN(types[i]);
    END_IF;
  END_REPEAT;
END_FUNCTION; -- product_type_of_product

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 'SHEET_METAL.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 sheet_metal_date_correlation(
            e: sheet_metal_date_assignment
    ): BOOLEAN;
  LOCAL
    dt_role : label;
  END_LOCAL;
  dt_role := e\date_assignment.role.name;
  CASE dt_role OF
```

```
    'creation'          :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (SIZEOF(['SHEET_METAL.' +
          'PRODUCT_DEFINITION','SHEET_METAL.PROCESS_PLAN'] * TYPEOF(x)) =
           1) )) THEN
        RETURN(FALSE);
      END_IF;
    'revision'          :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('SHEET_METAL.' + 'PRODUCT_DEFINITION')
          IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'review'            :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('SHEET_METAL.' + 'PROCESS_PLAN') IN
          TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'production year'   :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('SHEET_METAL.' + 'PRODUCT_CONCEPT') IN
           TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'completion'        :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('SHEET_METAL.' + 'EXECUTED_ACTION') IN
           TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'order'             :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('SHEET_METAL.' + 'EXECUTED_ACTION') IN
           TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'preliminary review'  :       IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('SHEET_METAL.' + 'EXECUTED_ACTION') IN
           TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'start'             :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('SHEET_METAL.' + 'EXECUTED_ACTION') IN
           TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'request'           :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('SHEET_METAL.' +
          'VERSIONED_ACTION_REQUEST') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    OTHERWISE           :         RETURN(TRUE);
    END_CASE;
END_FUNCTION; -- sheet_metal_date_correlation

FUNCTION sheet_metal_document_correlation(
            e: sheet_metal_document_reference
    ): BOOLEAN;
  LOCAL
    doc_type : label;
  END_LOCAL;
```

```
  doc_type := e\document_reference.assigned_document.kind.
      product_data_type;
  CASE doc_type OF
    'die layout'                    :         IF SIZEOF(e.items) <>
          SIZEOF(QUERY ( x <* e.items | (('SHEET_METAL.' +
          'PRODUCT_DEFINITION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'die structure'                 :         IF SIZEOF(e.items) <>
          SIZEOF(QUERY ( x <* e.items | (('SHEET_METAL.' +
          'PRODUCT_DEFINITION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'pattern casting'               :         IF SIZEOF(e.items) <>
          SIZEOF(QUERY ( x <* e.items | (('SHEET_METAL.' +
          'PRODUCT_DEFINITION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'press'                         :         IF SIZEOF(e.items) <>
          SIZEOF(QUERY ( x <* e.items | (('SHEET_METAL.' +
          'PRODUCT_DEFINITION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'generating system information'  :        IF SIZEOF(e.items) <>
          SIZEOF(QUERY ( x <* e.items | (('SHEET_METAL.' + 'PROCESS_PLAN')
          IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'applicable standards'          :         IF SIZEOF(e.items) <>
          SIZEOF(QUERY ( x <* e.items | (('SHEET_METAL.' +
          'EXECUTED_ACTION') IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    OTHERWISE           :         RETURN(TRUE);
    END_CASE;
END_FUNCTION; -- sheet_metal_document_correlation

FUNCTION sheet_metal_organization_correlation(
            e: sheet_metal_organization_assignment
    ): BOOLEAN;
  LOCAL
    o_role : label;
  END_LOCAL;
  o_role := e\organization_assignment.role.name;
  CASE o_role OF
    'design supplier'  :          IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('SHEET_METAL.' + 'PRODUCT_DEFINITION')
          IN TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'implementor'    :          IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('SHEET_METAL.' + 'EXECUTED_ACTION') IN
           TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'supplier'       :          IF SIZEOF(e.items) <> SIZEOF(
```

```
          QUERY ( x <* e.items | (SIZEOF(['SHEET_METAL.' + 'CONTRACT',
          'SHEET_METAL.' + 'EXECUTED_ACTION'] * TYPEOF(x)) = 1) )) THEN
        RETURN(FALSE);
      END_IF;
    'order source'   :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('SHEET_METAL.' + 'EXECUTED_ACTION') IN
           TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    'effector'       :         IF SIZEOF(e.items) <> SIZEOF(
          QUERY ( x <* e.items | (('SHEET_METAL.' + 'EXECUTED_ACTION') IN
           TYPEOF(x)) )) THEN
        RETURN(FALSE);
      END_IF;
    OTHERWISE          :         RETURN(TRUE);
    END_CASE;
END_FUNCTION; -- sheet_metal_organization_correlation

FUNCTION sheet_metal_person_and_organization_correlation(
            e: sheet_metal_person_and_organization_assignment
    ): BOOLEAN;
  LOCAL
    p_org_role : label;
  END_LOCAL;
  p_org_role := e\person_and_organization_assignment.role.name;
  CASE p_org_role OF
    'data owner' :         IF SIZEOF(e.items) <> SIZEOF(QUERY ( x <* e.
          items | (SIZEOF(['SHEET_METAL.' + 'PRODUCT_DEFINITION',
          'SHEET_METAL.PROCESS_PLAN'] * TYPEOF(x)) = 1) )) THEN
        RETURN(FALSE);
      END_IF;
    'planner'    :         IF SIZEOF(e.items) <> SIZEOF(QUERY ( x <* e.
          items | (('SHEET_METAL.' + 'PROCESS_PLAN') IN TYPEOF(x)) ))
           THEN
        RETURN(FALSE);
      END_IF;
    'implementor'  :         IF SIZEOF(e.items) <> SIZEOF(QUERY ( x <* e
          .items | (('SHEET_METAL.' + 'EXECUTED_ACTION') IN TYPEOF(x)) ))
           THEN
        RETURN(FALSE);
      END_IF;
    OTHERWISE          :         RETURN(TRUE);
    END_CASE;
END_FUNCTION; -- sheet_metal_person_and_organization_correlation

FUNCTION sheet_metal_person_correlation(
            e: sheet_metal_person_assignment
    ): BOOLEAN;
  LOCAL
    p_role : label;
  END_LOCAL;
  p_role := e\person_assignment.role.name;
  CASE p_role OF
    'designer'       :         IF SIZEOF(e.items) <> SIZEOF(QUERY ( x <*
          e.items | (('SHEET_METAL.' + 'PRODUCT_DEFINITION')
          IN TYPEOF(x)) ))
```

```
            THEN
          RETURN(FALSE);
        END_IF;
      'implementor'   :         IF SIZEOF(e.items) <> SIZEOF(QUERY ( x <*
            e.items | (('SHEET_METAL.' + 'EXECUTED_ACTION') IN TYPEOF(x)) ))
            THEN
          RETURN(FALSE);
        END_IF;
      'purchaser'     :         IF SIZEOF(e.items) <> SIZEOF(QUERY ( x <*
            e.items | (('SHEET_METAL.' + 'CONTRACT') IN TYPEOF(x)) )) THEN
          RETURN(FALSE);
        END_IF;
      'work requestor'  :         IF SIZEOF(e.items) <> SIZEOF(
            QUERY ( x <* e.items | (('SHEET_METAL.' +
            'VERSIONED_ACTION_REQUEST') IN TYPEOF(x)) )) THEN
          RETURN(FALSE);
        END_IF;
      OTHERWISE           :         RETURN(TRUE);
      END_CASE;
  END_FUNCTION; -- sheet_metal_person_correlation

  FUNCTION shell_reversed(
              a_shell: shell
      ): shell;
    LOCAL
      the_reverse : shell;
    END_LOCAL;
    IF 'SHEET_METAL.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 'SHEET_METAL.OPEN_SHELL' IN TYPEOF(a_shell) THEN
        the_reverse := oriented_open_shell(a_shell,FALSE);
      ELSE
        IF 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.EDGE' IN TYPEOF(an_item) THEN
      RETURN(edge_reversed(an_item));
    END_IF;
    IF 'SHEET_METAL.PATH' IN TYPEOF(an_item) THEN
      RETURN(path_reversed(an_item));
    END_IF;
    IF 'SHEET_METAL.FACE_BOUND' IN TYPEOF(an_item) THEN
      RETURN(face_bound_reversed(an_item));
    END_IF;
    IF 'SHEET_METAL.FACE' IN TYPEOF(an_item) THEN
      RETURN(face_reversed(an_item));
    END_IF;
    IF 'SHEET_METAL.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,'SHEET_METAL.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,'')) | (
        'SHEET_METAL.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_calendar_date(
            date: calendar_date
    ): LOGICAL;
  IF NOT ((1 <= date.day_component) AND (date.day_component <= 31))
       THEN
    RETURN(FALSE);
  END_IF;
  CASE date.month_component OF
    4 :          RETURN((1 <= date.day_component) AND (date.day_component
           <= 30));
    6 :          RETURN((1 <= date.day_component) AND (date.day_component
           <= 30));
    9 :          RETURN((1 <= date.day_component) AND (date.day_component
           <= 30));
    11  :          RETURN((1 <= date.day_component) AND (date.
           day_component <= 30));
    2 :          BEGIN
        IF leap_year(date.year_component) THEN
          RETURN((1 <= date.day_component) AND (date.day_component <= 29));
        ELSE
          RETURN((1 <= date.day_component) AND (date.day_component <= 28));
        END_IF;
      END;
    OTHERWISE           :         RETURN(TRUE);
    END_CASE;
END_FUNCTION; -- valid_calendar_date

FUNCTION valid_geometrically_bounded_wf_curve(
            crv: curve;
            schma: STRING
    ): BOOLEAN;
  IF SIZEOF([schma + '.POLYLINE',schma + '.B_SPLINE_CURVE',schma +
      '.ELLIPSE',schma + '.CIRCLE'] * TYPEOF(crv)) = 1 THEN
    RETURN(TRUE);
  ELSE
    IF (schma + '.TRIMMED_CURVE') IN TYPEOF(crv) THEN
      IF SIZEOF([schma + '.LINE',schma + '.PARABOLA',schma +
          '.HYPERBOLA'] * TYPEOF(crv\trimmed_curve.basis_curve)) = 1
           THEN
        RETURN(TRUE);
      ELSE
        RETURN(valid_geometrically_bounded_wf_curve(crv\trimmed_curve.
            basis_curve,schma));
      END_IF;
    ELSE
      IF (schma + '.OFFSET_CURVE_3D') IN TYPEOF(crv) THEN
        RETURN(valid_geometrically_bounded_wf_curve(crv\offset_curve_3d.
            basis_curve,schma));
      ELSE
```

```
        IF (schma + '.CURVE_REPLICA') IN TYPEOF(crv) THEN
          RETURN(valid_geometrically_bounded_wf_curve(crv\curve_replica.
              parent_curve,schma));
        ELSE
          IF (schma + '.COMPOSITE_CURVE') IN TYPEOF(crv) THEN
            RETURN(SIZEOF(QUERY ( ccs <* crv\composite_curve.segments |
                (NOT valid_geometrically_bounded_wf_curve(ccs.
                parent_curve,schma)) )) = 0);
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION; -- valid_geometrically_bounded_wf_curve

FUNCTION valid_geometrically_bounded_wf_point(
            pnt: point;
            schma: STRING
    ): BOOLEAN;
  IF (schma + '.CARTESIAN_POINT') IN TYPEOF(pnt) THEN
    RETURN(TRUE);
  ELSE
    IF (schma + '.POINT_ON_CURVE') IN TYPEOF(pnt) THEN
      RETURN(valid_geometrically_bounded_wf_curve(pnt\point_on_curve.
          basis_curve,schma));
    ELSE
      IF (schma + '.POINT_REPLICA') IN TYPEOF(pnt) THEN
        RETURN(valid_geometrically_bounded_wf_point(pnt\point_replica.
            parent_pt,schma));
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION; -- valid_geometrically_bounded_wf_point
FUNCTION valid_time(
            time: local_time
    ): BOOLEAN;
  IF EXISTS(time.second_component) THEN
    RETURN(EXISTS(time.minute_component));
  ELSE
    RETURN(TRUE);
  END_IF;
END_FUNCTION; -- valid_time

FUNCTION valid_units(
            m: measure_with_unit
    ): BOOLEAN;
  IF 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.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 'SHEET_METAL.POSITIVE_PLANE_ANGLE_MEASURE' IN TYPEOF(m.
      value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,0,0,0,0) THEN
      RETURN(FALSE);
    END_IF;
  END_IF;
  RETURN(TRUE);
END_FUNCTION; -- valid_units

FUNCTION valid_wireframe_edge_curve(
             crv: curve;
             schma: STRING
    ): BOOLEAN;
  IF SIZEOF([schma + '.LINE',schma + '.CIRCLE',schma + '.ELLIPSE',schma
      + '.PARABOLA',schma + '.HYPERBOLA',schma + '.B_SPLINE_CURVE',schma
      + '.POLYLINE'] * TYPEOF(crv)) = 1 THEN
    RETURN(TRUE);
  ELSE
    IF (schma + '.CURVE_REPLICA') IN TYPEOF(crv) THEN
      RETURN(valid_wireframe_edge_curve(crv\curve_replica.parent_curve,
          schma));
    ELSE
      IF (schma + '.OFFSET_CURVE_3D') IN TYPEOF(crv) THEN
        RETURN(valid_wireframe_edge_curve(crv\offset_curve_3d.
            basis_curve,schma));
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION; -- valid_wireframe_edge_curve

FUNCTION valid_wireframe_vertex_point(
             pnt: point;
             schma: STRING
```

```
    ): BOOLEAN;
  IF (schma + '.CARTESIAN_POINT') IN TYPEOF(pnt) THEN
    RETURN(TRUE);
  ELSE
    IF (schma + '.POINT_REPLICA') IN TYPEOF(pnt) THEN
      RETURN(valid_wireframe_vertex_point(pnt\point_replica.parent_pt,
          schma));
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION; -- valid_wireframe_vertex_point

FUNCTION vector_difference(
            arg1, arg2: vector_or_direction
    ): vector;
  LOCAL
    ndim   : INTEGER;
    mag2   : REAL;
    mag1   : REAL;
    mag    : REAL;
    res    : direction;
    vec1   : direction;
    vec2   : direction;
    result : vector;
  END_LOCAL;
  IF (NOT EXISTS(arg1)) OR (NOT EXISTS(arg2)) OR (arg1.dim <> arg2.dim)
       THEN
    result := ?;
  ELSE
    BEGIN
      IF 'SHEET_METAL.VECTOR' IN TYPEOF(arg1) THEN
        mag1 := arg1.magnitude;
        vec1 := arg1.orientation;
      ELSE
        mag1 := 1;
        vec1 := arg1;
      END_IF;
      IF 'SHEET_METAL.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; -- sheet_metal
(*
```

Annex B
(normative)

AIM short names

Table B.1 provides the short names of entities specified in the AIM of this part of ISO 10303. Requirements on the use of the short names are found in the implementation methods included in ISO 10303.

Table B.1 - AIM short names of entities

Long Name	Short Name
AXIS2_PLACEMENT_2D	A2PL2D
AXIS2_PLACEMENT_3D	A2PL3D
ADVANCED_BREP_SHAPE_REPRESENTATION	ABSR
ACTION_METHOD_RELATIONSHIP	ACMTRL
ACTION_PROPERTY_REPRESENTATION	ACPRRP
ACTION_REQUEST_SOLUTION	ACRQSL
ACTION_RESOURCE_REQUIREMENT	ACRSRQ
ACTION_RESOURCE_TYPE	ACRSTY
ACTION_ASSIGNMENT	ACTASS
ACTION_DIRECTIVE	ACTDRC
ACTION	ACTION
ACTION_METHOD	ACTMTH
ACTION_PROPERTY	ACTPRP
ACTION_RELATIONSHIP	ACTRLT

Table B.1 - AIM short names of entities (continued)

Long name	Short name
ACTION_RESOURCE	ACTRSR
ASSEMBLY_COMPONENT_USAGE_SUBSTITUTE	ACUS
ADVANCED_FACE	ADVFC
ACTION_METHOD_WITH_ASSOCIATED_DOCUMENTS	AMWAD
ANGULAR_LOCATION	ANGLCT
APPLICATION_CONTEXT_ELEMENT	APCNEL
APPROVAL_DATE_TIME	APDTTM
APPROVAL_ASSIGNMENT	APPASS
APPLICATION_CONTEXT	APPCNT
APPLICATION_PROTOCOL_DEFINITION	APPRDF
APPROVAL_ROLE	APPRL
APPROVAL_PERSON_ORGANIZATION	APPROR
APPROVAL	APPRVL
APPROVAL_STATUS	APPSTT
ASSEMBLY_COMPONENT_USAGE	ASCMUS
AXIS1_PLACEMENT	AX1PLC
BOOLEAN_RESULT	BLNRSL
BLOCK	BLOCK
BOUNDARY_CURVE	BNDCR

Table B.1 - AIM short names of entities (continued)

Long name	Short name
BOUNDED_CURVE	BNDCRV
BOUNDED_SURFACE	BNDSRF
BREP_WITH_VOIDS	BRWTVD
B_SPLINE_CURVE_WITH_KNOTS	BSCWK
B_SPLINE_CURVE	BSPCR
B_SPLINE_SURFACE	BSPSR
B_SPLINE_SURFACE_WITH_KNOTS	BSSWK
BOX_DOMAIN	BXDMN
BOXED_HALF_SPACE	BXHLSP
BEZIER_CURVE	BZRCRV
BEZIER_SURFACE	BZRSRF
COMPOSITE_CURVE_ON_SURFACE	CCOS
CHANGE_ORDER	CHNORD
CIRCLE	CIRCLE
CALENDAR_DATE	CLNDT
CLOSED_SHELL	CLSSHL
COMPOSITE_CURVE_SEGMENT	CMCRSG
COMPOSITE_CURVE	CMPCRV
CONVERSION_BASED_UNIT	CNBSUN

Table B.1 - AIM short names of entities (continued)

Long name	Short name
CONICAL_SURFACE	CNCSRF
CONTEXT_DEPENDENT_UNIT	CNDPUN
CONNECTED_EDGE_SET	CNEDST
CONNECTED_FACE_SET	CNFCST
CONFIGURATION_DESIGN	CNFDSG
CONFIGURATION_ITEM	CNFITM
CONTRACT_ASSIGNMENT	CNTASS
CONTRACT	CNTRCT
CONTRACT_TYPE	CNTTYP
CONIC	CONIC
CURVE_BOUNDED_SURFACE	CRBNSR
CARTESIAN_POINT	CRTPNT
CARTESIAN_TRANSFORMATION_OPERATOR	CRTROP
CURVE_REPLICA	CRVRPL
CSG_SOLID	CSGSLD
CSG_SHAPE_REPRESENTATION	CSSHRP
CARTESIAN_TRANSFORMATION_OPERATOR_3D	CTO3
CURVE	CURVE
COORDINATED_UNIVERSAL_TIME_OFFSET	CUTO

Table B.1 - AIM short names of entities (continued)

Long name	Short name
CYLINDRICAL_SURFACE	CYLSRF
DATE_AND_TIME_ASSIGNMENT	DATA
DATE	DATE
DATUM	DATUM
DOCUMENT	DCMNT
DOCUMENT_REFERENCE	DCMRFR
DOCUMENT_TYPE	DCMTYP
DIE_DEFINITION_CONSTRAINT_RELATIONSHIP	DDCR
DIE_DEFINITION_CONSTRAINT_SUBSTITUTE	DDCS
DEFINITIONAL_REPRESENTATION	DFNRPR
DEGENERATE_PCURVE	DGNPCR
DEGENERATE_TOROIDAL_SURFACE	DGTRSR
DIMENSIONAL_LOCATION_WITH_PATH	DLWP
DIMENSIONAL_CHARACTERISTIC_REPRESENTATION	DMCHRP
DIMENSIONAL_EXPONENTS	DMNEXP
DIMENSIONAL_LOCATION	DMNLCT
DIMENSIONAL_SIZE	DMNSZ
DIRECTED_ACTION	DRCACT
DIRECTION	DRCTN

Table B.1 - AIM short names of entities (continued)

Long name	Short name
DERIVED_UNIT_ELEMENT	DRUNEL
DERIVED_UNIT	DRVUNT
DATE_AND_TIME	DTANTM
DATE_ASSIGNMENT	DTASS
DATED_EFFECTIVITY	DTDEFF
DATUM_REFERENCE	DTMRFR
DATUM_TARGET	DTMTRG
DATE_ROLE	DTRL
DATE_TIME_ROLE	DTTMRL
EDGE_BASED_WIREFRAME_MODEL	EBWM
EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION	EBWSR
EDGE_CURVE	EDGCRV
EDGE	EDGE
EDGE_LOOP	EDGLP
EFFECTIVITY	EFFCTV
ELLIPSE	ELLPS
ELEMENTARY_SURFACE	ELMSRF
EVALUATED_DEGENERATE_PCURVE	EVDGPC
EXTRUDED_AREA_SOLID	EXARSL

Table B.1 - AIM short names of entities (continued)

Long name	Short name
EXECUTED_ACTION	EXCACT
FACE	FACE
FACETED_BREP_SHAPE_REPRESENTATION	FBSR
FACE_BOUND	FCBND
FACE_OUTER_BOUND	FCOTBN
FACE_SURFACE	FCSRF
FACETED_BREP	FCTBR
FUNCTIONALLY_DEFINED_TRANSFORMATION	FNDFTR
GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION	GBSSR
GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_REPRESENTATION	GBWSR
GLOBAL_UNCERTAINTY_ASSIGNED_CONTEXT	GC
GEOMETRIC_CURVE_SET	GMCRST
GEOMETRIC_REPRESENTATION_CONTEXT	GMRPCN
GEOMETRIC_REPRESENTATION_ITEM	GMRPIT
GEOMETRIC_SET	GMTST
GEOMETRIC_TOLERANCE	GMTTLR
GEOMETRIC_TOLERANCE_WITH_DATUM_REFERENCE	GTWDR
HALF_SPACE_SOLID	HLSPSL
HYPERBOLA	HYPRBL

Table B.1 - AIM short names of entities (continued)

Long name	Short name
INPUT_ITEM_DIE_RELATIONSHIP	IIDR
INTERSECTION_CURVE	INTCRV
ITEM_DEFINED_TRANSFORMATION	ITDFTR
LOCAL_TIME	LCLTM
LINE	LINE
LIMITS_AND_FITS	LMANFT
LENGTH_MEASURE_WITH_UNIT	LMWU
LENGTH_UNIT	LNGUNT
LOOP	LOOP
MODIFIED_GEOMETRIC_TOLERANCE	MDGMTL
MAKE_FROM_USAGE_OPTION	MFUO
MANIFOLD_SOLID_BREP	MNSLBR
MAPPED_ITEM	MPPITM
MEASURE_REPRESENTATION_ITEM	MSRPIT
MANIFOLD_SURFACE_SHAPE_REPRESENTATION	MSSR
MEASURE_WITH_UNIT	MSWTUN
MATERIAL_PROPERTY	MTRPRP
NAMED_UNIT	NMDUNT
OFFSET_CURVE_3D	OFCR3D

Table B.1 - AIM short names of entities (continued)

Long name	Short name
OFFSET_SURFACE	OFFSRF
OPEN_SHELL	OPNSHL
ORIENTED_CLOSED_SHELL	ORCLSH
ORDINAL_DATE	ORDDT
ORGANIZATION_ASSIGNMENT	ORGASS
ORGANIZATION	ORGNZT
ORGANIZATION_ROLE	ORGRL
ORIENTED_EDGE	ORNEDG
ORIENTED_FACE	ORNFC
ORIENTED_PATH	ORNPTH
ORIENTED_OPEN_SHELL	OROPSH
OUTER_BOUNDARY_CURVE	OTBNCR
PLANE_ANGLE_MEASURE_WITH_UNIT	PAMWU
PERSON_AND_ORGANIZATION_ASSIGNMENT	PAOA
PERSON_AND_ORGANIZATION_ROLE	PAOR
PATH	PATH
PCURVE	PCURVE
PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE	PDFWSS
PRODUCT_DEFINITION_WITH_ASSOCIATED_DOCUMENTS	PDWAD

Table B.1 - AIM short names of entities (continued)

Long name	Short name
PERSON	PERSON
PLANE	PLANE
PLANE_ANGLE_UNIT	PLANUN
PLACEMENT	PLCMNT
PLUS_MINUS_TOLERANCE	PLMNTL
POLYLINE	PLYLN
POLY_LOOP	PLYLP
PHYSICALLY_MODELLED_PRODUCT_DEFINITION	PMPD
POINT_ON_CURVE	PNONCR
POINT_ON_SURFACE	PNONSR
POINT_REPLICA	PNTRPL
POINT	POINT
PERSON_AND_ORGANIZATION	PRANOR
PARABOLA	PRBL
PRODUCT_CONCEPT_CONTEXT	PRCNCN
PROCESS_PLAN	PRCPLN
PRODUCT_CATEGORY_RELATIONSHIP	PRCTRL
PRODUCT_CONCEPT	PRDCNC
PRODUCT_CONTEXT	PRDCNT

Table B.1 - AIM short names of entities (continued)

Long name	Short name
PRODUCT	PRDCT
PRODUCT_CATEGORY	PRDCTG
PRODUCT_DEFINITION	PRDDFN
PRODUCT_DEFINITION_CONTEXT	PRDFCN
PRODUCT_DEFINITION_EFFECTIVITY	PRDFEF
PRODUCT_DEFINITION_FORMATION	PRDFFR
PRODUCT_DEFINITION_PROCESS	PRDFPR
PRODUCT_DEFINITION_RELATIONSHIP	PRDFRL
PROPERTY_DEFINITION_REPRESENTATION	PRDFRP
PRODUCT_DEFINITION_SUBSTITUTE	PRDFSB
PRODUCT_DEFINITION_SHAPE	PRDFSH
PRODUCT_DEFINITION_USAGE	PRDFUS
PRODUCT_TYPE	PRDTYP
PRODUCT_RELATED_PRODUCT_CATEGORY	PRPC
PROPERTY_DEFINITION	PRPDFN
PROCESS_PRODUCT_ASSOCIATION	PRPRAS
PARAMETRIC_REPRESENTATION_CONTEXT	PRRPCN
PERSON_ASSIGNMENT	PRSASS
PERSON_ROLE	PRSRL

Table B.1 - AIM short names of entities (continued)

Long name	Short name
PROMISSORY_USAGE_OCCURRENCE	PRUSOC
QUANTIFIED_ASSEMBLY_COMPONENT_USAGE	QACU
QUASI_UNIFORM_CURVE	QSUNCR
QUASI_UNIFORM_SURFACE	QSUNSR
RATIONAL_B_SPLINE_CURVE	RBSC
RATIONAL_B_SPLINE_SURFACE	RBSS
RECTANGULAR_COMPOSITE_SURFACE	RCCMSR
REPARAMETRISED_COMPOSITE_CURVE_SEGMENT	RCCS
RECTANGULAR_TRIMMED_SURFACE	RCTRSR
REQUIREMENT_FOR_ACTION_RESOURCE	RFAR
RIGHT_ANGULAR_WEDGE	RGANWD
RIGHT_CIRCULAR_CONE	RGCRCN
RIGHT_CIRCULAR_CYLINDER	RGCRCY
REPLACEMENT_RELATIONSHIP	RPLRLT
REPRESENTATION_CONTEXT	RPRCNT
REPRESENTATION_ITEM	RPRITM
REPRESENTATION_MAP	RPRMP
REPRESENTATION_RELATIONSHIP	RPRRLT
REPRESENTATION	RPRSNT

Table B.1 - AIM short names of entities (continued)

Long name	Short name
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION	RRWT
RESOURCE_PROPERTY_REPRESENTATION	RSPRRP
RESOURCE_PROPERTY	RSPRPP
REVOLVED_AREA_SOLID	RVARSL
SHELL_BASED_SURFACE_MODEL	SBSM
SHELL_BASED_WIREFRAME_MODEL	SBWM
SHELL_BASED_WIREFRAME_SHAPE_REPRESENTATION	SBWSR
SECURITY_CLASSIFICATION_ASSIGNMENT	SCCLAS
SECURITY_CLASSIFICATION_LEVEL	SCCLLV
SECURITY_CLASSIFICATION	SCRCLS
SHAPE_ASPECT_RELATIONSHIP	SHASRL
SHAPE_DEFINITION_REPRESENTATION	SHDFRP
SHAPE_DIMENSION_REPRESENTATION	SHDMRP
SHAPE_ASPECT	SHPASP
SHAPE_REPRESENTATION	SHPRPR
SHAPE_REPRESENTATION_RELATIONSHIP	SHRPRL
SURFACE_OF_LINEAR_EXTRUSION	SL
SOLID_MODEL	SLDMDL
SOLID_REPLICA	SLDRPL

Table B.1 - AIM short names of entities (continued)

Long name	Short name
SHEET_METAL_APPROVAL_ASSIGNMENT	SM
SHEET_METAL_ACTION_ASSIGNMENT	SMAA
SHEET_METAL_CONTRACT_ASSIGNMENT	SMCA
SEAM_CURVE	SMCRV
SHEET_METAL_DATE_ASSIGNMENT	SMDA
SHEET_METAL_DATE_AND_TIME_ASSIGNMENT	SMDATA
SHEET_METAL_DOCUMENT_REFERENCE	SMDR
SHEET_METAL_ORGANIZATION_ASSIGNMENT	SMOA
SHEET_METAL_PERSON_ASSIGNMENT	SMPA
SHEET_METAL_PERSON_AND_ORGANIZATION_ASSIGNMENT	SMPAOA
SHEET_METAL_SECURITY_CLASSIFICATION_ASSIGNMENT	SMSCA
SEQUENCED_PRODUCT_DEFINITION_RELATIONSHIP	SPDR
SPHERE	SPHERE
SPHERICAL_SURFACE	SPHSRF
SEQUENTIAL_METHOD	SQNMTH
SERIAL_ACTION_METHOD	SRACMT
SURFACE	SRFC
SURFACE_CURVE	SRFCRV
SURFACE_PATCH	SRFPTC

Table B.1 - AIM short names of entities (continued)

Long name	Short name
SURFACE_REPLICA	SRFRPL
SERIAL_NUMBERED_EFFECTIVITY	SRNMEF
SURFACE_OF_REVOLUTION	SROFRV
START_ORDER	STRORD
SI_UNIT	SUNT
SWEPT_AREA_SOLID	SWARSL
SWEPT_SURFACE	SWPSRF
TOLERANCE_VALUE	TLRVL
TOLERANCE_ZONE	TLRZN
TOLERANCE_ZONE_DEFINITION	TLZNDF
TOLERANCE_ZONE_FORM	TLZNFR
TORUS	TORUS
TOPOLOGICAL_REPRESENTATION_ITEM	TPRPIT
TOROIDAL_SURFACE	TRDSRF
TRIMMED_CURVE	TRMCRV
UNCERTAINTY_MEASURE_WITH_UNIT	UMWU
UNIFORM_CURVE	UNFCRV

Table B.1 - AIM short names of entities (concluded)

Long name	Short name
UNIFORM_SURFACE	UNFSRF
VECTOR	VECTOR
VERTEX	VERTEX
VERSIONED_ACTION_REQUEST	VRACRQ
VERTEX_LOOP	VRTLP
VERTEX_POINT	VRTPNT
VERTEX_SHELL	VRTSHL
WEEK_OF_YEAR_AND_DAY_DATE	WOYADD
WORK_ORDER	WRKORD
WORK_ORDER_RELATIONSHIP	WRORRL
WIRE_SHELL	WRSHL

Annex C
(normative)

Implementation method-specific requirements

The implementation method defines what types of exchange behavior are required with respect to this part of ISO 10303. Conformance to this part of ISO 10303 shall be realized in an exchange structure. The file format shall be encoded according to the syntax and EXPRESS language mapping defined in ISO 10303-21 and in the AIM defined in annex A of this part of ISO 10303. The header of the exchange structure shall identify use of this part of ISO 10303 by the schema name 'sheet_metal'.

Annex D
(normative)

Protocol Implementation Conformance Statement (PICS) proforma

This clause lists the optional elements of this part of ISO 10303. An implementation may choose to support any combination of these optional elements. However, certain combinations of options are likely to be implemented together. These combinations are called conformance classes and are described in the subclauses of this annex.

This annex is in the form of a questionnaire. This questionnaire is intended to be filled out by the implementor and may be used in preparation for conformance testing by a testing laboratory. The completed PICS proforma is referred to as a PICS.

A number of options are identified in this standard for possible use by conforming implementations. Some of these options may be dynamically (run-time) selected for use or non-use, for instance, OPTIONAL attributes of an entity. Others shall be statically (configuration-time) selected for use or non-use, such as a particular style of shape representation as defined in a conformance class.

Questions:

For simplicity of reference, an identifier for the product or system with which the tested STEP implementation is packaged in or procured by is required.

1. Product or system identifier (or name):_____

A conforming implementation shall provide at least conformance class 1 functionality for ISO 10303-207. There are fourteen classes defined in this international standard. Each class specifies a subset of ISO 10303-207 AIM constructs. These classes are detailed in clause 6 of this document.

2. Claimed classes of conformance (functionality) - circle choices:
   1-(PM Only)
   2-(PM & PPLAN)
   3-(PM & Wireframe & Topology)
   4-(PM & Wireframe & Surface & No Topology)
   5-(PM & Manifold Surfaces & Topology)
   6-(PM & Faceted BREP)
   7-(PM & Advanced BREP)
   8-(PM & CSG)
   9-(PM & PPLAN & Wireframe & Topology)
   10-(PM & PPLAN & Wireframe & Surface & No Topology)
   11-(PM & PPLAN & Manifold Surfaces & Topology)
   12-(PM & PPLAN & Faceted BREP)
   13-(PM & PPLAN & Advanced BREP)
   14-(PM & PPLAN & CSG)

Conformance to this international standard may be realized in one or more of several different implementation methods. The implementation methods define what types of implementation behavior are required with respect to this international standard.

3. Claimed implementation forms - circle choices:

   Implementation methods: Clear text encoding of the exchange structure (ISO 10303-21)
   Implementation methods: Standard data access interface specification (ISO 10303-22)

If the implementation receives data that does not comply with the requirements in this international standard for the selected conformance class(es), or with the requirements of the 20's series of parts for the selected implementation method, it shall execute a default response. A default response shall be statically set.

4. Default Response:_____

A conforming implementation shall maintain the static options selected throughout subsequent dynamic assessment (testing) without requiring modification. In a user environment, a conforming implementation shall permanently maintain the provision of selected static options, or it shall provide users discretionary control over the changing and setting of the static options, or both (depending on the option).

5. Does the IUT provide any user discretion over the changing and setting of static options?

   Yes or No

6. If yes, which ones?

   (a) Conformance class(es):_____

   (b) Default Response:_____

A statement of conformance shall include identification of at least one party deeming conformance for the implementation.

7. Evaluator(s) (tester or certifier or accreditor):_____

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(207) version(0)}

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

E.2 Schema identification

To provide for unambiguous identification of the schema specifications given in this application protocol sheet_metal in an open information system, object identifiers are assigned as follows:

{iso standard 10303 part(207) version(0) object(1) sheet-metal-schema(1)}

is assigned to the sheet_metal expanded schema (see annex A).

{iso standard 10303 part(207) version(0) object(1) sheet-metal-schema(2)}

is assigned to the sheet_metal 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
(normative)

Application interpreted constructs

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 annex specifies the application interpreted constructs for ISO 10303-207.

NOTE - RESOLUTION 286: (Kobe June 1996)
"SC4 resolves that
- an AIC shall be published as a separate part (500-series document)
- APs shall reference AICs only as a normative reference
- AICs shall be at the same or advanced voting status as an AP in which it is being used at the time of the AP's DIS or FDIS ballot.

SC4 authorises exemption for 10303-202 and 10303-207, recognising the current ballot status of these documents."

As a result of this resolution, ISO 10303-202 and ISO 10303-207 will have a variation in the annex sequence as defined by the ISO TC184/SC4/N433 *Guideline for the development and approval of STEP application protocols, Version 1.2* and ISO TC184/N-432 *Supplementary directives for the drafting and presentation of ISO 10303*. Annex F of this part of ISO 10303 contains text copied verbatim from applicable AICs.

F.1 Application interpreted construct 501: Edge-based wireframe

F.1.1 Scope

This application interpreted construct 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 connected edge sets.

The following are within the scope of this AIC:

— 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 connected edge sets which 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 AIC:

— geometry defined in a two-dimensional coordinate space;

— surface geometry;

— solid geometry;

— shell topology.

F.1.2 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 AIC.

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. References to eliminated constructs are outside the scope of the AIC.

```
*)
SCHEMA aic_edge_based_wireframe;

USE FROM geometric_model_schema                                    -- ISO 10303-42
  (edge_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_on_curve,
   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,
   vertex_loop,
   vertex_point);
(*
```

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
representation_schema	ISO 10303-43
topology_schema	ISO 10303-42

F.1.2.1 aic_edge_based_wireframe entity definition: edge_based_-wireframe_-shape_representation

An **edge_based_wireframe_shape_representation** is a three-dimensional **shape_representation** which represents the shape, or a portion of the shape, of a product by wireframes which are trimmed by edge topology. This includes all 3D curves and topological entities which define a graph of vertices and edges.

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

EXPRESS specification:

```
*)
ENTITY edge_based_wireframe_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (it <* SELF.items |
       NOT (SIZEOF(['AIC_EDGE_BASED_WIREFRAME.EDGE_BASED_WIREFRAME_MODEL',
                    'AIC_EDGE_BASED_WIREFRAME.MAPPED_ITEM',
                    'AIC_EDGE_BASED_WIREFRAME.AXIS2_PLACEMENT_3D'] *
            TYPEOF (it)) = 1
      ))) = 0;
  WR2: SIZEOF (QUERY (it <* SELF.items |
         SIZEOF(['AIC_EDGE_BASED_WIREFRAME.EDGE_BASED_WIREFRAME_MODEL',
                 'AIC_EDGE_BASED_WIREFRAME.MAPPED_ITEM'] * TYPEOF (it))
  = 1)) >= 1;
```

```
  WR3: SIZEOF (QUERY (ebwm <* QUERY (it <* SELF.items |
         ('AIC_EDGE_BASED_WIREFRAME.EDGE_BASED_WIREFRAME_MODEL'
             IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (eb <* ebwm\edge_based_wireframe_model.
                                   ebwm_boundary |
           NOT (SIZEOF (QUERY (edges <* eb.ces_edges |
              NOT ('AIC_EDGE_BASED_WIREFRAME.EDGE_CURVE'
                      IN TYPEOF (edges)))) = 0
      ))) = 0))) = 0;
  WR4: SIZEOF (QUERY (ebwm <* QUERY (it <* SELF.items |
          ('AIC_EDGE_BASED_WIREFRAME.EDGE_BASED_WIREFRAME_MODEL'
              IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (eb <* ebwm\edge_based_wireframe_model.
                                   ebwm_boundary |
           NOT (SIZEOF (QUERY (pline_edges <* QUERY (edges <*
            eb.ces_edges |
              ('AIC_EDGE_BASED_WIREFRAME.POLYLINE' IN
                  TYPEOF (edges\edge_curve.edge_geometry))) |
              NOT (SIZEOF (pline_edges\edge_curve.
                           edge_geometry\polyline.points) > 2))) = 0
      ))) = 0))) = 0;
  WR5: SIZEOF (QUERY (ebwm <* QUERY (it <* SELF.items |
          ('AIC_EDGE_BASED_WIREFRAME.EDGE_BASED_WIREFRAME_MODEL'
              IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (eb <* ebwm\edge_based_wireframe_model.
                                   ebwm_boundary |
           NOT (SIZEOF (QUERY (edges <* eb.ces_edges |
              NOT (('AIC_EDGE_BASED_WIREFRAME.VERTEX_POINT'
                       IN TYPEOF (edges.edge_start)) AND
                    ('AIC_EDGE_BASED_WIREFRAME.VERTEX_POINT'
                       IN TYPEOF (edges.edge_end))))) = 0
      ))) = 0))) = 0;
  WR6: SIZEOF (QUERY (ebwm <* QUERY (it <* SELF.items |
          ('AIC_EDGE_BASED_WIREFRAME.EDGE_BASED_WIREFRAME_MODEL'
               IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (eb <* ebwm\edge_based_wireframe_model.
                                   ebwm_boundary |
           NOT (SIZEOF (QUERY (edges <* eb.ces_edges |
             NOT (valid_wireframe_edge_curve
                    (edges\edge_curve.edge_geometry,
'AIC_EDGE_BASED_WIREFRAME')))) = 0))) = 0))) = 0;
  WR7: SIZEOF (QUERY (ebwm <* QUERY (it <* SELF.items|
          ('AIC_EDGE_BASED_WIREFRAME.EDGE_BASED_WIREFRAME_MODEL'
              IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (eb <* ebwm\edge_based_wireframe_model.
                                   ebwm_boundary |
           NOT (SIZEOF (QUERY (edges <* eb.ces_edges |
             NOT ((valid_wireframe_vertex_point
                    (edges.edge_start\vertex_point.vertex_geometry,
                     'AIC_EDGE_BASED_WIREFRAME')) AND
                   (valid_wireframe_vertex_point
                    (edges.edge_end\vertex_point.vertex_geometry,
                     'AIC_EDGE_BASED_WIREFRAME'))) )) = 0
          ))) = 0))) = 0;
  WR8: SIZEOF (QUERY (ebwm <* QUERY (it <* SELF.items |
          ('AIC_EDGE_BASED_WIREFRAME.EDGE_BASED_WIREFRAME_MODEL'
```

```
            IN TYPEOF (it))) |
        NOT (SIZEOF (QUERY (eb <* ebwm\edge_based_wireframe_model.
                                  ebwm_boundary |
          NOT (SIZEOF (QUERY (con_edges <* QUERY (edges <* eb.ces_edges |
             ('AIC_EDGE_BASED_WIREFRAME.CONIC'
                 IN TYPEOF (edges\edge_curve.edge_geometry))) |
           NOT ('AIC_EDGE_BASED_WIREFRAME.AXIS2_PLACEMENT_3D'
                  IN TYPEOF (con_edges\edge_curve.
                             edge_geometry\conic.position)))) = 0
      ))) = 0))) = 0;
  WR9: SIZEOF (QUERY (mi <* QUERY (it <* SELF.items |
         ('AIC_EDGE_BASED_WIREFRAME.MAPPED_ITEM' IN TYPEOF (it))) |
        NOT ('AIC_EDGE_BASED_WIREFRAME.' +
             'EDGE_BASED_WIREFRAME_SHAPE_REPRESENTATION'
          IN TYPEOF (mi\mapped_item.mapping_source.mapped_representation)
      ))) = 0;
  WR10: SELF.context_of_items\geometric_representation_context.
          coordinate_space_dimension = 3;
END_ENTITY;
(*
```

Formal propositions:

WR1: The items in an **edge_based_wireframe_shape_representation** shall be an **edge_based_wireframe_-model**, **mapped_item**, or **axis2_placement_3d**.

WR2: At least one of the items in an **edge_based_wireframe_shape_representation** shall be either an **edge_-based_wireframe_model** or a **mapped_item**.

WR3: Every edge in an **edge_based_wireframe_model** shall be an **edge_curve**.

WR4: Every polyline that underlies an edge of an **edge_based_wireframe_model** shall be defined by more than two distinct points.

WR5: Every vertex defined for an **edge_based_wireframe_model** shall be a **vertex_point**.

WR6: The **edge_geometry** that underlies an edge for an **edge_based_wireframe_model** shall be a **line**, **circle**, **ellipse**, **parabola**, **hyperbola**, **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.

WR7: The **vertex_geometry** that underlies the edges for an **edge_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: All conics shall be positioned by an **axis2_placement_3d**.

WR9: If there is a **mapped_item** in an **edge_based_wireframe_shape_representation**, the source of the **mapped_item** shall be an **edge_based_wireframe_shape_representation**.

WR10: The **edge_based_wireframe_shape_representation** shall have a **coordinate_space_dimension** equal to three.

F.1.2.2 aic_edge_based_wireframe function definitions

F.1.2.2.1 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;
                                     schma : STRING) : BOOLEAN;
  -- check for valid basic curve types

  IF SIZEOF ([schma + '.LINE',
              schma + '.CIRCLE',
              schma + '.ELLIPSE',
              schma + '.PARABOLA',
              schma + '.HYPERBOLA',
              schma + '.B_SPLINE_CURVE',
              schma + '.POLYLINE'] * TYPEOF (crv)) = 1
    THEN RETURN (TRUE);
  ELSE
  -- recursively check for valid basic curves for curve_replica
    IF (schma + '.CURVE_REPLICA') IN TYPEOF (crv)
      THEN RETURN (valid_wireframe_edge_curve
                        (crv\curve_replica.parent_curve, schma));
    ELSE
      -- recursively check for valid basis curves for offset_curve
      IF (schma + '.OFFSET_CURVE_3D') IN TYPEOF (crv)
        THEN RETURN (valid_wireframe_edge_curve
                          (crv\offset_curve_3d.basis_curve, schma));
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;
(*
```

Argument definitions:

crv: the input **curve** that is to be examined.

schma: the input string that defines the schema in which the **crv** is defined to be used for type checking.

F.1.2.2.2 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;
                                       schma : STRING) : BOOLEAN;
  -- check for valid basic point types
  IF (schma + '.CARTESIAN_POINT' IN TYPEOF (pnt))
    THEN RETURN (TRUE);
  ELSE
    -- recursively check for valid basic point types as parents for a
    -- point_replica

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

Argument definitions:

pnt: the input **point** that is to be examined.

schma: the input string that defines the schema in which the **pnt** is defined to be used for type checking.

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

F.2 Application interpreted construct 502: Shell-based wireframe

F.2.1 Scope

This application interpreted construct 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 AIC:

— 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 which 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 AIC:

— geometry defined in a two-dimensional coordinate space;

— surface geometry;

— solid geometry.

F.2.2 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 AIC.

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. References to eliminated constructs are outside the scope of the AIC.

```
*)
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_on_curve,
   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

F.2.2.1 aic_shell_based_wireframe entity definition: shell_based_wireframe_-shape_representation

A **shell_based_wireframe_shape_representation** is a three-dimensional **shape_representation** which represents the shape or a portion of the shape of a product by wireframes which define an implicit volume. This includes all 3D curves and topological entities which 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,
                       'AIC_SHELL_BASED_WIREFRAME')))) =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,
                         'AIC_SHELL_BASED_WIREFRAME'))
                 AND
                   (valid_wireframe_vertex_point
                  (el.edge_element.edge_end\vertex_point.vertex_geometry,
                         'AIC_SHELL_BASED_WIREFRAME')))
            )) = 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 (eloop <* QUERY (wsb <*
                              ws\wire_shell.wire_shell_extent |
            ('AIC_SHELL_BASED_WIREFRAME.EDGE_LOOP' IN TYPEOF (wsb))) |
            NOT (SIZEOF (QUERY (con_edges <* QUERY (el <*
                                eloop\path.edge_list |
              ('AIC_SHELL_BASED_WIREFRAME.CONIC' IN
                  TYPEOF (el.edge_element\edge_curve.edge_geometry))) |
              NOT ('AIC_SHELL_BASED_WIREFRAME.AXIS2_PLACEMENT_3D' IN
                   TYPEOF (con_edges.edge_element\edge_curve.
```

```
                          edge_geometry\conic.position))
            )) = 0)
          )) = 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 ('AIC_SHELL_BASED_WIREFRAME.VERTEX_POINT' IN
                    TYPEOF (vloop\vertex_loop.loop_vertex))
           )) = 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 (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,
                    'AIC_SHELL_BASED_WIREFRAME'))
           )) = 0)
         )) = 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 ('AIC_SHELL_BASED_WIREFRAME.VERTEX_POINT' IN
                  TYPEOF (vs\vertex_shell.vertex_shell_extent.loop_vertex))
         )) = 0)
       )) = 0;
  WR12: 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,'AIC_SHELL_BASED_WIREFRAME'))
         )) = 0)
       )) = 0;
  WR13: 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;
  WR14: 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 points.

WR5: The **edge_geometry** that underlies an edge for a **shell_based_wireframe_model** shall be a **line**, **circle**, **ellipse**, **parabola**, **hyperbola**, **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.

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 edges 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 position of a conic that underlies an edge in an **edge_loop** for a **shell_based_wireframe_model** shall only be an **axis2_placement_3d**.

WR9: The vertex that defines the **vertex_loop** which is used as a bound in a **shell_based_wireframe_model** shall be a **vertex_point**.

WR10: The vertex that defines the **vertex_loop** which is used as a bound 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**.

WR11: The vertex that defines the **vertex_loop** which is used as the **vertex_shell_extent** for a **vertex_shell**

in a **shell_based_wireframe_model** shall be a **vertex_point**.

WR12: The vertex that defines the **vertex_loop** which 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**.

WR13: 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**.

WR14: The **shell_based_wireframe_shape_representation** shall have **coordinate_space_dimension** equal to three.

F.2.2.2 aic_shell_based_wireframe function definitions

F.2.2.2.1 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;
                                     schma : STRING) : BOOLEAN;
  -- check for valid basic curve types

  IF SIZEOF ([schma + '.LINE',
              schma + '.CIRCLE',
              schma + '.ELLIPSE',
              schma + '.PARABOLA',
              schma + '.HYPERBOLA',
              schma + '.B_SPLINE_CURVE',
              schma + '.POLYLINE'] * TYPEOF (crv)) = 1
    THEN RETURN (TRUE);
  ELSE
  -- recursively check for valid basic curves for curve_replica
    IF (schma + '.CURVE_REPLICA') IN TYPEOF (crv)
      THEN RETURN (valid_wireframe_edge_curve
                        (crv\curve_replica.parent_curve, schma));
    ELSE
      -- recursively check for valid basis curves for offset_curve
      IF (schma + '.OFFSET_CURVE_3D') IN TYPEOF (crv)
        THEN RETURN (valid_wireframe_edge_curve
                          (crv\offset_curve_3d.basis_curve, schma));
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;
```

```
(*
```

Argument definitions:

crv: the input **curve** that is to be examined.

schma: the input string that defines the schema in which the **crv** is defined to be used for type checking.

F.2.2.2.2 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;
                                       schma : STRING) : BOOLEAN;
  -- check for valid basic point types
  IF (schma + '.CARTESIAN_POINT' IN TYPEOF (pnt))
    THEN RETURN (TRUE);
  ELSE
    -- recursively check for valid basic point types as parents for a
    -- point_replica

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

Argument definitions:

pnt: the input **point** that is to be examined.

schma: the input string that defines the schema in which the **pnt** is defined to be used for type checking.

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

F.3 Application interpreted construct 507: Geometrically bounded surface

F.3.1 Scope

This application interpreted construct specifies the interpretation of the integrated resources to satisfy requirements for the description of a geometric shape by means of geometrically bounded surfaces.

The following are within the scope of this AIC:

— 3D points;

— points defined in the parameter space of curves or surfaces;

— 3D curves;

— curves defined in the parameter space of surfaces;

NOTE - Such curves are also known as **pcurve**s or cons.

— the elementary curves line, circle, ellipse, parabola, and hyperbola;

— intersection curves;

— polylines;

— the elementary surfaces plane, cylinder, cone, torus, and sphere;

— swept surfaces created by rotation or linear extrusion of a curve;

— sculptured curves and surfaces;

— trimming of curves and surfaces;

— composition of curves and surfaces;

— replication of curves, surfaces, and surface models;

— 3D offsets of curves and surfaces.

The following are outside the scope of this AIC:

— unbounded geometry;

— self-intersecting geometry;

— geometry in a 2D cartesian coordinate space;

— topological entities.

F.3.2 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 AIC.

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. References to eliminated constructs are outside the scope of the AIC.

This application interpreted construct provides a consistent set of geometric entities for the definition of surface models that consist of points, elementary or sculptured curves, and elementary or sculptured surfaces. Geometry shall be bounded; no topological entities are used for bounding.

The highest level entity of this AIC is **geometrically_bounded_surface_shape_representation**. It is a **shape_representation** (see ISO 10303-41) consisting of **geometric_set**s. **Geometric_set**s shall be of dimensionality 3. **Point**s, **curve**s and **surface**s may be contained in **geometric_set**s provided they are of the same dimensionality (see ISO 10303-42, rule **compatible_dimension**). **Pcurve**s (see ISO 10303-42) may also be referenced. All geometric entities are of dimensionality 3. Two-dimensional geometry is only used for the purpose of defining **pcurve**s. The use of one-dimensional **cartesian_points** is excluded.

```
*)
SCHEMA aic_geometrically_bounded_surface;

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

USE FROM geometry_schema                                        -- ISO 10303-42
   (point,
    cartesian_point,
    point_on_curve,
    point_on_surface,
    degenerate_pcurve,
    evaluated_degenerate_pcurve,
    direction,
    vector,
    axis1_placement,
    axis2_placement_2d,
    axis2_placement_3d,
    cartesian_transformation_operator_3d,
    curve,
    line,
    circle,
    ellipse,
    hyperbola,
    parabola,
    polyline,
    b_spline_curve,
    b_spline_curve_with_knots,
    uniform_curve,
```

```
    quasi_uniform_curve,
    bezier_curve,
    rational_b_spline_curve,
    trimmed_curve,
    composite_curve,
    composite_curve_segment,
    reparametrised_composite_curve_segment,
    pcurve,
    surface_curve,
    intersection_curve,
    seam_curve,
    composite_curve_on_surface,
    offset_curve_3d,
    curve_replica,
    surface,
    plane,
    cylindrical_surface,
    conical_surface,
    spherical_surface,
    toroidal_surface,
    degenerate_toroidal_surface,
    swept_surface,
    surface_of_linear_extrusion,
    surface_of_revolution,
    b_spline_surface,
    b_spline_surface_with_knots,
    uniform_surface,
    quasi_uniform_surface,
    bezier_surface,
    rational_b_spline_surface,
    rectangular_trimmed_surface,
    curve_bounded_surface,
    boundary_curve,
    outer_boundary_curve,
    rectangular_composite_surface,
    surface_patch,
    offset_surface,
    surface_replica);

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

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

(*
```

NOTES

2 - The entities **b_spline_curve**, **b_spline_surface**, and **swept_surface** are explicitly interfaced (i.e. included in the USE FROM lists) to allow the **gbsf_check_curve** and **gbsf_check_surface** functions to access attributes of these entities.

3 - 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
representation_schema	ISO 10303-43

F.3.2.1 aic_geometrically_bounded_surface entity definition: geometrically_bounded_surface_shape_representation

A **geometrically_bounded_surface_shape_representation** is a **shape_representation** representing the shape or portions of the shape of a product using surface geometry without topology. Entity **product** is not part of this AIC.

A **geometrically_bounded_surface_shape_representation** consists of **point**s, **curve**s and **surface**s only. All unbounded **curve**s and **surface**s are explicitly trimmed. The boundaries of **curve**s are defined by **point**s on the **curve**s and explicit associations between the **point**s and the **curve**s which are bounded by them, or by parameter values. The boundaries of **surface**s are defined by **curve**s on the **surface**s and explicit associations between the **curve**s and the **surface**s which are bounded by them, or parameter values.

A **geometrically_bounded_surface_shape_representation** is represented by one or several **geometric_set**s. Each **geometric_set** contains only those **point**s, **curve**s, and **surface**s that define the physical object that is being represented by the particular instance of the **geometrically_bounded_surface_shape_representation**. The geometric entities that are used to support the definition of another geometric entity shall themselves not be among the elements of a **geometric_set**.

The items of a **geometrically_bounded_surface_shape_representation** may also be of type **mapped_item** or **axis2_placement_3d**. This enables the representation of assemblies of **geometrically_bounded_surface_shape_representation**s.

The WHERE rules below restrict the use of the entity data types of this AIC. To be able to apply these restrictions to each instance of a given model this model needs to be traced for all instances of a requested entity data type. The structure of references within this AIC is not top-down, but like a network. Some **curve**s may reference both other **curve**s and **surface**s as is the case for **pcurve**. **Point**s may reference **curve**s or **surface**s. This network can be checked for valid references using three functions which all are based on recursion:

— gbsf_check_point;

— gbsf_check_curve;

— gbsf_check_surface.

Each of these functions branch within its category into all the entity types that are within the scope of this AIC.

Any references from any of the types are further investigated by calling one of the three functions, possibly recursively. **Point**s, **curve**s, and **surface**s are not only checked for referential integrity, but also for other applicable constraints.

The functions shall be applied to all the elements of all **geometric_set**s in a **geometrically_bounded_surface_-shape_representation**.

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

EXPRESS specification:

```
*)
ENTITY geometrically_bounded_surface_shape_representation
   SUBTYPE OF (shape_representation);
WHERE
  WR1 : SIZEOF (QUERY (it <* SELF.items |
        NOT (SIZEOF (['AIC_GEOMETRICALLY_BOUNDED_SURFACE.GEOMETRIC_SET',
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.MAPPED_ITEM',
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.AXIS2_PLACEMENT_3D'] * TYPEOF
        (it)) = 1))) = 0;
  WR2 : SIZEOF (QUERY (it <* SELF.items |
        SIZEOF (['AIC_GEOMETRICALLY_BOUNDED_SURFACE.GEOMETRIC_SET',
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.MAPPED_ITEM'] * TYPEOF
        (it)) = 1)) > 0;
  WR3 : SIZEOF (QUERY (mi <* QUERY (it <* SELF.items |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.MAPPED_ITEM' IN TYPEOF (it)) |
        NOT ('AIC_GEOMETRICALLY_BOUNDED_SURFACE.' +
        'GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION'
        IN TYPEOF (mi\mapped_item.mapping_source.mapped_representation))))
        = 0;
  WR4 : SIZEOF (QUERY (gs <* QUERY (it <* SELF.items |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.GEOMETRIC_SET' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (pnt <* QUERY (gsel <*
        gs\geometric_set.elements |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.POINT' IN TYPEOF (gsel)) |
        NOT (gbsf_check_point(pnt)))) = 0))) = 0;
  WR5 : SIZEOF (QUERY (gs <* QUERY (it <* SELF.items |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.GEOMETRIC_SET' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (cv <* QUERY (gsel <*
        gs\geometric_set.elements |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.CURVE' IN TYPEOF (gsel)) |
        NOT (gbsf_check_curve(cv)))) = 0))) = 0;
  WR6 : SIZEOF (QUERY (gs <* QUERY (it <* SELF.items |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.GEOMETRIC_SET' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (sf <* QUERY (gsel <*
        gs\geometric_set.elements |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.SURFACE' IN TYPEOF (gsel)) |
        NOT (gbsf_check_surface(sf)))) = 0))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: The **items** in a **geometrically_bounded_surface_shape_representation** shall be **geometric_set**s, **mapped_item**s, or **axis2_placement_3d**s.

NOTE 1 - **Axis2_placement_3d** is valid as a reference as **mapping_target** from a **mapped_item**. To include another representation into the list of **items** of a **geometrically_bounded_surface_shape_representation** (see WR3 for valid **mapped_items**), the **mapped_item.mapping_source.mapping_origin** may be any entity that is geometrically founded in the **geometric_representation_context** of the **mapped_representation**. If this entity is an **axis2_placement_3d**, the mapping operator for the **mapped_representation** to the **geometrically_bounded_surface_shape_representation** corresponds to a transformation matrix with only translation and rotation enabled. If a **cartesian_transformation_operator_3d** is used as **mapping_origin**, scaling and mirroring are possible.

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

WR3: If there is a **mapped_item** in a **geometrically_bounded_surface_shape_representation**, the **mapped_representation** of its **mapping_source** shall be a **geometrically_bounded_surface_shape_representation**.

WR4: Each **point** that is among the elements of a **geometric_set** that is one of the **items** of a **geometrically_bounded_surface_shape_representation** shall be a valid **point**.

WR5: Each **curve** that is among the elements of a **geometric_set** that is one of the **items** of a **geometrically_bounded_surface_shape_representation** shall be a valid **curve**.

WR6: Each **surface** that is among the elements of a **geometric_set** that is one of the **items** of a **geometrically_bounded_surface_shape_representation** shall be a valid **surface**.

Informal propositions:

IP1: A **b_spline_curve** shall not self-intersect.

IP2: A **composite_curve** shall not self-intersect.

IP3: An **offset_curve_3d** shall not self-intersect.

IP4: A **b_spline_surface** shall not self-intersect.

IP5: An **offset_surface** shall not self-intersect.

IP6: The geometric entities that are exclusively used to support the definition of other geometric entities shall not themselves exist in the sets of elements of a **geometric_set**.

F.3.2.2 aic_geometrically_bounded_surface function definitions

The EXPRESS language has a number of built in functions. This section describes additional functions required for the definition and constraints on the **aic_geometrically_bounded_surface** schema. These functions are used in the specification of the entity **geometrically_bounded_surface_shape_representation**.

F.3.2.2.1 gbsf_check_point

The **gbsf_check_point** function performs type checking for **point**s. An instance is checked against all types of points that are valid in the context of a **geometrically_bounded_surface_shape_representation**. All related geometry, that is curves and surfaces, are correspondingly checked. When curves or surfaces are referenced, the functions **gbsf_check_curve** and **gbsf_check_surface**, respectively, are called. The recursion within these functions terminates at entity types that do not reference any points, curves, or surfaces.

The following point types are within the scope of the **geometrically_bounded_surface_shape_representation** and are valid input to this function:

— cartesian_point;

— degenerate_pcurve;

— point_on_curve;

— point_on_surface.

The three latter ones in the list reference either curves or surfaces or both.

The **basis_surface** of a **degenerate_pcurve** may be any of the valid surfaces in a **geometrically_bounded_surface_shape_representation**. The **reference_to_curve** of a **degenerate_pcurve** shall be of one of the following types:

— b_spline_curve;

— composite_curve (recursive);

— conic;

— curve_replica (recursive);

— line;

— polyline;

— trimmed_curve (recursive).

The **basis_curve** of a **point_on_curve** may be any of the valid curves in a **geometrically_bounded_surface_-shape_representation**.

The **basis_surface** of a **point_on_surface** may be any of the valid surfaces in a **geometrically_- bounded_-surface_shape_representation**.

This function returns TRUE if the types of all referenced geometries are within the scope of the **geometrically_bounded_surface_shape_representation**; else it returns FALSE.

NOTES

1 - This function does not check the correctness of references with respect to ISO 10303-42. Only requirements due to the scope of the **geometrically_bounded_surface_shape_representation** are checked.

2 - The function uses recursion to trace entity references.

EXPRESS specification:

```
*)
FUNCTION gbsf_check_point (pnt : point) : BOOLEAN;

  -- check whether the input has the right type;
  -- a cartesian_point is valid and has no further references

  IF 'GEOMETRY_SCHEMA.CARTESIAN_POINT' IN TYPEOF (pnt) THEN
    RETURN(TRUE);
  ELSE

  -- a point_on_curve needs to be checked for the validity of its curve;
  -- further references down the tree are taken care of by the function
  -- gbsf_check_curve

    IF 'GEOMETRY_SCHEMA.POINT_ON_CURVE' IN TYPEOF (pnt) THEN
      RETURN (gbsf_check_curve
      (pnt\point_on_curve.basis_curve));
    ELSE

    -- a point_on_surface needs to be checked for the validity of its surface;
    -- further references down the tree are taken care of by the function
    -- gbsf_check_surface

      IF 'GEOMETRY_SCHEMA.POINT_ON_SURFACE' IN TYPEOF (pnt) THEN
        RETURN(gbsf_check_surface
        (pnt\point_on_surface.basis_surface));
      ELSE

      -- a degenerate_pcurve needs to be checked for the validity of its
      -- defining geometry; further references down the tree are taken care
      -- of by the functions gbsf_check_curve and gbsf_check_surface;
```

```
      -- both return true for valid points

        IF 'GEOMETRY_SCHEMA.DEGENERATE_PCURVE' IN TYPEOF (pnt) THEN
          RETURN
          ((gbsf_check_curve
          (pnt\degenerate_pcurve.reference_to_curve.items[1]))
          AND (gbsf_check_surface (pnt\degenerate_pcurve.basis_surface)));
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION;
(*
```

Argument definitions:

pnt: (input) the **point** that is being checked for being a valid **point** in a **geometrically_bounded_surface_shape_representation**.

BOOLEAN: (return) is TRUE if the **point** is a valid point; else FALSE.

F.3.2.2.2 gbsf_check_curve

The **gbsf_check_curve** function performs type and constraint checking for curves. An instance is checked against all types of curves that are valid in the context of a **geometrically_bounded_surface_shape_representation**. All related geometry, that is curves and surfaces, are correspondingly checked.

EXAMPLE - Whether the instance of a **b_spline_curve** self-intersects, is an example of a constraint that is validated by this function.

Where appropriate an instance is investigated recursively. This means if a curve references another curve as a basis curve or parent curve, the **gbsf_check_curve** function is called again. If a surface is referenced, the **gbsf_check_surface** function is called. The recursion terminates at entity types that do not reference any curves or surfaces.

The following curve types are within the scope of the **geometrically_bounded_surface_shape_representation** and are valid input to this function:

— b_spline_curve;

— composite_curve;

— conic;

— curve_replica;

— line;

— offset_curve_3d;

— pcurve;

— polyline;

— surface_curve;

— trimmed_curve.

The segments of a **composite_curve** shall be of type **composite_curve_segment** only. The **parent_curve** of a **composite_curve_segment** shall be of one of the following types:

— b_spline_curve;

— composite_curve (recursive);

— polyline;

— surface_curve (recursive);

— trimmed_curve (recursive).

The **parent_curve** of a **curve_replica**, the **basis_curve** of an **offset_curve_3d**, and the **basis_curve** of a **trimmed_curve** shall all be of one of the following types:

— b_spline_curve;

— composite_curve (recursive);

— conic;

— curve_replica (recursive);

— line;

— offset_curve_3d (recursive);

— pcurve (recursive);

— polyline;

— surface_curve (recursive);

— trimmed_curve (recursive).

The one instance in the set of **items** of a **definitional_representation** that is referenced as **reference_to_curve** by a **pcurve** shall be of one of the following types:

— b_spline_curve;

— composite_curve (recursive);

— conic;

— curve_replica (recursive);

— line;

— polyline;

— trimmed_curve (recursive).

The **curve_3d** of a **surface_curve** shall be of one of the following types:

— b_spline_curve;

— composite_curve (recursive);

— conic;

— curve_replica (recursive);

— line;

— offset_curve_3d (recursive);

— polyline;

— surface_curve (recursive);

— trimmed_curve (recursive).

This function returns TRUE if the types of all referenced geometries are within the scope of the **geometrically_bounded_surface_shape_representation** and if all constraints are satisfied; else it returns

FALSE.

NOTES

1 - This function does not check the correctness of references with respect to ISO 10303-42. Only requirements due to the scope of the **geometrically_bounded_surface_shape_representation** are checked.

2 - The function uses recursion to trace entity references.

EXPRESS specification:

```
*)
FUNCTION gbsf_check_curve (cv : curve) : BOOLEAN;

  -- check whether the input has the right type;

  -- let those types pass that do not have any further references
  -- respectively rules to be applied; ensure that complex entities
  -- with bounded_curve do not pass

  IF SIZEOF (['GEOMETRY_SCHEMA.BOUNDED_CURVE', 'GEOMETRY_SCHEMA.CIRCLE',
    'GEOMETRY_SCHEMA.ELLIPSE'] * TYPEOF(cv)) = 1 THEN RETURN(TRUE);
  ELSE

    -- the b_spline_curve shall not self intersect

    IF (('GEOMETRY_SCHEMA.B_SPLINE_CURVE' IN TYPEOF(cv)) AND
      (cv\b_spline_curve.self_intersect = FALSE))
    THEN RETURN(TRUE);
    ELSE

      -- if the curve is a composite_curve, all of its segments shall be valid

      IF (('GEOMETRY_SCHEMA.COMPOSITE_CURVE' IN TYPEOF(cv)) AND
        (cv\composite_curve.self_intersect = FALSE)) THEN
        RETURN (SIZEOF (QUERY (seg <* cv\composite_curve.segments |
        NOT (gbsf_check_curve(seg.parent_curve)))) = 0);
      ELSE

        -- the curve_replica references other curves that need to be checked

        IF SIZEOF (['GEOMETRY_SCHEMA.BOUNDED_CURVE',
          'GEOMETRY_SCHEMA.CURVE_REPLICA'] * TYPEOF(cv)) = 1 THEN
          RETURN (gbsf_check_curve (cv\curve_replica.parent_curve));
        ELSE

          -- offset_curve_3d references a curve and shall not self intersect

          IF ((SIZEOF (['GEOMETRY_SCHEMA.BOUNDED_CURVE',
            'GEOMETRY_SCHEMA.OFFSET_CURVE_3D'] * TYPEOF(cv)) = 1) AND
            (cv\offset_curve_3d.self_intersect = FALSE)) THEN
            RETURN (gbsf_check_curve (cv\offset_curve_3d.basis_curve));
          ELSE
```

```
-- pcurve references a curve - indirectly, and a basis_surface

IF SIZEOF (['GEOMETRY_SCHEMA.BOUNDED_CURVE',
  'GEOMETRY_SCHEMA.PCURVE'] * TYPEOF(cv)) = 1 THEN
  RETURN ((gbsf_check_curve
  (cv\pcurve.reference_to_curve.items[1])) AND
  (gbsf_check_surface (cv\pcurve.basis_surface)));
ELSE

  -- polyline shall have at least 3 points and shall only
  -- be used to represent an intersection_curve

  IF 'GEOMETRY_SCHEMA.POLYLINE' IN TYPEOF(cv) THEN
    IF (SIZEOF (cv\polyline.points) >= 3)  AND
      (SIZEOF ((bag_to_set (USEDIN (cv,'')) -
      bag_to_set (USEDIN (cv,
      'GEOMETRY_SCHEMA.INTERSECTION_CURVE.BASIS_CURVE')))) = 0)
      THEN RETURN (TRUE);
    END_IF;
  ELSE

    -- surface_curve references a curve_3d and one or two
    -- pcurves or one or two surface_curves or one of each

    IF SIZEOF (['GEOMETRY_SCHEMA.BOUNDED_CURVE',
      'GEOMETRY_SCHEMA.SURFACE_CURVE'] * TYPEOF(cv)) = 1 THEN

      -- if the curve reference is correct, check also the rest

      IF gbsf_check_curve (cv\surface_curve.curve_3d) THEN
        REPEAT i := 1 TO SIZEOF
        (cv\surface_curve.associated_geometry);

        -- do for one or two associated_geometrys:

        IF 'GEOMETRY_SCHEMA.SURFACE' IN TYPEOF
          (cv\surface_curve.associated_geometry[i]) THEN
          IF NOT gbsf_check_surface
            (cv\surface_curve.associated_geometry[i]) THEN
            RETURN(FALSE);
          END_IF;
        ELSE
          IF 'GEOMETRY_SCHEMA.PCURVE' IN TYPEOF
            (cv\surface_curve.associated_geometry[i]) THEN
            IF NOT gbsf_check_curve
              (cv\surface_curve.associated_geometry[i]) THEN
              RETURN(FALSE);
            END_IF;
          END_IF;
        END_IF;
      END_REPEAT;
      RETURN(TRUE);
    END_IF;
  ELSE

    -- if the curve is a trimmed_curve
```

```
                IF ('GEOMETRY_SCHEMA.TRIMMED_CURVE') IN TYPEOF (cv) THEN

                -- if a line, parabola or hyperbola is being trimmed,
                -- then valid

                  IF SIZEOF (['GEOMETRY_SCHEMA.BOUNDED_CURVE',
                  'GEOMETRY_SCHEMA.LINE',
                  'GEOMETRY_SCHEMA.PARABOLA',
                  'GEOMETRY_SCHEMA.HYPERBOLA'] *
                  TYPEOF (cv\trimmed_curve.basis_curve)) = 1
                  THEN RETURN (TRUE);

                    -- otherwise, recursively check the basis curve

                    ELSE RETURN (gbsf_check_curve
                      (cv\trimmed_curve.basis_curve));
                    END_IF;
                  END_IF;
                END_IF;
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION;
(*
```

Argument definitions:

cv: (input) the **curve** that is being checked for a valid **curve** in a **geometrically_bounded_surface_shape_-representation**.

BOOLEAN: (return) is TRUE if the **curve** is a valid **curve**; else FALSE.

F.3.2.2.3 gbsf_check_surface

The **gbsf_check_surface** function performs type and constraint checking for surfaces. An instance is checked against all types of surfaces that are valid in the context of a **geometrically_bounded_surface_shape_-representation**. All related geometry, that is curves and surfaces, are correspondingly checked.

> EXAMPLE - Whether the instance of a **b_spline_surface** self-intersects, is an example of a constraint that is validated by this function.

Where appropriate an instance is investigated recursively. This means if a surface references another surface as a basis surface or parent surface, the **gbsf_check_surface** function is called again. If a curve is referenced, the **gbsf_check_curve** function is called. The recursion terminates at entity types that do not reference any

curves or surfaces.

The following surface types are within the scope of the **geometrically_bounded_surface_shape_representation** and are valid input to this function:

— b_spline_surface;

— curve_bounded_surface;

— elementary_surface;

— offset_surface;

— rectangular_composite_surface;

— rectangular_trimmed_surface;

— surface_replica;

— swept_surface.

Both the **basis_surface** of a **curve_bounded_surface**, the **basis_surface** of an **offset_surface**, the **basis_surface** of a **rectangular_trimmed_surface**, and the **parent_surface** of a **surface_replica** shall all be one of the surface types listed above.

The segments of a **rectangular_composite_surface** shall be **surface_patch**es. The **parent_surface** of a **surface_patch** shall be of one of the following types:

— b_spline_surface;

— rectangular_composite_surface (recursive);

— rectangular_trimmed_surface (recursive).

Swept_surfaces reference curves. Function **gbsf_check_curve** is called for type checking. The **geometrically_bounded_surface_shape_representation** requires the same constraints on valid sweeping curves as specified in ISO 10303-42. All curves that are in the scope of a **geometrically_bounded_surface_shape_representation** are valid **swept_curve**s. This function returns TRUE if the types of all referenced geometries are within the scope of the **geometrically_bounded_surface_shape_representation** and if all constraints are satisfied; else it returns FALSE.

NOTES

1 - This function does not check the correctness of references with respect to ISO 10303-42. Only requirements due to the scope of the **geometrically_bounded_surface_shape_representation** are checked.

2 - The function uses recursion to trace entity references.

EXPRESS specification:

```
*)
FUNCTION gbsf_check_surface (sf : surface) : BOOLEAN;

  -- check whether the input has the right type and for some whether
  -- attribute restrictions are fulfilled (self-intersect e.g.)

  -- b_spline_surface has a self_intersect attribute that shall be false

  IF (('GEOMETRY_SCHEMA.B_SPLINE_SURFACE' IN TYPEOF(sf)) AND
    (sf\b_spline_surface.self_intersect = FALSE)) THEN
    RETURN(TRUE);
  ELSE

    -- basis surface types return true

    IF SIZEOF (['GEOMETRY_SCHEMA.SPHERICAL_SURFACE',
      'GEOMETRY_SCHEMA.TOROIDAL_SURFACE'] * TYPEOF(sf)) = 1 THEN
      RETURN(TRUE);
    ELSE

      IF 'GEOMETRY_SCHEMA.CURVE_BOUNDED_SURFACE' IN TYPEOF(sf) THEN

      -- if there is a simple basis surface, check the curves

        IF SIZEOF (['GEOMETRY_SCHEMA.CONICAL_SURFACE',
          'GEOMETRY_SCHEMA.CYLINDRICAL_SURFACE',
          'GEOMETRY_SCHEMA.PLANE']
          * TYPEOF(sf\curve_bounded_surface.basis_surface)) = 1 THEN
          RETURN(SIZEOF (QUERY (bcurve <*
          sf\curve_bounded_surface.boundaries |
          NOT (gbsf_check_curve(bcurve)))) = 0);

        ELSE

        -- recursively check the basis_surface and then the curves

          IF gbsf_check_surface
          (sf\curve_bounded_surface.basis_surface) THEN
            RETURN(SIZEOF (QUERY (bcurve <*
            sf\curve_bounded_surface.boundaries |
            NOT (gbsf_check_curve(bcurve)))) = 0);
          END_IF;
        END_IF;
      ELSE

        -- offset_surface references a surface and shall not self intersect

        IF (('GEOMETRY_SCHEMA.OFFSET_SURFACE' IN TYPEOF(sf)) AND
```

```
        (sf\offset_surface.self_intersect = FALSE)) THEN
        RETURN (gbsf_check_surface (sf\offset_surface.basis_surface));
      ELSE

        -- rectangular_composite_surface references a matrix of surfaces

        IF 'GEOMETRY_SCHEMA.RECTANGULAR_COMPOSITE_SURFACE' IN TYPEOF(sf)
        THEN
          REPEAT i := 1 TO SIZEOF
            (sf\rectangular_composite_surface.segments);
            REPEAT j := 1 TO SIZEOF
              (sf\rectangular_composite_surface.segments[i]);
              IF NOT (gbsf_check_surface
                (sf\rectangular_composite_surface.segments[i][j].
                parent_surface))
                THEN RETURN (FALSE);
              END_IF;
            END_REPEAT;
          END_REPEAT;
          RETURN(TRUE);
        ELSE

          -- rectangular_trimmed_surface has a basis_surface

          IF 'GEOMETRY_SCHEMA.RECTANGULAR_TRIMMED_SURFACE' IN TYPEOF(sf)
          THEN
            IF  SIZEOF (['GEOMETRY_SCHEMA.CONICAL_SURFACE',
              'GEOMETRY_SCHEMA.CYLINDRICAL_SURFACE',
              'GEOMETRY_SCHEMA.PLANE'] * TYPEOF
              (sf\rectangular_trimmed_surface.basis_surface)) = 1 THEN
              RETURN (TRUE);
            ELSE
              RETURN(gbsf_check_surface
              (sf\rectangular_trimmed_surface.basis_surface));
            END_IF;
          ELSE

            -- parent_surfaces need to be checked for their validity

            IF 'GEOMETRY_SCHEMA.SURFACE_REPLICA' IN TYPEOF (sf) THEN
              RETURN (gbsf_check_surface
              (sf\surface_replica.parent_surface));
            ELSE

              -- and the swept_surface must have a valid swept_curve

              IF 'GEOMETRY_SCHEMA.SWEPT_SURFACE' IN TYPEOF(sf) THEN
                RETURN(gbsf_check_curve (sf\swept_surface.swept_curve));
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
END_IF;
```

```
  RETURN(FALSE);
END_FUNCTION;
(*
```

Argument definitions:

sf:(input) the **surface** that is being checked for a valid **surface** in a **geometrically_bounded_surface_shape_representation**.

BOOLEAN: (return) is TRUE if the **surface** is a valid **surface**; else FALSE.

F.3.2.2.4 bag_to_set

This function converts BAGs into SETs.

EXAMPLE - It can be used to convert the BAGs returned by the USEDIN function into SETs that can be properly assigned to variables that are SETs.

NOTE - This function specification is copied from ISO 10303-41. It is included in this AIC to resolve all references from this AIC schema. The function is unchanged.

EXPRESS specification:

```
*)
FUNCTION bag_to_set (the_bag : BAG OF GENERIC : intype) :
                      SET OF GENERIC : intype;

  LOCAL
    the_set: SET OF GENERIC : intype := [];
    i      : INTEGER;
  END_LOCAL;

  IF SIZEOF (the_bag) > 0 THEN
    REPEAT i := 1 to HIINDEX (the_bag);
      the_set := the_set + the_bag [i];
    END_REPEAT;
  END_IF;

  RETURN (the_set);

END_FUNCTION;
(*
```

Argument definitions:

the_bag: the BAG that is to be converted into a SET.

*)

END_SCHEMA; -- aic_geometrically_bounded_surface
(*

F.4 Application interpreted construct 509: Manifold surface

F.4.1 Scope

This application interpreted construct specifies the interpretation of the integrated resources to satisfy requirements for the description of a geometric shape by means of manifold surfaces.

The following are within the scope of this AIC:

— 3D points;

— points defined in the parameter space of curves or surfaces;

— 3D curves;

— curves defined in the parameter space of surfaces;

NOTE - Such curves are also known as **pcurve**s or coons.

— the elementary curves line, circle, ellipse, parabola, and hyperbola;

— intersection curves;

— surfaces;

— the elementary surfaces plane, cylinder, cone, torus, and sphere;

— swept surfaces created by rotation or linear extrusion of a curve;

— sculptured curves and surfaces;

— trimming of curves and surfaces using topological entities;

— composition of curves and surfaces using topological entities;

— replication of curves, surfaces, and surface models;

— 3D offsets of curves and surfaces;

— 2-manifolds.

The following are outside the scope of this AIC:

— unbounded geometry;

— self—intersecting geometry;

— geometry in a 2D cartesian coordinate space;

— topology that is defined independent of geometry;

— non-manifolds.

F.4.2 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 AIC.

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. References to eliminated constructs are outside the scope of the AIC.

This application interpreted construct provides a consistent set of geometric and topological entities for the definition of manifold surface representations that consist elementary or sculptured surfaces. The highest level entity of this AIC is **manifold_surface_shape_representation**. A **manifold_surface_shape_representation** is bounded. The bounding of the geometry is achieved by topological entities, such as **edge** and **face**.

A **manifold_surface_shape_representation** is capable of representing 2-manifolds. A maximum of two faces may share the same edge. A **manifold_surface_shape_representation** is a **shape_representation** (see ISO 10303-41) that consists of a set of **shell_based_surface_model**s. A **shell_based_surface_model** references topological entities which in turn reference geometric entities such as **point**s, **curve**s, and **surface**s. **Pcurves** (see ISO 10303-42) may also be referenced. All geometric entities are of dimensionality 3. Two-dimensional geometry is only used for the purpose of defining **pcurve**s. The use of one-dimensional **cartesian_point**s is excluded.

The assembly of **manifold_surface_representation**s is provided by the use of **mapped_item**s as members of **manifold_surface_shape_representation_item**s. **Mapped_item** is specified in ISO 10303-43.

Topology shall not exist independent of geometry.

```
*)
SCHEMA aic_manifold_surface;

  USE FROM aic_topologically_bounded_surface;                    -- Annex F
```

```
  USE FROM geometric_model_schema (shell_based_surface_model); -- ISO 10303-42

  USE FROM geometry_schema (                                   -- ISO 10303-42
      point_on_curve,
      point_on_surface,
      degenerate_pcurve,
      evaluated_degenerate_pcurve,
      axis2_placement_2d,
      cartesian_transformation_operator_3d,
      curve,
      intersection_curve,
      seam_curve,
      offset_curve_3d,
      curve_replica,
      surface,
      offset_surface,
      surface_replica);

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

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

  USE FROM topology_schema (                                   -- ISO 10303-42
      oriented_face,
      connected_face_set,
      open_shell,
      closed_shell);

(*
```

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

aic_topologically_bounded_surface	Annex F
geometric_model_schema	ISO 10303-42
geometry_schema	ISO 10303-42
product_property_representation_schema	ISO 10303-41
representation_schema	ISO 10303-43
topology_schema	ISO 10303-42

F.4.2.1 aic_manifold_surface entity definition: manifold_surface_shape_representation

A **manifold_surface_representation** is a **shape_representation** which represents the shape or portions of the shape of a product using 2-manifolds with boundaries. 2-manifolds are topologically constrained in a way that makes them suitable for inclusion into solid representations. A **manifold_surface_shape_representation** consists of a set of shells, each of which consists of a set of faces. These use among others edges and vertices that in turn reference geometric entities. **Open_shell** and **closed_shell**, which both are subtypes of **connected_face_set**, shall be used. **Connected_face_set** shall not be instantiated.

Topological entities shall not exist without an association to a corresponding geometric domain.

The **items** of a **manifold_surface_shape_representation** may also be of type **mapped_item** or **axis2_-placement_3d**. This enables the representation of assemblies of **manifold_surface_shape_representation**s.

All unbounded geometry shall be trimmed by using topological constructs.

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

EXPRESS specification:

```
*)
ENTITY manifold_surface_shape_representation
   SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (it <* SELF.items |
       NOT (SIZEOF (['AIC_MANIFOLD_SURFACE.SHELL_BASED_SURFACE_MODEL',
       'AIC_MANIFOLD_SURFACE.MAPPED_ITEM',
       'AIC_MANIFOLD_SURFACE.AXIS2_PLACEMENT_3D'] * TYPEOF (it)) = 1))) = 0;
  WR2: SIZEOF (QUERY (it <* SELF.items |
       SIZEOF (['AIC_MANIFOLD_SURFACE.SHELL_BASED_SURFACE_MODEL',
       'AIC_MANIFOLD_SURFACE.MAPPED_ITEM'] * TYPEOF (it)) = 1)) > 0;
  WR3: SIZEOF (QUERY (mi <* QUERY (it <* SELF.items |
       'AIC_MANIFOLD_SURFACE.MAPPED_ITEM' IN TYPEOF (it)) |
       NOT ('AIC_MANIFOLD_SURFACE.MANIFOLD_SURFACE_SHAPE_REPRESENTATION'
       IN TYPEOF (mi\mapped_item.mapping_source.mapped_representation))))
       = 0;
  WR4: SIZEOF (QUERY (sbsm <* QUERY (it <* SELF.items |
       'AIC_MANIFOLD_SURFACE.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
       NOT (SIZEOF (QUERY (sh <*
       sbsm\shell_based_surface_model.sbsm_boundary |
       NOT (SIZEOF (['AIC_MANIFOLD_SURFACE.OPEN_SHELL',
       'AIC_MANIFOLD_SURFACE.CLOSED_SHELL']
       * TYPEOF (sh)) = 1))) = 0))) = 0;
  WR5: SIZEOF (QUERY (sbsm <* QUERY (it <* SELF.items |
       'AIC_MANIFOLD_SURFACE.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
       NOT (SIZEOF (QUERY (cfs <*
       sbsm\shell_based_surface_model.sbsm_boundary |
       NOT (SIZEOF (QUERY (fa <* cfs\connected_face_set.cfs_faces |
       NOT (SIZEOF (['AIC_MANIFOLD_SURFACE.FACE_SURFACE',
       'AIC_MANIFOLD_SURFACE.ORIENTED_FACE'] * TYPEOF (fa)) = 1))) = 0)))
       = 0))) = 0;
  WR6: SIZEOF (QUERY (sbsm <* QUERY (it <* SELF.items |
       'AIC_MANIFOLD_SURFACE.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
       NOT (SIZEOF (QUERY (cfs <*
       sbsm\shell_based_surface_model.sbsm_boundary |
       NOT (SIZEOF (QUERY (f_sf <* QUERY (fa <*
       cfs\connected_face_set.cfs_faces |
       'AIC_MANIFOLD_SURFACE.FACE_SURFACE' IN TYPEOF (fa)) |
       NOT (('AIC_MANIFOLD_SURFACE.ADVANCED_FACE' IN TYPEOF (f_sf))
       OR
       (SIZEOF (['AIC_MANIFOLD_SURFACE.OFFSET_SURFACE',
```

```
      'AIC_MANIFOLD_SURFACE.SURFACE_REPLICA'] * TYPEOF
      (f_sf\face_surface.face_geometry)) = 1)))) = 0))) = 0))) = 0;
 WR7: SIZEOF (QUERY (sbsm <* QUERY (it <* SELF.items |
      'AIC_MANIFOLD_SURFACE.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs\connected_face_set.cfs_faces |
      NOT (('AIC_MANIFOLD_SURFACE.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (basis_surface_check(fa\face_surface.face_geometry))))) = 0)))
      = 0))) = 0;
 WR8: SIZEOF (QUERY (sbsm <* QUERY (it <* SELF.items |
      'AIC_MANIFOLD_SURFACE.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs\connected_face_set.cfs_faces |
      NOT (('AIC_MANIFOLD_SURFACE.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (SIZEOF (QUERY (bnds <* fa.bounds |
      NOT (SIZEOF (['AIC_MANIFOLD_SURFACE.EDGE_LOOP',
      'AIC_MANIFOLD_SURFACE.VERTEX_LOOP']
      * TYPEOF (bnds.bound)) = 1))) = 0)))) = 0))) = 0))) = 0;
 WR9: SIZEOF (QUERY (sbsm <* QUERY (it <* SELF.items|
      'AIC_MANIFOLD_SURFACE.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs\connected_face_set.cfs_faces |
      NOT (('AIC_MANIFOLD_SURFACE.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
      'AIC_MANIFOLD_SURFACE.EDGE_LOOP' IN TYPEOF (bnds)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
      NOT ('AIC_MANIFOLD_SURFACE.EDGE_CURVE' IN TYPEOF
      (oe.edge_element)))) = 0))) = 0)))) = 0))) = 0))) = 0;
WR10: SIZEOF (QUERY (sbsm <* QUERY (it <* SELF.items |
      'AIC_MANIFOLD_SURFACE.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs\connected_face_set.cfs_faces |
      NOT (('AIC_MANIFOLD_SURFACE.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
      'AIC_MANIFOLD_SURFACE.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
      NOT (SIZEOF (QUERY (oe_cv <* QUERY (oe <*
      elp_fbnds.bound\path.edge_list |
      'AIC_MANIFOLD_SURFACE.EDGE_CURVE' IN TYPEOF (oe.edge_element)) |
      NOT (SIZEOF (['AIC_MANIFOLD_SURFACE.CURVE_REPLICA',
      'AIC_MANIFOLD_SURFACE.OFFSET_CURVE_3D',
      'AIC_MANIFOLD_SURFACE.PCURVE'] *
      TYPEOF (oe_cv.edge_element\edge_curve.edge_geometry))
      = 1))) = 0))) = 0)))) = 0))) = 0))) = 0;
WR11: SIZEOF (QUERY (sbsm <* QUERY (it <* SELF.items |
      'AIC_MANIFOLD_SURFACE.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs\connected_face_set.cfs_faces |
```

```
      NOT (('AIC_MANIFOLD_SURFACE.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
      'AIC_MANIFOLD_SURFACE.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
      NOT (basis_curve_check (oe.edge_element\edge_curve.edge_geometry))))
      = 0))) = 0)))) = 0))) = 0))) = 0;
WR12: SIZEOF (QUERY(sbsm <* QUERY (it <* SELF.items |
      'AIC_MANIFOLD_SURFACE.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs\connected_face_set.cfs_faces |
      NOT (('AIC_MANIFOLD_SURFACE.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
      'AIC_MANIFOLD_SURFACE.EDGE_LOOP' IN TYPEOF (bnds)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list|
      NOT (('AIC_MANIFOLD_SURFACE.VERTEX_POINT' IN TYPEOF
      (oe.edge_element.edge_start))
      AND
      ('AIC_MANIFOLD_SURFACE.VERTEX_POINT' IN
      TYPEOF (oe.edge_element.edge_end)))))
      = 0))) = 0)))) = 0))) = 0))) = 0;
WR13: SIZEOF (QUERY (sbsm <* QUERY (it <* SELF.items |
      'AIC_MANIFOLD_SURFACE.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs\connected_face_set.cfs_faces |
      NOT (('AIC_MANIFOLD_SURFACE.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
      'AIC_MANIFOLD_SURFACE.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
      NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
      NOT ((SIZEOF (['AIC_MANIFOLD_SURFACE.CARTESIAN_POINT',
      'AIC_MANIFOLD_SURFACE.DEGENERATE_PCURVE',
      'AIC_MANIFOLD_SURFACE.POINT_ON_CURVE',
      'AIC_MANIFOLD_SURFACE.POINT_ON_SURFACE'] * TYPEOF
      (oe.edge_element.edge_start\vertex_point.vertex_geometry)) = 1)
      AND
      (SIZEOF (['AIC_MANIFOLD_SURFACE.CARTESIAN_POINT',
      'AIC_MANIFOLD_SURFACE.DEGENERATE_PCURVE',
      'AIC_MANIFOLD_SURFACE.POINT_ON_CURVE',
      'AIC_MANIFOLD_SURFACE.POINT_ON_SURFACE'] * TYPEOF
      (oe.edge_element.edge_end\vertex_point.vertex_geometry)) = 1
      )))) = 0))) = 0)))) = 0))) = 0))) = 0;
WR14: SIZEOF (QUERY (sbsm <* QUERY (it <* SELF.items |
      'AIC_MANIFOLD_SURFACE.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <*
      sbsm\shell_based_surface_model.sbsm_boundary |
      NOT (SIZEOF (QUERY (fa <* cfs\connected_face_set.cfs_faces |
      NOT (('AIC_MANIFOLD_SURFACE.ADVANCED_FACE' IN TYPEOF (fa))
      OR
      (SIZEOF (QUERY (vlp_fbnds <* QUERY (bnds <* fa.bounds |
      'AIC_MANIFOLD_SURFACE.VERTEX_LOOP' IN TYPEOF (bnds.bound)) |
      NOT ('AIC_MANIFOLD_SURFACE.VERTEX_POINT' IN TYPEOF
      (vlp_fbnds.bound\vertex_loop.loop_vertex)))) = 0)))) = 0)))
```

```
       = 0))) = 0;
 WR15: SIZEOF (QUERY (sbsm <* QUERY (it <* SELF.items |
       'AIC_MANIFOLD_SURFACE.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
       NOT (SIZEOF (QUERY (cfs <*
       sbsm\shell_based_surface_model.sbsm_boundary |
       NOT (SIZEOF (QUERY (fa <* cfs\connected_face_set.cfs_faces |
       NOT (('AIC_MANIFOLD_SURFACE.ADVANCED_FACE' IN TYPEOF (fa))
       OR
       (SIZEOF (QUERY (vlp_fbnds <* QUERY (bnds <* fa.bounds |
       'AIC_MANIFOLD_SURFACE.VERTEX_LOOP' IN TYPEOF (bnds.bound)) |
       NOT (SIZEOF (['AIC_MANIFOLD_SURFACE.CARTESIAN_POINT',
       'AIC_MANIFOLD_SURFACE.DEGENERATE_PCURVE',
       'AIC_MANIFOLD_SURFACE.POINT_ON_CURVE',
       'AIC_MANIFOLD_SURFACE.POINT_ON_SURFACE'] * TYPEOF
       (vlp_fbnds.bound\vertex_loop.loop_vertex\vertex_point.vertex_geometry))
       = 1))) = 0)))) = 0))) = 0))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: The **items** in a **manifold_surface_shape_representation** shall be **shell_based_surface_model**s, **mapped_item**s, or **axis2_placement_3d**s.

NOTE - **Axis2_placement_3d** is valid as a reference as **mapping_target** from a **mapped_item**. To include another representation into the list of items of a **manifold_surface_shape_ representation** (see WR3 for valid **mapped_-items**), the **mapped_item.mapping_source.mapping_origin** may be any entity that is geometrically founded in the **geometric_representation_context** of the **mapped_representation**. If this entity is an **axis2_placement_3d**, the mapping operator for the **mapped_representation** to the **manifold_surface_shape_representation** corresponds to a transformation matrix with only translation and rotation enabled. If a **cartesian_transformation_-operator_3d** is used as **mapping_origin**, scaling and mirroring are possible.

WR2: At least one of the **items** in a **manifold_surface_shape_representation** shall be either a **shell_-based_surface_model** or a **mapped_item**.

WR3: If there is a **mapped_item** in a **manifold_surface_shape_representation**, the **mapped_representation** of its **mapping_source** shall be a **manifold_surface_shape_representation**.

WR4: A **shell_based_surface_model** that is one of the items of a **manifold_surface_shape_representation** shall only reference **open_shell** or **closed_shell** as its **sbsm_boundary**.

WR5: A face which is one of the **cfs_face**s of a **connected_face_set** of a **manifold_surface_shape_-representation** shall be either a **face_surface** or an **oriented_face**.

WR6: The **face_geometry** of a **face_surface** that is a face in a **connected_face_set** of a **manifold_surface_-shape_representation** shall be either an **offset_surface**, a **surface_replica**, or a surface within the reference tree of an **advanced_face**.

WR7: All basis geometry that is referenced by surfaces that are within the reference tree of a **manifold_surface_shape_representation** shall be valid curves and surfaces.

The **basis_surface** of an **offset_surface** shall be either an **elementary_surface**, **b_spline_surface**, **offset_surface**, **swept_surface**, or **surface_replica**.

The **parent_surface** of a **surface_replica** shall be either an **elementary_surface**, **b_spline_surface**, **offset_surface**, **swept_surface**, or **surface_replica**.

The **swept_curve** of a **swept_surface** shall be either a **line**, **conic**, **pcurve**, **surface_curve**, **offset_curve_3d**, **b_spline_curve**, **polyline**, or **curve_replica**.

WR8: The bound of a **face_bound** that is one of the bounds of a face in a **connected_face_set** in a **shell_based_surface_model** in a **manifold_surface_shape_representation** shall be either within the reference tree of an **advanced_face** or shall be an **edge_loop** or a **vertex_loop**.

WR9: The **edge_element** of an **oriented_edge** that is part of a **connected_face_set** and within the reference tree of a **manifold_surface_shape_representation** shall be either within the reference tree of an **advanced_face** or shall be an **edge_curve**.

WR10: The **edge_geometry** of an **edge_curve** that is an edge and within an **edge_loop** that is one of the bounds of a face in a **connected_face_set** in a **manifold_surface_shape_representation** shall be either a **pcurve**, an **offset_curve_3d**, a **curve_replica**, or a **curve** in the reference tree of an **advanced_face**.

WR11: All basis geometry that is referenced by curves that are within the reference tree of a **manifold_surface_shape_representation** shall be valid **curves** and **surfaces**. The **parent_curve** of a **curve_replica** shall be either a **line**, **conic**, **pcurve**, **surface_curve**, **offset_curve_3d**, **b_spline_curve**, **polyline**, or **curve_replica**.

The **basis_curve** of an **offset_curve_3d** shall be either a **line**, **conic**, **pcurve**, **surface_curve**, **offset_curve_3d**, **b_spline_curve**, **polyline**, or **curve_replica**.

The **curve_3d** of a **surface_curve** shall be either a **line**, **conic**, **offset_curve_3d**, **b_spline_curve**, **polyline**, or **curve_replica**.

The **basis_surface** of a **surface_curve** shall be either a **b_spline_surface**, **elementary_surface**, **offset_surface**, **surface_replica**, or **swept_surface**.

WR12: The **edge_start** and **edge_end** of an **edge** that is part of a **connected_face_set** and within the reference tree of a **manifold_surface_shape_representation** shall be either within the reference tree of an **advanced_face** or **vertex_point**s.

WR13: The **vertex_geometry** of a **vertex** that is part of an **edge_loop** and within the reference tree of a

manifold_surface_shape_representation shall be either within the reference tree of an **advanced_face** or shall be a **cartesian_point**, **point_on_curve**, **point_on_surface**, or **degenerate_pcurve**.

WR14: The **loop_vertex** of a **vertex_loop** that is part of a **connected_face_set** and within the reference tree of a **manifold_surface_shape_representation** shall be either within the reference tree of an **advanced_face** or shall be a **vertex_point**.

WR15: The **vertex_geometry** of a **vertex** that is part of a **vertex_loop** and within the reference tree of a **manifold_surface_shape_representation** shall be either within the reference tree of an **advanced_face** or shall be a **cartesian_point**, **point_on_curve**, **point_on_surface**, or **degenerate_pcurve**.

Informal propositions:

IP1: A **b_spline_curve** shall not self-intersect.

IP2: A **b_spline_surface** shall not self-intersect.

IP3: An **offset_curve_3d** shall not self-intersect.

IP4: An **offset_surface** shall not self-intersect.

IP5: If a **face** within the reference tree of a **manifold_surface_shape_representation** has only one connected outer bound, the corresponding loop shall be represented as **face_outer_bound**. If the outer bound is not connected, **face_outer_bound** shall not be used.

F.4.2.2 aic_manifold_surface function definitions

The EXPRESS language has a number of built in functions. This section describes additional functions required for the definition and constraints on the **aic_manifold_surface** schema. These functions are used in the specification of the entity **manifold_surface_shape_representation**.

F.4.2.2.1 basis_curve_check

The **basis_curve_check** function performs type checking for curves. An instance is checked against all types of curves that are valid in the context of a **manifold_surface_shape_representation**. All related geometry, that is curves and surfaces, are correspondingly checked.

Where appropriate an instance is investigated recursively. This means if a curve references another curve as a basis curve or parent curve, the **basis_curve_check** function is called again. If a surface is referenced, the **basis_surface_check** function is called. The recursion terminates at entity types that do not reference any curves or surfaces.

The following curve types are within the scope of the **manifold_surface_shape_representation** and are, thus, valid input to this function:

— b_spline_curve;

— conic;

— curve_replica;

— line;

— offset_curve_3d;

— pcurve;

— polyline;

— surface_curve.

Four of these curve types reference basis or parent curves. The lists below indicate the valid references.

The **parent_curve** of a **curve_replica** and the **basis_curve** of an **offset_curve_3d** shall be of one of the following types:

— b_spline_curve;

— conic;

— curve_replica (recursive);

— line;

— offset_curve_3d (recursive);

— pcurve (recursive);

— polyline;

— surface_curve (recursive).

The one instance in the set of **items** of a **definitional_representation** that is referenced as **reference_to_curve** by a **pcurve** shall be of one of the following types:

— b_spline_curve;

— conic;

— curve_replica (recursive);

— line;

— polyline.

The **curve_3d** of a **surface_curve** shall be of one of the following types:

— b_spline_curve;

— conic;

— curve_replica (recursive);

— line;

— offset_curve_3d (recursive);

— polyline;

— surface_curve (recursive).

Pcurve and **surface_curve** also reference surfaces. Function **basis_surface_check** is called for type checking. The **manifold_surface_shape_representation** requires the same constraints on valid surface references for **pcurves** and **surface_curves** as specified in ISO 10303-42. A valid polyline shall consist of at least three **cartesian_point**s. The attribute **self_intersect** shall for **B-spline** and offset geometry be set to FALSE.

This function returns TRUE if the types of all referenced geometries are within the scope of the **manifold_surface_shape_representation** and if all constraints are fulfilled; else it returns FALSE.

NOTES

1 - This function does not check the correctness of references with respect to ISO 10303-42. Only requirements due to the scope of the **manifold_surface_shape_representation** are checked.

2 - The function uses recursion to trace entity references.

EXPRESS specification:

```
*)
FUNCTION basis_curve_check (cv: curve) : BOOLEAN;

  -- let those types pass that do reference curves or surfaces and
  -- that do not have any additional constraints

  IF (('GEOMETRY_SCHEMA.B_SPLINE_CURVE' IN TYPEOF (cv)) AND
    (cv\b_spline_curve.self_intersect = FALSE))
  THEN RETURN(TRUE);
  ELSE

    -- avoid line and conic to be in complex entities with bounded_curves,
    -- but else they are valid

    IF SIZEOF (['GEOMETRY_SCHEMA.BOUNDED_CURVE', 'GEOMETRY_SCHEMA.CONIC',
      'GEOMETRY_SCHEMA.LINE'] * TYPEOF (cv)) = 1 THEN
      RETURN(TRUE);
    ELSE

      -- check in case curve_replica is the type of the input curve

      IF SIZEOF (['GEOMETRY_SCHEMA.BOUNDED_CURVE',
        'GEOMETRY_SCHEMA.CURVE_REPLICA'] * TYPEOF (cv)) = 1 THEN
        RETURN (basis_curve_check(cv\curve_replica.parent_curve));
      ELSE

        -- check in case offset_curve_3d is the type of the input curve;
        -- attribute self_intersect shall be false

        IF ((SIZEOF (['GEOMETRY_SCHEMA.BOUNDED_CURVE',
          'GEOMETRY_SCHEMA.OFFSET_CURVE_3D'] * TYPEOF (cv)) = 1) AND
          (cv\offset_curve_3d.self_intersect = FALSE)) THEN
          RETURN (basis_curve_check (cv\offset_curve_3d.basis_curve));
        ELSE

          -- check in case pcurve is the type of the input curve

          IF SIZEOF (['GEOMETRY_SCHEMA.BOUNDED_CURVE',
            'GEOMETRY_SCHEMA.PCURVE'] * TYPEOF(cv)) = 1 THEN
            RETURN ((basis_curve_check
            (cv\pcurve.reference_to_curve\representation.items[1])) AND
            (basis_surface_check (cv\pcurve.basis_surface)));

          ELSE

            -- check in case surface_curve is the type of the input curve

            IF SIZEOF (['GEOMETRY_SCHEMA.BOUNDED_CURVE',
              'GEOMETRY_SCHEMA.SURFACE_CURVE'] * TYPEOF(cv)) = 1 THEN

              -- if the curve reference is correct, check also the rest

              IF basis_curve_check (cv\surface_curve.curve_3d) THEN
                REPEAT i := 1 TO SIZEOF
                (cv\surface_curve.associated_geometry);
```

```
                    -- do for one or two associated_geometrys:

                    IF 'GEOMETRY_SCHEMA.SURFACE' IN
                      TYPEOF (cv\surface_curve.associated_geometry[i]) THEN
                      IF NOT basis_surface_check
                        (cv\surface_curve.associated_geometry[i]) THEN
                        RETURN(FALSE);
                      END_IF;
                    ELSE
                      IF 'GEOMETRY_SCHEMA.PCURVE' IN
                        TYPEOF (cv\surface_curve.associated_geometry[i]) THEN
                        IF NOT basis_curve_check
                          (cv\surface_curve.associated_geometry[i]) THEN
                          RETURN(FALSE);
                        END_IF;
                      END_IF;
                    END_IF;
                  END_REPEAT;
                  RETURN(TRUE);
                END_IF;
              ELSE

                -- check in case polyline is the type of the input curve;
                -- polyline shall have at least 3 points (as required for
                -- advanced_face)

                IF 'GEOMETRY_SCHEMA.POLYLINE' IN TYPEOF(cv) THEN
                  IF (SIZEOF (cv\polyline.points) >= 3) THEN RETURN (TRUE);
                  END_IF;
                END_IF;
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;
(*
```

Argument_definitions:

cv: (input) the **curve** that is being checked for a valid **curve** in a **manifold_surface_shape_representation**.

BOOLEAN: (output) is TRUE if the **curve** is a valid **curve**; else FALSE.

F.4.2.2.2 basis_surface_check

The **basis_surface_check** function performs type checking for surfaces. An instance is checked against all types of surfaces that are valid in the context of a **manifold_surface_shape_representation**. All related geometry, that is curves and surfaces, are correspondingly checked.

Where appropriate an instance is investigated recursively. This means if a surface references another surface as a basis surface or parent surface the **basis_surface_check** function is called again. If a curve is referenced, the **basis_curve_check** function is called. The recursion terminates at entity types that do not reference any curves or surfaces.

The following surface types are within the scope of the **manifold_surface_shape_representation** and are, thus, valid input to this function:

— b_spline_surface;

— elementary_surface;

— offset_surface;

— surface_replica;

— swept_surface.

Two of these surface types reference basis or parent surfaces. The **parent_surface** of a **surface_replica** and the **basis_surface** of an **offset_surface** shall both be of one of the following types:

— b_spline_surface;

— elementary_surface;

— offset_surface (recursive);

— surface_replica (recursive);

— swept_surface.

Swept_surfaces reference curves. Function **basis_curve_check** is called for type checking. The **manifold_surface_shape_representation** requires the same constraints on valid sweeping curves as specified in ISO 10303-42. All curves that are in the scope of a **manifold_surface_shape_representation** are valid **swept_curve**s.

The attribute **self_intersect** shall for **B-spline** and offset geometry be set to FALSE.

This function returns TRUE if the types of all referenced geometries are within the scope of the **manifold_surface_shape_representation** and if all constraints are fulfilled; else it returns FALSE.

NOTES

1 - This function does not check the correctness of references with respect to requirements specified by ISO 10303-42. Only requirements due to the scope of the **manifold_surface_shape_representation** are checked.

2 - The function uses recursion to trace entity references.

EXPRESS specification:

```
*)
FUNCTION basis_surface_check (surf : surface) : BOOLEAN;

-- let those types pass that do not reference curves or surfaces

  IF 'GEOMETRY_SCHEMA.ELEMENTARY_SURFACE' IN TYPEOF(surf) THEN
    RETURN(TRUE);
  ELSE

      -- check the swept_curve in case the input surface is a swept_surface

    IF 'GEOMETRY_SCHEMA.SWEPT_SURFACE' IN TYPEOF (surf) THEN
      RETURN (basis_curve_check (surf\swept_surface.swept_curve));
    ELSE

      -- check in case offset_surface is the type of the input surface;
      -- attribute self_intersect shall be false

      IF (('GEOMETRY_SCHEMA.OFFSET_SURFACE' IN TYPEOF (surf)) AND
        (surf\offset_surface.self_intersect = FALSE)) THEN
        RETURN (basis_surface_check (surf\offset_surface.basis_surface));
      ELSE

        -- check in case surface_replica is the type of the input surface

        IF 'GEOMETRY_SCHEMA.SURFACE_REPLICA' IN TYPEOF(surf) THEN
          RETURN(basis_surface_check (surf\surface_replica.parent_surface));
        ELSE

          -- check for self_intersection flag in case b_spline_surface

          IF (('GEOMETRY_SCHEMA.B_SPLINE_SURFACE' IN TYPEOF(surf)) AND
            (surf\b_spline_surface.self_intersect = FALSE)) THEN
            RETURN(TRUE);
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION;
(*
```

Argument definitions:

surf: (input) the **surface** that is being checked for a valid **surface** in a **manifold_surface_shape_-**

representation.

BOOLEAN: (output) is TRUE if the **surface** is a valid **surface**; else FALSE.

*)
END_SCHEMA; -- aic_manifold_surface
(*

F.5 Application interpreted construct 510: Geometrically bounded wireframe

F.5.1 Scope

This application interpreted construct specifies the interpretation of the integrated resources to satisfy requirements for the representation of the product shape using geometrically bounded wireframe models.

The following are within the scope of this AIC:

— points and curve geometry in 3D cartesian space;

— the use of wireframe geometry to represent a shape;

— the combination of representations to form an aggregation of representations.

The following are outside the scope of this AIC:

— surface geometry;

— topological connectivity of geometry;

— product definitions.

F.5.2 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 AIC.

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. References to eliminated constructs are outside the scope of the AIC.

```
*)
SCHEMA aic_geometrically_bounded_wireframe;

  USE FROM geometric_model_schema                          -- ISO 10303-42
```

```
    (geometric_curve_set,
     geometric_set);

  USE FROM geometry_schema                                    -- ISO 10303-42
    (axis1_placement,
     axis2_placement_3d,
     b_spline_curve_with_knots,
     bezier_curve,
     cartesian_transformation_operator_3d,
     circle,
     composite_curve,
     curve,
     curve_replica,
     ellipse,
     geometric_representation_context,
     hyperbola,
     line,
     offset_curve_3d,
     parabola,
     point,
     point_on_curve,
     point_replica,
     polyline,
     quasi_uniform_curve,
     rational_b_spline_curve,
     reparametrised_composite_curve_segment,
     trimmed_curve,
     uniform_curve);

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

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

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
representation_schema	ISO 10303-43

F.5.2.1 aic_geometrically_bounded_wireframe entity definition: geometrically_-bounded_wireframe_shape_representation

A **geometrically_bounded_wireframe_shape_representation** is a **shape_representation** which represents the shape or portions of the shape of a **product** by wireframe geometry without topology. These representations are formed by the use of points and curves only. All unbounded curves shall be explicitly trimmed unless they are closed. The boundaries of the curves shall be defined explicitly by points on the curves and explicit associations between the points and the curves which they bound or by parameter values. Each

geometric_set in a **geometrically_bounded_wireframe_shape_representation** shall contain only those entities that define the physical object that is being represented by a particular instance of the **geometrically_-bounded_wireframe_shape_representation**. The geometric entities that are used to support the definition of another geometric entity shall not exist themselves in the **elements** set of a **geometric_set**.

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

EXAMPLE - A circular arc is to be used to define the corner radius on a part that is being represented using a **geometrically_bounded_wireframe_shape_representation**. The representation of that arc is a **circle** that is referenced by a **trimmed_curve** as it **basis_curve**. The **geometric_set** that contains the shape of the part has the **trimmed_curve** in its set of **elements**, but does not contain the **circle** that is used as the **basis_curve**.

EXPRESS specification:

```
*)
ENTITY geometrically_bounded_wireframe_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (it <* SELF.items |
         NOT (SIZEOF (TYPEOF(it) *
           ['AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.GEOMETRIC_CURVE_SET',
            'AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.AXIS2_PLACEMENT_3D',
            'AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.MAPPED_ITEM']) = 1)
       )) = 0;
  WR2: SIZEOF (QUERY (it <* SELF.items |
         SIZEOF (TYPEOF (it) *
           ['AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.GEOMETRIC_CURVE_SET',
            'AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.MAPPED_ITEM']) = 1
       )) >= 1;
  WR3: SIZEOF (QUERY (gcs <* QUERY (it <* SELF.items |
         ('AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.GEOMETRIC_CURVE_SET'
              IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (crv <*
           QUERY (elem <* gcs\geometric_set.elements |
              ('AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.CURVE' IN TYPEOF (elem))) |
           NOT (valid_geometrically_bounded_wf_curve
                  (crv, 'AIC_GEOMETRICALLY_BOUNDED_WIREFRAME'))
         )) = 0)
       )) = 0;
  WR4: SIZEOF (QUERY (gcs <* QUERY (it <* SELF.items |
         ('AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.GEOMETRIC_CURVE_SET'
              IN TYPEOF (it))) |
         NOT (SIZEOF (QUERY (pnts <*
           QUERY (elem <* gcs\geometric_set.elements |
              ('AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.POINT' IN TYPEOF (elem))) |
           NOT (valid_geometrically_bounded_wf_point
                  (pnts, 'AIC_GEOMETRICALLY_BOUNDED_WIREFRAME'))
         )) = 0)
       )) = 0;
  WR5: SIZEOF (QUERY (gcs <* QUERY (it <* SELF.items |
         ('AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.GEOMETRIC_CURVE_SET'
```

```
            IN TYPEOF (it))) |
        NOT (SIZEOF (QUERY (cnc <*
          QUERY (elem <* gcs\geometric_set.elements |
            ('AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.CONIC' IN TYPEOF (elem))) |
          NOT ('AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.AXIS2_PLACEMENT_3D'
            IN TYPEOF (cnc\conic.position))
        )) = 0)
      )) = 0;
  WR6: SIZEOF (QUERY (gcs <* QUERY (it <* SELF.items |
        ('AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.GEOMETRIC_CURVE_SET'
            IN TYPEOF (it))) |
        NOT (SIZEOF (QUERY (pline <*
          QUERY (elem <* gcs\geometric_set.elements |
            ('AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.POLYLINE'
              IN TYPEOF (elem))) |
          NOT (SIZEOF (pline\polyline.points) > 2)
        )) = 0)
      )) = 0;
  WR7: SIZEOF (QUERY (mi <* QUERY (it <* SELF.items |
         ('AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.MAPPED_ITEM'
            IN TYPEOF (it))) |
          NOT ('AIC_GEOMETRICALLY_BOUNDED_WIREFRAME.' +
            'GEOMETRICALLY_BOUNDED_WIREFRAME_SHAPE_REPRESENTATION'
          IN TYPEOF (mi\mapped_item.mapping_source.mapped_representation))
      )) = 0;
  WR8: SELF.context_of_items\geometric_representation_context.
         coordinate_space_dimension = 3;
END_ENTITY;
(*
```

Formal propositions:

WR1: The **items** in a **geometrically_bounded_wireframe_shape_representation** shall be a **geometric_-curve_set**, **axis2_placement_3d**, or **mapped_item** .

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

WR3: All curves that are in the **elements** of a **geometric_curve_set** for a **geometrically_bounded_-wireframe_shape_representation** shall be bounded explicitly by geometry.

WR4: All **points** that are in the **elements** of the **geometric_curve_set** for a **geometrically_bounded_-wireframe_shape_representation** shall be a **cartesian_point**, **point_on_curve**, or **point_replica**. The **point_-replica** shall replicate either another **point_replica** or a **cartesian_point**. The **point_on_curve** shall lie on a valid curve type for a **geometrically_bounded_wireframe_shape_representation**.

WR5: The **position** for a **conic** in the **elements** of a **geometric_curve_set** for a **geometrically_bounded_-wireframe_shape_representation** shall only be an **axis2_placement_3d**.

WR6: Every **polyline** in the **elements** of a **geometric_curve_set** for a **geometrically_bounded_wireframe_-**

shape_representation shall contain more than two distinct **point**s.

WR7: If there is a **mapped_item** in a **geometrically_bounded_wireframe_shape_representation**, the source of the **mapped_item** shall be a **geometrically_bounded_wireframe_shape_representation**.

WR8: The **geometrically_bounded_wireframe_shape_representation** shall have a **coordinate_space_-dimension** equal to three.

F.5.2.2 aic_geometrically_bounded_wireframe function definitions

F.5.2.2.1 valid_geometrically_bounded_wf_curve

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

EXPRESS specification:

```
*)
FUNCTION valid_geometrically_bounded_wf_curve
   (crv   : curve; schma : STRING) : BOOLEAN;

  -- check for valid basic curve types
  IF SIZEOF ([schma + '.POLYLINE',
              schma + '.B_SPLINE_CURVE',
              schma + '.ELLIPSE',
              schma + '.CIRCLE'] * TYPEOF (crv)) = 1
    THEN RETURN (TRUE);
  ELSE
    -- if the curve is a trimmed_curve
    IF ((schma + '.TRIMMED_CURVE') IN TYPEOF (crv)) THEN
      -- if a line, parabola, or hyperbola is being trimmed, then valid
      IF SIZEOF ([schma + '.LINE',
                  schma + '.PARABOLA',
                  schma + '.HYPERBOLA'] *
           TYPEOF(crv\trimmed_curve.basis_curve)) = 1
        THEN RETURN (TRUE);

      -- otherwise, recursively check basis_curve
      ELSE RETURN (valid_geometrically_bounded_wf_curve
                        (crv\trimmed_curve.basis_curve, schma));
      END_IF;
    ELSE
      -- recursively check the offset_curve basis curve
      IF ((schma + '.OFFSET_CURVE_3D') IN TYPEOF (crv))
        THEN RETURN (valid_geometrically_bounded_wf_curve
                          (crv\offset_curve_3d.basis_curve, schma));
      ELSE
        -- recursively check the curve_replica parent curve
        IF ((schma + '.CURVE_REPLICA') IN TYPEOF (crv))
          THEN RETURN (valid_geometrically_bounded_wf_curve
```

```
                        (crv\curve_replica.parent_curve, schma));
        ELSE
          -- recursively check the composite_curve segments
          IF ((schma + '.COMPOSITE_CURVE') IN TYPEOF (crv)) THEN
            RETURN (SIZEOF (QUERY (ccs <* crv\composite_curve.segments |
                      NOT (valid_geometrically_bounded_wf_curve
                             (ccs.parent_curve, schma)))) = 0);
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
END_FUNCTION;
(*
```

Argument definitions:

crv: the input **curve** that is to be examined.

schma: the input string that defines the schema in which the **crv** is defined to be used for type checking.

F.5.2.2.2 valid_geometrically_bounded_wf_point

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

EXPRESS specification:

```
*)
FUNCTION valid_geometrically_bounded_wf_point
   (pnt : point; schma : STRING) : BOOLEAN;

  -- check for valid basis types
  IF ((schma + '.CARTESIAN_POINT') IN TYPEOF (pnt))
    THEN RETURN (TRUE);
  ELSE
    -- if the input type is a point_on_curve then check for a valid
    -- geometrically bounded curve type as the basis
    IF ((schma + '.POINT_ON_CURVE') IN TYPEOF (pnt))
      THEN RETURN (valid_geometrically_bounded_wf_curve
                     (pnt\point_on_curve.basis_curve, schma));

    ELSE
      -- if the input type is a point_replica check for a valid parent point
      IF ((schma + '.POINT_REPLICA') IN TYPEOF (pnt))
        THEN RETURN (valid_geometrically_bounded_wf_point
                       (pnt\point_replica.parent_pt, schma));
      END_IF;
    END_IF;
  END_IF;
  RETURN (FALSE);
```

```
END_FUNCTION;
(*
```

Argument definitions:

pnt: the input **point** that is to be examined.

schma: the input string that defines the schema in which the **pnt** is defined to be used for type checking.

```
*)
END_SCHEMA; -- aic_geometrically_bounded_wireframe
(*
```

F.6 Application interpreted construct 511: Topologically bounded surface

F.6.1 Scope

This application interpreted construct specifies the interpretation of the integrated resources in order to satisfy the requirement for the definition of a face with explicit topological bounds and fully defined geometry.

The following are within the scope of this AIC:

- 3D geometry;
- B—spline curves and surfaces;
- conics;
- elementary curves;
- elementary surfaces;
- polylines;
- pcurves;
- sculptured surfaces;
- surface curves referencing pcurves;
- swept surfaces;
- twisted curves;

— unbounded geometry;

— geometric transformations;

— use of topology to bound geometric entities.

The following are outside the scope of this AIC:

— 2D geometry other than for the definition of a pcurve in the parameter space of a surface;

— bounded curves other than polylines and B-spline curves;

— bounded surfaces other than B-spline surfaces;

— offset curves and surfaces.

F.6.2 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 the EXPRESS short listing of this AIC.

NOTE 1 - There may be supertypes 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. References to eliminated constructs are outside the scope of the AIC.

This application interpreted construct provides a consistent set of geometric and topological entities for the definition of a face with fully defined geometry and explicit topology defining the boundaries. The permissible types of face geometry include elementary surfaces and B-spline surfaces. Edges are required to have their geometry defined by curves, which may include **pcurve**s.

The highest level entity in this AIC is **advanced_face** which is a specialised subtype of **face_surface** (see ISO 10303-42). The rules on this entity ensure that the topology and geometry are fully defined.

```
*)
SCHEMA aic_topologically_bounded_surface;

  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,
    circle,
    conical_surface,
```

```
    cylindrical_surface,
    degenerate_toroidal_surface,
    direction,
    ellipse,
    hyperbola,
    line,
    parabola,
    pcurve,
    plane,
    polyline,
    quasi_uniform_curve,
    quasi_uniform_surface,
    rational_b_spline_curve,
    rational_b_spline_surface,
    spherical_surface,
    surface_curve,
    surface_of_linear_extrusion,
    surface_of_revolution,
    swept_surface,
    toroidal_surface,
    uniform_curve,
    uniform_surface,
    vector);

  USE FROM topology_schema(                                    -- ISO 10303-42
    edge_curve,
    edge_loop,
    face_bound,
    face_outer_bound,
    face_surface,
    path,
    vertex_loop,
    vertex_point);

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

  USE FROM measure_schema                                      -- ISO 10303-41
   (parameter_value);

(*
```

NOTES

2 - The entities **path** and **swept_surface** are explicitly interfaced (i.e. included in the USE FROM lists) to allow rules within the **advanced_face** entity to access attributes of these entities.

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

geometry_schema	ISO 10303-42
topology_schema	ISO 10303-42
representation_schema	ISO 10303-43
measure_schema	ISO 10303-41

F.6.2.1 aic_topologically_bounded_surface entity definition: 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 the top level entity which is used to formulate the precise meaning of a topologically bounded surface corresponding to the scope of this AIC.

NOTE 1 - An application protocol that uses this AIC may ensure that a **face** entity is instantiated as an **advanced_face**.

EXPRESS specification:

```
*)
ENTITY advanced_face
  SUBTYPE OF (face_surface);
WHERE
  WR1 : SIZEOF
        (['AIC_TOPOLOGICALLY_BOUNDED_SURFACE.ELEMENTARY_SURFACE',
          'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.B_SPLINE_SURFACE',
          'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.SWEPT_SURFACE'] *
           TYPEOF(face_geometry)) = 1;
  WR2 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* SELF.bounds |
          'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.EDGE_LOOP' IN
                                          TYPEOF(bnds.bound)) |
         NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
             NOT('AIC_TOPOLOGICALLY_BOUNDED_SURFACE.EDGE_CURVE' IN
             TYPEOF(oe.edge_element))) = 0))) = 0;
  WR3 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* SELF.bounds |
          'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.EDGE_LOOP' IN
                                            TYPEOF(bnds.bound)) |
           NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
           NOT (SIZEOF (['AIC_TOPOLOGICALLY_BOUNDED_SURFACE.LINE',
                         'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.CONIC',
                         'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.POLYLINE',
                         'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.SURFACE_CURVE',
                         'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.B_SPLINE_CURVE'] *
              TYPEOF(oe.edge_element\edge_curve.edge_geometry)) = 1 )
                ) = 0))) = 0;
  WR4 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* SELF.bounds |
           'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.EDGE_LOOP' IN
                                             TYPEOF(bnds.bound)) |
            NOT(SIZEOF(QUERY (oe <* elp_fbnds.bound\path.edge_list |
             NOT((('AIC_TOPOLOGICALLY_BOUNDED_SURFACE.VERTEX_POINT' IN
                 TYPEOF(oe.edge_start)) AND
             ( 'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.CARTESIAN_POINT' IN
              TYPEOF(oe.edge_start\vertex_point.vertex_geometry)))) AND
                 (('AIC_TOPOLOGICALLY_BOUNDED_SURFACE.VERTEX_POINT' IN
                    TYPEOF(oe.edge_end)) AND
                  ( 'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.CARTESIAN_POINT' IN
                    TYPEOF(oe.edge_end\vertex_point.vertex_geometry)))
              ))) = 0))) = 0;
  WR5 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* SELF.bounds |
         'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.EDGE_LOOP' IN
                                           TYPEOF(bnds.bound)) |
```

```
            'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.ORIENTED_PATH' IN
             TYPEOF(elp_fbnds.bound))) = 0;
  WR6 : (NOT ('AIC_TOPOLOGICALLY_BOUNDED_SURFACE.SWEPT_SURFACE' IN
                                          TYPEOF(face_geometry))) OR
             (SIZEOF (['AIC_TOPOLOGICALLY_BOUNDED_SURFACE.LINE',
                       'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.CONIC',
                       'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.POLYLINE',
                       'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.B_SPLINE_CURVE'] *
                   TYPEOF(face_geometry\swept_surface.swept_curve)) = 1);
  WR7 : SIZEOF(QUERY (vlp_fbnds <* QUERY (bnds <* SELF.bounds |
          'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.VERTEX_LOOP' IN
                                           TYPEOF(bnds.bound)) |
             NOT(('AIC_TOPOLOGICALLY_BOUNDED_SURFACE.VERTEX_POINT' IN
               TYPEOF(vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex)) AND
                  ('AIC_TOPOLOGICALLY_BOUNDED_SURFACE.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(['AIC_TOPOLOGICALLY_BOUNDED_SURFACE.EDGE_LOOP',
         'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.VERTEX_LOOP'] * TYPEOF(bnd.bound))
                                 = 1))) = 0;
  WR9 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* SELF.bounds |
          'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.EDGE_LOOP' IN
                                           TYPEOF(bnds.bound)) |
            NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
            ('AIC_TOPOLOGICALLY_BOUNDED_SURFACE.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 ('AIC_TOPOLOGICALLY_BOUNDED_SURFACE.PCURVE' IN
            TYPEOF(sc_ag)))) = 0))) = 0))) = 0;
  WR10 : ((NOT ('AIC_TOPOLOGICALLY_BOUNDED_SURFACE.SWEPT_SURFACE' IN
         TYPEOF(face_geometry))) OR
         ((NOT ('AIC_TOPOLOGICALLY_BOUNDED_SURFACE.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 |
         'AIC_TOPOLOGICALLY_BOUNDED_SURFACE.EDGE_LOOP' IN TYPEOF(bnds.bound))
|
         NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
         ('AIC_TOPOLOGICALLY_BOUNDED_SURFACE.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 **lines**, **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.

*)
END_SCHEMA; -- end AIC_TOPOLOGICALLY_BOUNDED_SURFACE SCHEMA
(*

F.7 Application interpreted construct 512: Faceted boundary representation

F.7.1 Scope

This application interpreted construct specifies the interpretation of the integrated resources in order to satisfy the requirements for the description of a three dimensional shape by means of a boundary representation model with planar faces and implicit straight line edges; and for the composition of one or more such shapes as a **faceted_brep_shape_representation**.

The following are within the scope of this AIC:

— 3D geometry;

— B-reps;

— B-rep models;

— faceted B-reps;

— polyloops, used to bound faces in a faceted B-rep;

— unbounded geometry;

— use of topology to bound geometric entities;

— geometric transformations.

The following are outside the scope of this AIC:

— 2D geometry;

— bounded curves;

— curved surfaces including B-spline surfaces;

— offset curves and surfaces.

F.7.2 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 the EXPRESS short listing of this AIC.

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. References to eliminated constructs are outside the scope of the AIC.

This application interpreted construct provides a consistent set of geometric and topological entities for the definition of manifold solid models with planar faces and implicitly defined edges and vertices. The faces of the **B-rep model**s are bounded by **poly_loop**s and each face is required to have an explicit outer bound.

The highest level entity in this AIC is the **faceted_brep_shape_representation**. This is a **shape_-representation** (see: ISO 10303-41) consisting of **faceted_brep**s and **mapped_item**s defined as translated or

transformed copies of **faceted_brep**s.

```
*)
SCHEMA aic_faceted_brep;

  USE FROM geometry_schema(                                    -- ISO 10303-42
     cartesian_point,
     axis2_placement_3d,
     cartesian_transformation_operator_3d,
     elementary_surface,
     plane);

  USE FROM topology_schema(                                    -- ISO 10303-42
     poly_loop,
     face_surface,
     face_bound,
     face_outer_bound,
     closed_shell,
     oriented_closed_shell);

  USE FROM geometric_model_schema(                             -- ISO 10303-42
     brep_with_voids,
     manifold_solid_brep,
     faceted_brep);

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

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

(*
```

NOTES

2 - The entity **manifold_solid_brep** is explicitly interfaced (i.e. included in the USE FROM lists) to enable compilation of the function **msb_shell**s. The entity **elementary_surface** is explicitly interfaced to allow **faceted_brep_shape_representation** to access attributes of this entity.

3 - 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
topology_schema	ISO 10303-42
representation_schema	ISO 10303-43
product_property_representation_schema	ISO 10303-41

F.7.2.1 aic_faceted_brep schema entity definition: faceted_brep_shape_-representation

The **faceted_brep_shape_representation** is a subtype of shape representation in which the representation

items are specialisations of **faceted_brep** entities. These differ from the more general B-rep in having only planar faces and implicit edge geometry.

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

EXPRESS specification:

```
*)
ENTITY faceted_brep_shape_representation
 SUBTYPE OF (shape_representation);
 WHERE
 WR1 : SIZEOF (QUERY (it <* SELF.items |
          NOT (SIZEOF(['AIC_FACETED_BREP.FACETED_BREP',
            'AIC_FACETED_BREP.MAPPED_ITEM',
            'AIC_FACETED_BREP.AXIS2_PLACEMENT_3D'] *
             TYPEOF(it)) = 1))) = 0;
 WR2 : SIZEOF (QUERY (it <* SELF.items |
          SIZEOF(['AIC_FACETED_BREP.FACETED_BREP',
          'AIC_FACETED_BREP.MAPPED_ITEM'] * TYPEOF(it)) = 1)) > 0;
 WR3 : SIZEOF (QUERY (fbrep <* QUERY ( it <* SELF.items |
          'AIC_FACETED_BREP.FACETED_BREP' IN TYPEOF(it)) |
          NOT (SIZEOF (QUERY (csh <* msb_shells(fbrep, 'AIC_FACETED_BREP') |
           NOT (SIZEOF (QUERY (fcs <* csh.cfs_faces |
          NOT (('AIC_FACETED_BREP.FACE_SURFACE' IN TYPEOF (fcs)) AND
          (('AIC_FACETED_BREP.PLANE' IN TYPEOF
               (fcs\face_surface.face_geometry)) AND
          ('AIC_FACETED_BREP.CARTESIAN_POINT' IN TYPEOF (
               fcs\face_surface.face_geometry\
                    elementary_surface.position.location))))))
                      = 0))) = 0))) = 0;
   WR4 : SIZEOF (QUERY (fbrep <* QUERY ( it <* SELF.items |
          'AIC_FACETED_BREP.FACETED_BREP' IN TYPEOF(it)) |
          NOT (SIZEOF (QUERY (csh <* msb_shells(fbrep, 'AIC_FACETED_BREP') |
          NOT (SIZEOF (QUERY (fcs <* csh.cfs_faces |
          NOT (SIZEOF (QUERY (bnds <* fcs.bounds |
           'AIC_FACETED_BREP.FACE_OUTER_BOUND' IN TYPEOF(bnds)))
              = 1))) = 0))) = 0))) = 0;
   WR5 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
          'AIC_FACETED_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
          'AIC_FACETED_BREP.ORIENTED_CLOSED_SHELL' IN
              TYPEOF (msb\manifold_solid_brep.outer))) = 0;
   WR6 : SIZEOF (QUERY (brv <* QUERY (it <* SELF.items |
          'AIC_FACETED_BREP.BREP_WITH_VOIDS' IN TYPEOF(it)) |
          NOT (SIZEOF (QUERY (csh <* brv\brep_with_voids.voids |
           csh\oriented_closed_shell.orientation)) = 0))) = 0;
   WR7 : SIZEOF (QUERY (mi <*  QUERY (it <* SELF.items |
                'AIC_FACETED_BREP.MAPPED_ITEM' IN TYPEOF(it)) |
          NOT ('AIC_FACETED_BREP.FACETED_BREP_SHAPE_REPRESENTATION' IN
            TYPEOF(mi\mapped_item.mapping_source.mapped_representation))))
                = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: The **items** attribute of the **representation** supertype may contain **faceted_brep**s, **mapped_item**s and **axis2_placement_3d**s only.

WR2: At least one item in the **items** set shall be a **faceted_brep** entity, or a **mapped_item**, (see also WR7).

WR3: For each **faceted_brep** in the **items** set: the faces shall be **face_surface**s, the associated surface for each **face** shall be a **plane**, and each **plane** shall use a **cartesian_point** for its location.

WR4: An explicit outer bound is specified for each **face** of the **faceted_brep**.

WR5: For each **manifold_solid_brep** in the **items** set the outer shell attribute shall not be of the oriented subtype.

WR6: If the **manifold_solid_brep** is also a **brep_with_void**s then each **shell** in the voids set shall be an **oriented_closed_shell** with orientation value FALSE.

WR7: If a **mapped_item** is included in the **items** set then the **mapped_representation** of the **mapping_source** attribute shall be a **faceted_brep_shape_representation**.

NOTE 2 - If a **cartesian_transformation_operator_3d** is included as **mapped_item.mapping_target** with an **axis2_placement_3d** that corresponds to the original coordinate system as **mapped_representation.mapping_origin**, then the resulting **mapped_item** is a transformed copy of the **faceted_brep_shape_representation**. The precise definition of the transformation, including translation, rotation, scaling and, if appropriate, mirroring, is given by the transformation operator.

F.7.2.2 aic_faceted_brep function definition: msb_shells

This function determines the set of all **closed_shell**s used in the definition of a **manifold_solid_brep**. Special provision is made for the **brep_with_voids subtype**.

EXPRESS specification:

```
*)
FUNCTION msb_shells (brep: manifold_solid_brep;
                     schema_name : STRING) :
                     SET [1:?] OF  closed_shell;
    IF (schema_name + '.BREP_WITH_VOIDS' IN TYPEOF (brep)) 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: A STRING giving the name of the schema to which **brep** belongs.

result:(output) A SET of all the **closed_shell**s used to define **brep**.

*)
END_SCHEMA; -- end AIC FACETED BREP SCHEMA
(*

F.8 Application interpreted construct 514: Advanced boundary representation

F.8.1 Scope

This application interpreted construct specifies the interpretation of the integrated resources in order to satisfy the requirement for the definition of an advanced boundary representation model. An advanced B-rep model is a representation composed of one or more **manifold_solid_brep**s each of which is defined with elementary geometry or sculptured geometry.

This AIC is independent of any industrial application domain.

F.8.2 Itemized scope

The following are within the scope of this AIC:

— 3D geometry;

— B-reps;

— B-rep models;

— B-spline curves and surfaces;

— conics;

— elementary curves;

— elementary surfaces;

— geometric transformations;

— polylines;

— pcurves;

— sculptured surfaces;

— surface curves;

— swept surfaces;

— twisted curves;

— unbounded geometry;

— use of topology to bound geometric entities.

The following are outside the scope of this AIC:

— 2D geometry other than for the definition of a pcurve in the parameter space of a surface;

— bounded curves other than polylines and B-spline curves;

— bounded surfaces other than B-spline surfaces;

— offset curves and surfaces.

F.8.3 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 the EXPRESS short listing of this AIC.

NOTES

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. References to eliminated constructs are outside the scope of the AIC.

2 - This AIC uses all the entities and types from the topologically bounded surface AIC (**aic_topology_bounded_surface**). AIC 511 should be referred to in order to obtain the complete data set.

```
*)
SCHEMA aic_advanced_brep;

  USE FROM aic_topologically_bounded_surface;                    -- Annex F

  USE FROM geometry_schema(                                      -- ISO 10303-42
    cartesian_transformation_operator_3d,
    geometric_representation_context);

  USE FROM geometric_model_schema(                               -- ISO 10303-42
    manifold_solid_brep,
    brep_with_voids);

  USE FROM topology_schema(                                      -- ISO 10303-42
    closed_shell,
    connected_face_set,
    oriented_closed_shell);

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

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

(*
```

NOTES

3 - The **connected_face_set** entity is explicitly interfaced (i.e. included in the USE FROM lists) to allow rules in the **advanced_brep_shape_representation** entity to access attributes of this entity. For the use of this AIC, this entity may be instantiated as one of its subtypes.

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

aic_topologically_bounded_surface	Annex F
geometry_schema	ISO 10303-42
geometric_model_schema	ISO 10303-42
topology_schema	ISO 10303-42
representation_schema	ISO 10303-43
product_property_representation_schema	ISO 10303-41

F.8.3.1 aic_advanced_brep schema entity definition: advanced_brep_shape_representation

The **advanced_brep_shape_representation** is a subtype of **shape_representation** in which the representation items are specialisations of manifold solid brep entities. 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.

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

as an **advanced_brep_shape_representation**.

EXPRESS specification:

```
*)
ENTITY advanced_brep_shape_representation
  SUBTYPE OF (shape_representation);
  WHERE
     WR1: SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
               'AIC_ADVANCED_BREP.MANIFOLD_SOLID_BREP',
               'AIC_ADVANCED_BREP.FACETED_BREP',
               'AIC_ADVANCED_BREP.MAPPED_ITEM',
               'AIC_ADVANCED_BREP.AXIS2_PLACEMENT_3D'] * TYPEOF(it)) = 1)) ))
               = 0;
     WR2: SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
               'AIC_ADVANCED_BREP.MANIFOLD_SOLID_BREP',
               'AIC_ADVANCED_BREP.MAPPED_ITEM'] * TYPEOF(it)) = 1) )) > 0;
     WR3: SIZEOF(QUERY ( msb <* QUERY ( it <* SELF.items |
               ('AIC_ADVANCED_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) ) |
              ( NOT (SIZEOF(QUERY ( csh <* msb_shells(msb,
               'AIC_ADVANCED_BREP') | (NOT (SIZEOF(QUERY ( fcs <* csh\
               connected_face_set.cfs_faces | (NOT (
               'AIC_ADVANCED_BREP.ADVANCED_FACE' IN TYPEOF(fcs))) )) = 0)) ))
               = 0)) )) = 0;
     WR4: SIZEOF(QUERY ( msb <* QUERY ( it <* SELF.items |
            ( 'AIC_ADVANCED_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) ) |
            (  'AIC_ADVANCED_BREP.ORIENTED_CLOSED_SHELL' IN TYPEOF(msb\
              manifold_solid_brep.outer) )) = 0;
     WR5: SIZEOF(QUERY ( brv <* QUERY ( it <* SELF.items |
           ( 'AIC_ADVANCED_BREP.BREP_WITH_VOIDS' IN TYPEOF(it)) ) | (NOT
              (SIZEOF(QUERY ( csh <* brv\brep_with_voids.voids |
              ( csh\oriented_closed_shell.orientation)))  = 0)) ))
              = 0;
     WR6: SIZEOF(QUERY ( mi <* QUERY ( it <* SELF.items |
            ( 'AIC_ADVANCED_BREP.MAPPED_ITEM' IN TYPEOF(it)) ) | (NOT
            ( 'AIC_ADVANCED_BREP.ADVANCED_BREP_SHAPE_REPRESENTATION' IN
              TYPEOF(mi\mapped_item.mapping_source.mapped_representation))) ))
              = 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 only. A **faceted_brep** is excluded from the **items** set since this would be of type **faceted_brep** and of type **manifold_solid_brep**.

WR2: At least one item in the **items** set shall be a **manifold_solid_brep** entity or a **mapped_item** (see also WR6).

WR3: For each **manifold_solid_brep** in the **items** set each face shall be an **advanced_face**. This 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 edges used to define the boundaries of the **face** shall all be of type **edge_curve**.

d) Each **curve** used to define the geometry of the **face**s and face bounds shall be either a **conic**, or a **line** or a **polyline** or a 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 **items** set the outer shell attribute shall not be of the oriented subtype.

WR5: If a **brep_with_void**s is included in the **items** set 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 **items** set then the **mapped_representation** of the **mapping_source** attribute shall be an **advanced_brep_shape_representation**.

NOTE 2 - If a **cartesian_transformation_operator_3d** is included as **mapped_item.mapping_target** with an **axis2_placement_3d** corresponding to the original coordinate system as **mapped_representation.mapping_origin** then the resulting **mapped_item** is a transformed copy of the **advanced_brep_shape_representation**. The precise definition of the transformation, including translation, rotation, scaling and, if appropriate, mirroring, is given by the transformation operator.

F.8.3.2 aic_advanced_brep schema function definition: msb_shells

NOTE - The function below is common to all B-rep AICs.

This function determines the set of all **closed_shell**s used in the definition of a **manifold_solid_brep**. Special provision is made for the **brep_with_void**s subtype.

EXPRESS specification:

```
*)
FUNCTION msb_shells (brep: manifold_solid_brep;
                     schema_name : STRING) :
                     SET [1:?] OF  closed_shell;
```

```
      IF (schema_name + '.BREP_WITH_VOIDS' IN TYPEOF (brep)) 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: A STRING giving the name of the schema to which **brep** belongs.

result: (output) A SET of all the **closed_shell**s used to define **brep**.

*)
END_SCHEMA; -- end AIC_ADVANCED_BREP SCHEMA
(*

F.9 Application interpreted construct 515: Constructive solid geometry

F.9.1 Scope

This part of ISO 10303 specifies the interpretation of the integrated resources to satisfy requirements for representation of product shape using constructive solid geometry.

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

— solid primitives such as sphere, right circular cone, right circular cylinder, and torus;

— regularised boolean operations of union, intersection, and difference on solid primitives and general solids such as advanced and faceted boundary representation (B-rep) solids, half-space solids, and swept area solids (extruded area solids, revolved area solids);

— boolean results generated by applying operators to solids.

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

— 2D geometric entities;

— self-intersecting geometry;

— evaluation of csg models to generate B-rep models.

F.9.2 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. References to eliminated constructs are outside the scope of the AIC.

```
*)
SCHEMA aic_csg;

USE FROM aic_advanced_brep;

USE FROM aic_faceted_brep;

USE FROM geometric_model_schema
     (block,
      boolean_operand,
      boolean_operator,
      boolean_result,
      box_domain,
      boxed_half_space,
      csg_primitive,
      csg_select,
      csg_solid,
      half_space_solid,
      right_angular_wedge,
      right_circular_cone,
      right_circular_cylinder,
      solid_replica,
      sphere,
      torus,
      extruded_area_solid,
      revolved_area_solid);

USE FROM geometry_schema
     (geometric_representation_item,
      geometric_representation_context,
      axis1_placement,
      axis2_placement_3d,
      cartesian_point,
      direction);

USE FROM representation_schema(mapped_item);

USE FROM
product_property_representation_schema(shape_representation);

(*
```

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

aic_advanced_brep	ISO 10303-514
aic_faceted_brep	ISO 10303-512
geometric_model_schema	ISO 10303-42
geometry_schema	ISO 10303-42
representation_schema	ISO 10303-43
product_property_representation_schema	ISO 10303-41

F.9.2.1 aic_csg entity definition: csg_shape_representation

A **csg_shape_representation** is a three-dimensional **shape_representation** which repesents the shape of a **product** using constructive solid geometry operations and techniques. The solid objects used may be primitives, faceted B-rep solids, or advanced B-rep solids.

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

EXPRESS Specification:

```
*)
ENTITY csg_shape_representation
 SUBTYPE OF (shape_representation);

   WHERE
    WR1: SELF.context_of_items\
  geometric_representation_context.coordinate_space_dimension = 3;

    WR2 : SIZEOF (QUERY (it <* SELF.items | NOT
     (SIZEOF(['AIC_CSG.CSG_SOLID', 'AIC_CSG.EXTRUDED_AREA_SOLID',
     'AIC_CSG.REVOLVED_AREA_SOLID', 'AIC_CSG.SOLID_REPLICA',
     'AIC_CSG.MAPPED_ITEM', 'AIC_CSG.AXIS2_PLACEMENT_3D'] * TYPEOF(it)) =
     1 ))) = 0;

    WR3 : SIZEOF (QUERY (it <* SELF.items |
     SIZEOF(['AIC_CSG.CSG_SOLID', 'AIC_CSG.EXTRUDED_AREA_SOLID',
     'AIC_CSG.REVOLVED_AREA_SOLID','AIC_CSG.SOLID_REPLICA',
     'AIC_CSG.MAPPED_ITEM']*TYPEOF(it)) = 1 )) >= 1;

    WR4 : SIZEOF (QUERY (it <* SELF.items |
     ('AIC_CSG.MAPPED_ITEM' IN TYPEOF(it)) AND NOT
     (SIZEOF(['AIC_CSG.CSG_SHAPE_REPRESENTATION',
              'AIC_CSG.FACETED_BREP_SHAPE_REPRESENTATION',
              'AIC_CSG.ADVANCED_BREP_SHAPE_REPRESENTATION']*
     TYPEOF(it\mapped_item.mapping_source.mapped_representation)) =1) ))
     = 0;

   END_ENTITY;
END_SCHEMA;
(*
```

Formal propositions:

WR1: The **geometric_representation_context** of the **csg_shape_representation** shall have a **coordinate_space_dimension** equal to three.

WR2: The items in a **csg_shape_representation** shall be a **csg_solid**, **extruded_area_solid**, **revolved_area_solid**, **solid_replica**, **mapped_item**, or an **axis2_placement_3d**.

WR3: At least one of the items in a csg_shape_representation shall be a **csg_solid, extruded_area_solid, revolved_area_solid, solid_replica,** or a **mapped_item**.

WR4: If a **mapped_item** is in the items set then the **mapped_representation** of the mapping_source attribute shall be a **csg_shape_representation**, a **faceted_brep_shape_representation**, or an **advanced_brep_shape_representation**.

Annex G
(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 definitions of the activities and the data and a set of activity figures. The AAM covers activities that go beyond the subject of this application protocol.

The viewpoint of the application activity model is that of a sheet metal die designer.

The diagrams use the IDEF0 activity modelling format.

NOTES - The IDEF0 language is described in [2, annex N].

G.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.

Following each term definition, in parentheses, is the location of the term in the diagram. For activities, the node number is given. For ICOMs, the node number of the highest level activity or activities with which it interacts is given, followed by the ICOM number designation for that node.

ICOM number designations are the letter "I" for Input, "C" for Control, "O" for Output, or "M" for Mechanism, followed by a number. By referring to any of the AAM diagrams, it may be seen that inputs are arrows entering an activity box, or node, at the left side of the node. These are assigned numbers sequentially from top to bottom, for instance, I1, I2, I3. Controls are arrows entering the top of a node, and are numbered from left to right. Outputs exit the right side of a node, and are numbered from top to bottom. Mechanisms are arrows entering the bottom of a node, and are numbered from left to right.

G.1.1 applicable standards (A21 C5, A232 C4): a reference to established standards, procedures, and guidelines to be followed during the design of die assemblies or die components to insure adequacy in terms of safety, timing, strength, quality, appropriate use of standard components including functionality thereof, subassemblies, interference avoidance, CAD policies, NC machining requirements, production standards, manufacturability standards and plant, division or corporate conventions. These may be referenced by document id, section, and page number.

G.1.2 communicated part design change request (A0 O5): a request for any change to be made to the part design, required to facilitate the manufacturability or operation of the die, the production of the sheet metal part, or the pressline automation. These include all the supporting material required to document the problem, and the status: whether or not it was accepted by the intended department and how far it has progressed through the process.

This is the same as a part design change request, except for the fact that it is communicated by those involved with the part design to those involved in engineering the die and producing die manufacturing data.

G.1.3 communicated die design change request (A221 O3, A222O3): a request for any change to be made to the die design, required to facilitate the manufacturability or operation of the die, the production of the sheet metal part, or the pressline automation. These include all the supporting material required to document the problem, and the status: whether or not it was accepted by the intended department and how far it has progressed through the process.

This is the same as die design change request, except for the fact that it is communicated by those involved with the die design to those involved in engineering the die and producing die manufacturing data.

G.1.4 conceptual die design (A23 I2): rough design of die, indicating major functional components and other cast portions of the die, and possibly including the specification of the sheet metal to be stamped therein. Several alternatives may be prepared for review and approval by the customer. The conceptual die design may include the following information: conceptual design drawings, design parameters, automation criteria, die or component strength requirements, design improvements and modifications, tool design requirements, part orientation, quick change information, rough dimensions, pad type, die material, size or weight requirements, safety requirements, and concept design problems. Design improvements and modifications are a result of design process iterations. Design parameters are definitions of functionality and type of die required including assessment of reusable die design data, which is existing data from previous die designs and externally originated components, such as standard components. These design parameters may include die clearance requirements, die working height, scrap shedding angles, scrap cutter locations, panel location, tipping finger, panel sensor locations, interference curves, holding, and stripping pressure.

It may include data ownership information, such as: name, phone number, department, company; date of creation or processing; who has authority to access or modify; and an ownership history. It may also include data source information, such as: System created on if not manual; file and directory name if not manual; and perhaps some information on the history of the file, and data on where the file has been or will be sent.

G.1.5 conceptual die design data (A22 O4): formability quality criteria, and the outline form of the die design, related in-process sheet metal, and process plans. Formability quality criteria are those criteria which

must be met by a die design in order to ensure that the resulting sheet metal product is of acceptable quality. The ICOMs within conceptual die design data are:

— conceptual die design;

— die face design;

— formability quality criteria;

— part process package;

— work in process design;

— pressline sheet metal input specification;

— enhanced part process plan.

G.1.6 conceptualize part manufacture and die function (A22): creation of, identification of the requirements for, and analysis of die design and process alternatives, that are then used to arrive at a conceptual die design and sculpted surfaces.

G.1.7 conceptualizer die design change request (A22 O1): a request for any change to be made to the die design, required to facilitate the manufacturability or operation of the die, the production of the sheet metal parts, or the pressline automation. These include all the supporting material required to document the problem, and the status: whether or not it was accepted by the intended department and how far it has progressed through the process.

This is the same as a die design change request, except the manufacturing person may conceptualize what the design should be and communicate this back and forth to the die designer.

G.1.8 design approval (A21 O3): authorization by a managerial authority that a die face, die structure design, a process plan, or all of the above have achieved a designated state of completion and correctness.

G.1.9 design die (A23): the design of die components and die assemblies for those portions of the die that must be manufactured, and integration of those designs with the portions of the die already available. Die components include functional, structural or support, and internal die automation elements. Die assemblies can consist of the entire assembly or of one or more subassemblies.

G.1.10 design die component (A231): the design of functional, structural or support, and internal die automation elements. Die components may either be individually detailed stand-alone components, or subassemblies.

G.1.11 design die face (A222): creation or identification of all of the data for those surfaces of a die that

contact the sheet metal being formed in a die. These surfaces are sometimes known as sculpted, or active surfaces. This activity includes a determination of the formability, that is out of the scope of this AP, and the tipping angle of the sheet metal part to be formed by the die, that is in scope.

G.1.12 design material handling * (A232): the design of the automated portion of material handling within a pressline, but external to the dies.

G.1.13 design part * (A1): design of a sheet metal part for which sheet metal dies will subsequently be required.

G.1.14 design reviewer (A0 M4): individual with authority to review a design for conformance to requirements.

G.1.15 detailed die design data (A23 O2): the detailed, complete design of the die and peripheral material handling. The contents of detailed die design data are:

— die assembly design;

— die component design;

— die quality characteristics;

— pressline automation selections.

G.1.16 die assembly design (A233 O2): the complete design of an assembled die, including the assembled die bill of materials, that may consist of one or more die subassembly designs. This complete design includes press information: press type, press force (tonnage), style, size, location, Joint Industry Conference (JIC) specification; handling equipment and press line information; scrap disposal constraints such as where scrap exits, and maximum scrap size allowed; indications of where sculpted and planar areas are to be machined, as indicated by surface finish marks on drawings or layers or colors on CAD drawings; features such as draw or lock beads, lock steps, sausages, and metal gain devices; overdraws, trim lines; and the locations of laid out flanges. There may be references to subsequent die operations: operation Identifications and approach angles. And the in-process stamping shape may be referenced or included.

It includes data ownership information, such as: name, phone number, department, company; date of creation or processing; who has authority to access or modify; and an ownership history. It includes data source information, such as: System created on if not manual; file and directory name if not manual; and possibly information on the history of the file, and data on where the file has been or will be sent.

G.1.17 die component design (A231 O4): the design of die components that are assembled to form a die assembly or subassembly, including the following: the bill of materials for the component; indications of where sculpted and planar areas are to be machined as indicated by surface finish marks on drawings or layers or colors on CAD drawings; features such as draw or lock beads, lock steps, sausages, and metal gain devices;

overdraws, trim lines; and the locations of laid out flanges. The in-process stamping shape may be referenced.

Data ownership information is included, such as: name, phone number, department, company; date of creation or processing; who has authority to access or modify; and an ownership history. Also included is data source information, such as: System created on if not manual; file and directory name if not manual; and perhaps some information on the history of the file, and data on where the file has been or will be sent.

Die components may be functional components such as steels, pads, cam components, adapters, die, punches, binders; die structural or support components such as die shoes, die heels; scrap handling components; internal automation components; or fastening selections such as screws and dowels. This design may include nominal casting raw material design. A die component may also be a subassembly.

G.1.18 die design (A21 C1): the designs of the die components and their assembly. This includes the definitions of: the die face, sometimes referred to as the active surfaces; the die layout, or sculpted surfaces; castings, steels, gages, guidance systems, lifters, shoes, pressure systems, externally originated components such as standard components, cam mechanisms, section layouts, and upper or lower components.

The die design may include effectivity of the design, the part it produces, or of any portion of the die itself stated as the volume change or coordinated change date, the number of weeks before production, or the production year by or for or during which the design is to be in use. The die design includes the location within the die of the material to be formed; the version number of the part to be formed and of the die design itself; a bill of materials for the die; accumulated engineering change notes and change identifications; and specifications of the areas to be machined.

Each die design includes a status and approval. Each die design also includes data ownership information, such as: name, phone number, department, company; date of creation or processing; who has authority to access or modify; and an ownership history. And each die design includes data source information, such as: System created on if not manual; file and directory name if not manual; and perhaps some information on the history of the file, and data on where the file has been or will be sent.

G.1.19 die design and related data (A0 O2): the cumulative set of die design data, including preliminary through final work and documenting modifications made along the way. The die design and related data consists of preliminary die design, released die design, and these ICOMs:

— conceptual die design data;

— detailed die design data;

— die design change notification;

— die design.

Preliminary die design is a die design that is not yet approved or is incomplete, that is used as information only and is subject to change.

Released die design is a completed die design approved for release from the die design department. It may include die face design, functional component designs, die structural or support component designs, component bills of material including die material specification, and die assembly design. It includes data ownership information, such as: name, phone number, department, company; date of creation or processing; who has authority to access or modify; and an ownership history. It includes data source information, such as: system created on if not manual; file and directory name if not manual; and perhaps some information on the history of the file, and data on where the file has been or will be sent.

G.1.20 die design change notification (A21 O2): notification that the design of either a die component or a complete die has changed since its prior release.

G.1.21 die design change request (A0 I3): a request for any change to be made to the die design, required to facilitate the manufacturability or operation of the die, the production of the sheet metal parts, or the pressline automation. These include all the supporting material required to document the problem, and the status: whether or not it was accepted by the intended department and how far it has progressed through the process.

This is the same as a conceptualizer die design change request, except it may not originate with those involved in conceptualizing part manufacture and die function.

G.1.22 die design library (A21 C6, A231 C7, A233 C7): a reference to the collection of existing externally originated and standard component designs, reusable die designs, externally originated and standard fastening component designs, and pressline and internal automation components designs. Also may contain standard drawing components such as borders, notes, and title blocks. The library consists of all the various media such as tapes, engineering drawings, and electronic files, that can contain the information. The use of such a library is to reduce design time by providing proven past work for incorporation in or checking of present work.

G.1.23 die design request (A0 I1): request for a die design to be created. It may include the scope of the work in a verbal description; a purchase order number; the name, organization, and appropriate management and accounting sign offs of the requesting party and of the planner; part numbers, production rates and volumes; the type, serial numbers, quantity, and priority of die required; the delivery date, time, or both; the payment scenario; the required media of tape, drawing, or disk; manual or electronic mode of creation; which CAD system to use; and which output format. This request may be associated with a maximum cost, but such would not appear on the purchase order.

G.1.24 die designer (A0 M6): one who designs or who mechanically or electronically drafts the designs for sheet metal dies.

G.1.25 die face design (A23 I1, A222 O5): the math data of the face of the die including the tipping angle, if necessary. The face of the die consists of those surfaces that contact the metal during the forming operation:

addendum, developed, active, and sculpted surfaces are terms used to describe them. This math data may take the form of a point array with a reference system, as is obtained by scanning in or gathering by a CMM machine from a hard master, and may be supplemented by a reference to a hard aid or master, which is a physical model data source.

Elements of the die face design may include features such as draw or lock beads, lock steps, sausages, and metal gain devices; overdraws, trim lines; the locations of laid out flanges and scrap handling. There may be references to subsequent die operations: operation identifications and approach angles. And the in-process stamping shape may be referenced or included.

It includes data ownership information, such as: name, phone number, department, company; date of creation or processing; who has authority to access or modify; and an ownership history. It includes data source information, such as: System created on if not manual; file and directory name if not manual; and perhaps some information on the history of the file, and data on where the file has been or will be sent.

G.1.26 die process planner (A0 M5): one who creates or checks die process plans.

G.1.27 die quality characteristics (A231 O3): the key features on the part and the die that can be used to verify the quality of the die. They include the quality of surfaces, die fixture holding location, and specific points to check on the die, typically with a CMM. The specific points to check include tolerances, dimensions, matching surfaces, master control surfaces or datums, part material, static load, overcrown or planar overbend, springback, placed out or linear overbend, and flange lines.

G.1.28 draft work in process design (A221 O5): a preliminary work in process design that outlines the part design in a specific stage of the manufacturing process. Each part design may have many work in process designs. A work in process design includes data ownership information, such as: name, phone number, department, company; date of creation or processing; who has authority to access or modify; and an ownership history. It also includes data source information, such as: System created on if not manual; file and directory name if not manual; and perhaps some information on the history of the file, and data on where the file has been or will be sent.

G.1.29 engineer die (A2): design of the dies required to produce the sheet metal part, including managing the die design, developing the conceptual die design, designing the die and die components, analyzing the die design, and designing the external automation.

G.1.30 enhanced part process plan (A221 O4): the process steps for manufacturing the sheet metal part based on the type of die required, with the addition of information to enable die design. The information may include items such as: material specifications; press force, size, features, type such as toggle versus transfer; operational sequence sketch or mickey mouse diagram; name, description, and tool numbers of dies; part name, number, tipping angle, orientation, quality, grain direction, surface finish; version, change level and change history for part and die; planner name, supervisor name, approval; date of process plan.

Data ownership information is included, such as: name, phone number, department, company; date of creation

or processing; who has authority to access or modify; and an ownership history. Also included is data source information, such as: System created on if not manual; file and directory name if not manual; and perhaps some information on the history of the file, and data on where the file has been or will be sent.

G.1.31 estimator (A0 M2): individual who refines a part process plan by estimating time, cost, press requirements, and materials required.

G.1.32 integrate die design (A233): unification of the individual component designs and verification that these conform to each other as well as to the given envelope.

G.1.33 invoice (A0 O1): a bill for goods delivered, expenses incurred, services rendered or all of the above. In the context of this AP, the goods are one or more of: part process plan, die face design, and die structure design; and the services are the time taken to develop those goods.

NOTE - Invoice is in scope only with respect to non-financial data.

G.1.34 libraries (A0 C2): references to part data and process data from previous designs and plans. Libraries includes:

— die design library;

— pressline interface specifications library;

— prototype and developmental die information.

G.1.35 manage die design (A21): the assignment and monitoring of personnel and resources, maintenance of a work or release schedule for the die design process, checking of die designs to verify that they conform to the proper standards and the corresponding part design, and approval of a die design release status.

G.1.36 managerial data (A0 O3): a report of the operating costs, including estimated versus actual costs; a schedule of the work required to complete a die design, including die design release schedule that contains required date to start, complete, or both, preliminary review date, release dates of related data, and intermediate milestones; and work assignment package.

G.1.37 mechanical handling standards (A232 C4): a reference to standard criteria that guide mechanical handling design to ensure its adequacy in terms including those of safety, timing, strength, quality, appropriate use of standard components and subassemblies, interference avoidance, plant or division or corporate conventions, and scrap handling.

G.1.38 NC programmer (A0 M1): one who generates machine control data for the creation of die face designs, based on supplied designs.

G.1.39 part design change request (A3 O1): a request for any change to be made to the part design required to facilitate the manufacturability or operation of the die, or the pressline information impacted by the production of the sheet metal part. These include all the supporting material required to document the problem, and the status: whether or not it was accepted by the intended department and how far it has progressed through the process.

This is the same as the communicated part design change request, except this is the request that results from problems encountered by those involved with engineering the die or producing die manufacturing data, and is communicated to those involved with the part design.

G.1.40 part design data (A0 O4): all of the information available to die designers on the design of the part. Consists of Part Design Change Notification, that is notification of changes since the last part design release, used to inform effectors of downstream activities so they can adjust their process accordingly or even halt further development if a change is significant; and the following:

— part design package;

— part quality characteristics.

G.1.41 part design package (A23 C4): a reference to that part design data that needs to be known by the die designers and part process planners who are designing dies and material handling to create the part.

This package includes the part data, approval or release sign off, part process data, part revision level and change information, flat pattern part design, and selected material and formability information. It also includes the master surface data, that is part design sculpted surface data intended to be re-used in the design of the dies to make said parts.

It may include effectivity of the part design, stated as the volume change or coordinated change date, the number of weeks before production, or the production year by or for or during which the design is to be in use. Includes indication of the location within the die of the part to be formed; the part number and version number of the part to be formed; a bill of materials for the part, likely a single entry; accumulated engineering change notes and change identifications; the orientation, grain direction, surface treatment, surface finish, material specification, and allowable alternative material specification of the material from which the part is to be formed; and indications of the areas to be machined.

If the part design has a CAD representation, that CAD design additionally contains: dimensions, tolerances, section views, view identifiers, line labels, breaklines, feature lines, matching surfaces, right or left hand id, weld spots, principal locating sections, feature callouts adhesive locations, body grid lines; title block, approvals, designer, supervisor, department; and relevant dates.

It includes data ownership information, such as: name, phone number, department, company; date of creation or processing; who has authority to access or modify; and an ownership history. It includes data source information, such as: System created on if not manual; file and directory name if not manual; and perhaps

some information on the history of the file, and data on where the file has been or will be sent.

G.1.42 part process package (A23 C1): contains the design of a sheet metal part before every die operation, and the process plan that will carry it all out. The contents of a part process package include:

— enhanced part process plan;

— pressline sheet metal input specification;

— work in process design.

G.1.43 part process planner (A0 M3): one who creates or checks part process plans.

G.1.44 part process plan change request (A0 O6): a request for change to a part process plan. Such a request originates with a die designer who determines that the design of a die component, design of material handling within and peripherally to a die or integration of the complete die assembly design requires a change in the way a part is processed. This change could be one that would correct a manufacturing difficulty, such as excessive die wear or metal thinning; or a change could reduce cost or effort.

G.1.45 part process plan draft (A0 I2): a draft version of the factory floor work instructions for manufacture of the sheet metal part, not including the process steps for designing and manufacturing the sheet metal die. Includes the number of dies, their functions and sequence, possible presslines, and proposed changes. The draft plan may also refer to the basis sheet metal grain direction, surface finish, quality, and orientation; and may have a standard plan id to identify the generic template from which it was derived.

G.1.46 part quality characteristics (A23 C4): data related to the quality of the part, used to verify the quality of designed and completed dies. This data includes tolerances, dimensions, surface class, matching surfaces, part features, master control surfaces or datums, required overcrown or planar overbend, and placed out or linear overbend.

G.1.47 prepare for sheet metal production (A0): preparations to make a sheet metal production part, including the preparation of the part design, the engineering of a die to produce the part, and the development of the data required to manufacture the die.

G.1.48 pressline automation selections (A232 O3): selected values for the pressline parameters or requirements. Pressline automation is commonly referred to as being designed to, meaning that the equipment, such as conveyer belts and robots, is reusable and needs only to be reconfigured, as opposed to being redesigned and built from scratch. This eventually becomes part of Released Die Design.

G.1.49 pressline interface specifications library (A221 C3, A232 C8): a reference to the collection of specifications for part-handling and part stamping machines that may be used to determine how they may be interfaced with each other to create an integrated and synchronous pressline. This collection may include plant constraints; for instance, crane capacity may limit the weight allowed for a die.

The press information may include the press types, press force, styles, sizes, locations, and Joint Industry Conference (JIC) specifications; plus scrap handling constraints. This library may also include press line parameters, pressline capabilities, and requirements. Pressline parameters include transfer height, shut height, mechanical handling and transfer of the part, and interferences between adjacent stations. Pressline requirements are used for press selection, designing the die, and checking of the die design.

G.1.50 pressline sheet metal input specification (A221 O4): the description of the physical shape and substance of the material that must be procured to feed into the pressline. Possible values for this item include planar or nonplanar sheet metal blanks and strip rolls of sheet metal in specific widths and thicknesses.

This description may have included with it data ownership information, such as: name, phone number, department, company; date of creation or processing; who has authority to access or modify; and an ownership history. It may also include data source information, such as: system created on if not manual; file and directory name if not manual; and perhaps some information on the history of the file, and data on where the file has been or will be sent.

G.1.51 produce die manufacturing data (A3): the development of the data required to manufacture the die components and the assembled die. This includes the development of NC and casting data.

G.1.52 prototype and developmental die information (libraries) (A222 C4): the library information related to prototype die designs and in-process die designs. It includes information derived from hard aids such as plaster, wood, or clay models.

G.1.53 refine part manufacturing process (A221): detailing of the machine and process requirements for manufacturing a sheet metal part based on the type of die required. This includes a determination of the number and sequence of processes, press force requirements, die process requirements, pressline sheet metal input to the process, and part material handling requirements.

G.1.54 specifications and standards (A0 C1): any standard applicable to die design. Specifications and standards includes cost standards,
die material specifications, manufacturing equipment lists, manufacturing standards, part design standard criteria, part material specifications, and the following:

— applicable standards;

— mechanical handling standards.

Cost standards are the costs of typical die design tasks, often developed by industrial engineering. Die material specifications are specifications of various materials and including their quality, thickness, coatings, and properties used in the die design process, including die material tables, textual description and specification source. Manufacturing equipment lists are lists of the existing manufacturing equipment used for the fabrication of dies, including the location of the equipment in a given enterprise, their process capabilities, and their availability.

Manufacturing standards are a collection of established manufacturing procedures, specifications, and parameters, along with NC documentation and reference manuals, used for generating manufacturing data and manufacturing parts. Part design standard criteria are guidelines and restrictions developed through experience or experiments, that are used by the design staff to increase efficiency and avoid repeated design errors. Part material specifications are data for available part materials including the physical and chemical properties, plus acceptable alternate materials.

G.1.55 work assignment package (A22 C2, A23 C3): information required to schedule the creation of a complete die design. It consists of a Die Design Work Schedule, that is a schedule of the work required to complete a die design, including die design release schedule, that in turn contains required date to start or complete, preliminary review date, release dates of related data, and intermediate milestones; a work schedule for this particular assignment, description of tasks to be done, required start and end dates, applicable release dates, and references to applicable documents, personnel assignments, and resource assignments.

G.1.56 work in process design (A222 O4): the work in process design is the design of a sheet metal part before every die operation except the initial blanking operation. It may be used to check each die. Each part design may have many such work in process designs. The work in process design may include flat pattern designs. No work in process design exists that corresponds to the output of the final manufacturing step. The design at this stage is the part design as provided in the part design package.

Elements of the work in process design may include features such as draw or lock beads, lock steps, sausages, and metal gain devices; overdraws, trim lines; the locations of laid out flanges and scrap handling. There may be references to subsequent die operations: operation identifications and approach angles.

A work in process design includes data ownership information, such as: name, phone number, department, company; date of creation or processing; who has authority to access or modify; and an ownership history. It includes data source information, such as: System created on if not manual; file and directory name if not manual; and perhaps some information on the history of the file, and data on where the file has been or will be sent.

G.2 Application activity model diagrams

The application activity model diagrams are given in figures G.1 through G.5. The graphical form of the application activity model is presented in the IDEF0 activity modelling format. Activities and data flows that are out of scope are marked with asterisks. Data flows which use the ampersand character are a concatenation of the indicated data flows.