INTERNATIONAL STANDARD

ISO 10303-55

First edition
2005-02-01

Industrial automation systems — Product data representation and exchange —

Part 55:
Integrated generic resource: Procedural and hybrid representation

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

Partie 55: Ressources génériques intégrées — Représentation procédurale et hybride

Reference number
ISO 10303-55:2005(E)

Contents

Page

Figures

Table

Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

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

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

ISO 10303-55 was prepared by Technical Committee ISO/TC 184/SC 4, *Industrial automation systems and integration*, Subcommittee SC 4, *Industrial data*.

ISO 10303 consists of a series of parts, under the general title *Industrial automation systems and integration — Product data representation and exchange*. The structure of ISO 10303 is described in ISO 10303-1.

Each part of ISO 10303 is a member of one of the following series: description methods, implementation methods, conformance testing methodology and framework, integrated generic resources, integrated application resources, application protocols, abstract test suites, application interpreted constructs, and application modules. This part is a member of the integrated generic resources series. The integrated generic resources and the integrated application resources specify a single conceptual product data model.

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

`<http://www.tc184-sc4.org/titles/STEP_Titles.htm>`

Should further parts of ISO 10303 be published, they will follow the same numbering pattern.

Introduction

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

This part of ISO 10303 is a member of the integrated resources series. Major subdivisions of this part of ISO 10303 are:

— Procedural model schema;

— Procedural shape model schema.

This part of ISO 10303 provides general mechanisms for the representation of models defined in terms of the operations used to construct them. The constructional operations themselves are represented by entity data types defined in other parts of ISO 10303, interpreted as constructors. Procedural models have the advantage of being easy to edit, simply by changing values of parameters used as arguments of their constructional operations. Such models are said to embody *design intent* information, in the sense that modifications to them conform to the scheme of parameterization imposed by their original creator, and also comply with any constraints implied by the particular constructional operations used. Thus the transfer of a procedural model into a receiving system carries with it information as to how the model will behave when edited following the transfer.

However, procedural models also have the disadvantage of containing (in their purest form) little or no explicit information about the result of actually performing the sequence of operations. This fact makes them unsuitable as a basis for the automation of many engineering processes that depend on the use of explicit geometric information, for example numerically controlled machining or inspection.

Systems for engineering purposes commonly achieve the advantages of both modelling approaches through the use of a *dual* representation, comprising a primary representation of the *procedural* or *construction history* type together with a secondary explicit representation. Other ISO 10303 resources provide the elements needed for explicit represenions. This part of the standard not only specifies resources for procedural representations but also provides a dual model capability by enabling the association of such a model with its corresponding explicit counterpart.

The initial focus of this part of ISO 10303 was to allow the capture and exchange of CAD shape representations of the procedural and hybrid types (a *hybrid representation* is basically procedural but also contains some explicit elements). However, the capabilities provided also have general applicability for the transfer of any type of procedurally represented or hybrid model, whether geometric or non-geometric. In the case of shape models, ISO 10303-42 is the primary resource for the corresponding explicit representations.

Because procedural representations are inherently parametric, they can be edited by changing the values of input arguments of constructional procedures. However, this requires that the system operator has an appropriate level of understanding of the rationale underlying the original constructional method. At the time of writing, no method is known for capturing design rationale information automatically during model construction, and provision is therefore made in this part of ISO 10303 for its representation as descriptive text, assumed to be supplied by the original designer.

It is useful to emphasize the distinction between design intent and design rationale. *Design intent* is captured in the schemes of parameterization and constraints imposed upon models during their construction.

It therefore governs the ways in which a model may be edited. *Design rationale*, on the other hand, is concerned with the reasons *why* a particular configuration or constructional process was adopted, and therefore with the logic underlying the design intent.

The industry motivation for the exchange of procedural, hybrid and dual representations arises from the difficulties that have been encountered in the editing of ISO 10303 explicit models in a receiving system, following a model transfer. If only an explicit model is transferred, as in the past, the design intent embodied in the procedural component of the dual model in the sending system is lost in the transfer. The consequences are that received model is incomplete in vital respects, and that editing it is difficult or impossible.

Three books and a conference paper providing further background on the topics covered by this part of ISO 10303 are given in the Bibliography [6 – 9].

The contents of the two schemas making up this part of ISO 10303 are as follows:

procedural_model_schema: Fundamental mechanisms for the representation of procedural and hybrid models, and for the capture of design rationale.

procedural_shape_model_schema: Specialization of the foregoing schema for the specific case of geometric models.

The relationships of the schemas in this part of ISO 10303 to other schemas that define the integrated resources of ISO 10303 are illustrated in Figure 1 using the EXPRESS-G notation. EXPRESS-G is defined in annex D of ISO 10303-11. The schemas occurring in Figure 1 are components of ISO 10303 integrated resources, and they are specified in the following resource parts:

`product_property_representation_schema`	ISO 10303-41
`support_resource_schema`	ISO 10303-41
`geometric_model_schema`	ISO 10303-42
`geometry_schema`	ISO 10303-42
`topology_schema`	ISO 10303-42
`representation_schema`	ISO 10303-43
`variational_representation_schema`	ISO 10303-108

NOTE 1 A procedural model is a representation of a constructional process, and it may therefore be envisaged that ISO 10303-49 (‘Process structure and properties’) [1] would be a suitable underlying resource for this part of ISO 10303. However, the definition of ‘process’ as given in ISO 10303-49 is a narrow one:

process: a particular procedure for doing something involving one or more steps or operations. The process may produce *a product, a property of a product, or an aspect of a product.*

Thus the ISO 10303-49 view of a process is one that is concerned with the generation of a physical object or some characteristic of it. The purpose of this part of ISO 10303, by contrast, is to provide the means for capturing and transferring constructional processes for *representations or models of general objects,* which only exist as abstractions in a computer or database. For this reason, and also because advantage can be taken of the very close relationship between procedural modelling operations and existing entities defined in other ISO 10303 integrated resources, ISO 10303-49 has not been used as the basis for the present part of ISO 10303.

NOTE 2 In the diagram on the following page, the schemas occurring in this part of ISO 10303 are enclosed in a heavy rectangular box. The specific entities interfaced are not indicated.

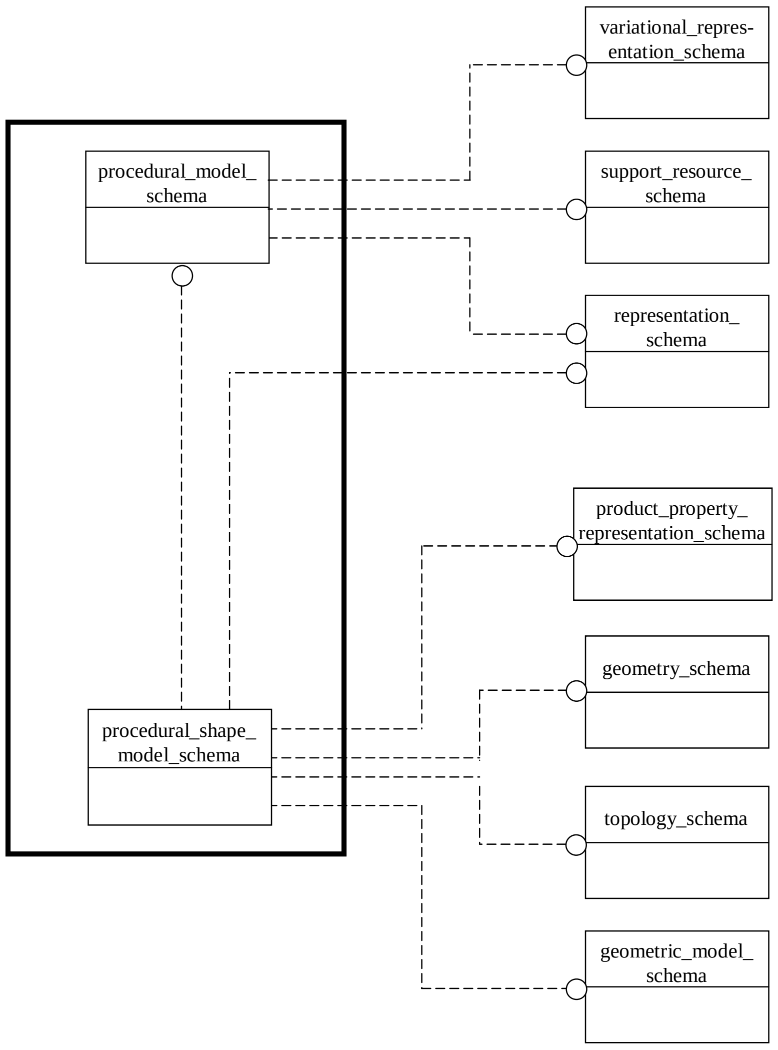

Figure 1 – Schema level diagram of relationships among ISO 10303-55 schemas (inside the box) and other resource schemas

Industrial automation systems and integration — Product data representation and exchange — Part 55: Integrated generic resource: Procedural and hybrid representation

1 Scope

This part of ISO 10303 specifies resource constructs for the representation of models of the procedural or construction history type, defined in terms of the sequence of constructional operations used to build them. Representations of the operations themselves are not specified here; the mechanisms provided in this document allow the use of entity data types defined in other parts of ISO 10303 for that purpose (see clause 4.2.5).

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

— The specification of sequences of constructional operations for the generation of any kind of explicit representation or model;

— The hierarchical structuring of constructional sequences;

— The embedding of explicitly defined elements in constructional sequences for the representation of hybrid models;

— The use of **representation_item** definitions from other parts of ISO 10303 to represent constructional operations for instances of those **representation_item**s;

— The definition of a dual representation by association of a procedural model with an explicit 'current result' model, the latter acting as a representative example of the parametric family of models defined by the former;

— The association of design rationale information with a procedural model;

— The identification, in a procedural model, of explicit elements selected by interactive picking from the visual display of the model in the sending system;

— The identification, in a procedural model, of constructional operations that can be suppressed for purposes of model simplification;

— Specialization of the foregoing capabilities for the procedural representation of shape models.

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

— Any mechanism for the 'persistent naming' of elements of an explicit model based on details of the procedural sequence used to create them;

— ‘Macro’ capabilities requiring the use of control structures such as IF… THEN… ELSE or REPEAT… UNTIL. Such structures are defined in ISO 10303-11 for use in local and global rules, but no analogous facilities are provided in this document to allow conditional operations in procedural models.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

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

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

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

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

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

ISO 10303-108, *Industrial automation systems and integration — Product data representation and exchange — Part 108: Integrated application resource: Parameterization and constraints for explicit geometric product models.*

3 Terms, definitions and abbreviations

3.1 Terms defined in ISO 10303-1

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

— application;

— application context;

— application protocol (AP);

— assembly;

— component;

— data exchange;

— exchange structure;

— implementation method;

— integrated resource (IR);

— product;

— product data;

— structure.

3.2 Terms defined in ISO 10303-11

For the purposes of this part of ISO 10303, the following terms defined in ISO 10303-11 apply.

— entity;

— entity data type;

— entity (data type) instance;

— instance;

— value.

3.3 Terms defined in ISO 10303-42

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

— boundary representation solid model (B-rep);

— constructive solid geometry (CSG);

— coordinate space;

— dimensionality;

— model space.

3.4 Terms defined in ISO 10303-43

For the purposes of this part of ISO 10303, the following terms defined in ISO 10303-43 apply.

— context of representation;

— element of representation;

— founded;

— representation.

3.5 Terms defined in ISO 10303-108

For the purposes of this part of ISO 10303, the following terms defined in ISO 10303-108 apply.

NOTE The capabilities specified in ISO 10303-108 are very closely related to those specified in this part of ISO 10303. Consequently, acquaintance with these terms and their definitions is crucial for understanding the present document.

— constraint;

— constraint solution;

— current result;

— current value;

— declarative constraint;

— declarative model;

— design intent;

— element;

— evaluated model;

— explicit constraint;

— explicit model;

— feature;

— generative model;

— history-based model;

— hybrid model;

— implicit constraint;

— model parameter;

— procedural constraint;

— procedural model;

— sketch;

— unevaluated model;

— variational.

3.6 Other terms and definitions

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

3.6.1
design rationale
logic underlying the methodology used in constructing the design

3.6.2
dual model
combination of a procedural or hybrid representation with an explicit representation, the second of which represents an example of the parametric class of models defined by the first

3.7 Abbreviations

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

AP	application protocol (of ISO 10303)
B-rep	boundary representation
CAD	computer aided design
CSG	constructive solid geometry
IR	integrated resource (of ISO 10303)

4 Procedural model

The following EXPRESS declaration begins the procedural model schema and identifies the necessary external references.

EXPRESS specification:

```
*)
SCHEMA procedural_model_schema;

REFERENCE FROM support_resource_schema                    -- ISO 10303-41
  (text);

REFERENCE FROM representation_schema                      -- ISO 10303-43
  (item_in_context,
   representation,
   representation_item,
   representation_item_relationship,
   representation_relationship,
   using_representations);

REFERENCE FROM variational_representation_schema          -- ISO 10303-108
  (variational_representation);
(*
```

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

support_resource_schema	ISO 10303-41
representation_schema	ISO 10303-43
variational_representation_schema	ISO 10303-108

NOTE 2 See annex D, Figure D.1, for a graphical presentation of this schema.

4.1 Introduction

The subject of the **procedural_representation_schema** is representation or modelling in terms of constructional operations. This may be contrasted with representation or modelling in terms of elements that are explicitly created as the result of performing those operations.

EXAMPLE ISO 10303-42 defines the entity data type **manifold_solid_brep**. This is a representation of a solid shape in terms of the faces, edges and vertices occurring in the boundary separating the interior from the exterior of the solid. Such a representation contains no information as to how the shape was actually created, though whatever constructional operations were used clearly had the effect of generating all the low-level geometrical and topological elements involved in the **manifold_solid_brep**. This part of ISO 10303 provides an alternative method of representing such a shape, in terms of the manner of its generation.

4.2 Fundamental concepts and assumptions

This schema provides representation methods for the following:

— The specification of sequences of constructional operations for the generation of models or representations of any type;

— The hierarchical structuring of constructional sequences;

— The embedding of explicitly defined elements in constructional sequences for the representation of hybrid models;

— The use of **representation_item** definitions from other parts of ISO 10303 to represent constructional operations for instances of **representation_item** in procedural and hybrid models;

— The definition of a dual representation by association of a procedural model with an explicit 'current result' model, the latter acting as a representative example of the parametric family of models defined by the former;

— The association of design rationale information with procedural models;

— The identification, in a procedural model, of explicit elements selected by interactive picking from the visual display of the model in the sending system;

— The identification, in a procedural model, of constructional operations that can be suppressed for purposes of model simplification.

The primary initial aim of this part of ISO 10303 is to provide the means for representing procedural and hybrid models of geometric shapes as generated by CAD systems. For this reason, many of the examples given in the descriptive text of this schema are concerned with aspects of CAD modelling. However, the resource constructs provided in the schema are of general utility in the representation, exchange and sharing of procedurally defined and hybrid models for any application. An example of a non-geometric application of procedural modelling is given in clause E.1 of annex E.

4.2.1 Procedural models

A *procedural model* is represented in terms of the operations used in its creation. For this reason it is also frequently known as a *construction history model*. A pure procedural model is defined exclusively in terms of operations, and it is therefore impossible to refer in such a model to most specific constituents of the explicit model that is generated when the operations are performed.

EXAMPLE A shape model of a cylindrical solid with radius R and height H may be generated from a procedural model containing just two operations:

a) Create a circular area with radius R;

b) Sweep the circular area through a distance H normal to its plane.

The cylinder resulting from the performance of these operations has two circular edges. One of them will correspond to the boundary of the circular area created by the first operation, but even this will not exist explicitly until that first operation has been carried out. The second edge will only be called into existence by the performance of the second operation. The two operations represent the shape of the cylinder, but by themselves provide no means of referencing its individual geometrical or topological elements.

In a data exchange context, a transferred procedural model will specify only operations, and the generation of an explicit model from them will occur after the transfer, in the receiving system. This process is called *evaluation*; its input is the *unevaluated* procedural model, and its output is the *evaluated* explicit model.

A procedural model is inherently parametric. The arguments of many of its constructional operations will include numerical values. Parametric variations in the model may be made simply by changing the values of these parameters. To see the effect of these changes in the corresponding explicit model it is necessary to re-evaluate on the basis of the modified construction history.

The basic element of a procedural model, as defined in this part of ISO 10303, is the entity data type **procedural_representation_sequence** (see clause 4.3.4).

4.2.2 Hybrid models

A *hybrid model* contains a combination of explicit elements and constructional operations.

EXAMPLE A common constructional procedure in CAD systems makes use of an explicitly defined planar *sketch* or *profile*. This is composed of geometrical and topological elements (curves, points, edges, vertices), possibly subject to geometric constraints such as parallelism, perpendicularity or tangency between curve elements. The constructional procedure is generically known as a *sweep operation*. The sketch is moved through space to sweep out a volume, and that volume is hence defined in terms of an operation upon an explicit element. Its representation is therefore of the hybrid type. Various classes of swept surfaces and volumes are defined in ISO 10303-42.

In the exchange of a hybrid shape model, the explicit element transferred is usually a data structure built from lower-level elements which must be transmitted with the model in order to specify it completely. These may include variational elements such as model parameters and constraints (see ISO 10303-108). Explicitly transferred elements are distinguished from procedurally defined elements simply by the fact that they do not participate in instances of **procedural_representation_sequence**.

4.2.3 Explicit selected elements

In the exchange of procedural or hybrid models, it is convenient to use explicit model elements for another important purpose, the identification of elements picked from an evaluated explicit model displayed on the screen of the sending system.

EXAMPLE Suppose that the system is a CAD system, and consider the linear sweeping (or *extrusion*) of an explicitly represented closed two-dimensional sketch composed of straight line segments enclosing an L-shaped area. The operation will generate a block volume with a step feature defined on it. This object has one concave edge, the inner edge of the step. Suppose now that it is required to generate a fillet to round off that concave edge. The following is a possible sequence of operations for generating the desired shape:

a) Sweep the sketch linearly (construction operation);

b) Select the edge to be filleted from the screen display (selection operation);

c) Fillet the edge (construction operation).

The problem here is how to represent the selection operation in the hybrid model defined by the above operation sequence. The selected edge is not present in that model; it will not arise until the model is evaluated, when it will be swept out by the motion of one of the vertices of the initial sketch. The solution is to transmit that edge explicitly from the sending system (it will have been selected from the secondary, evaluated, model in that system). Then, when the model is being regenerated in the receiving system, the explicit edge information can be used to determine which edge is to be filleted. The actions in that system will be

a) Perform the sweep operation to generate the L-block. The edge to be filleted will be created in the explicit model in the receiving system by this operation;

b) Compare the explicitly transmitted selected edge with all the edges of the newly created L-block until a match is found. The matching edge is the edge to be filleted;

c) Fillet the identified edge.

There is an important semantic distinction between the roles of an explicit model element in a hybrid model, as described in the previous clause, and an explicit selected element of the type discussed here:

explicit model element: (see clause 4.2.2). This is used for the direct transfer of an element from the sending system into the receiving system, where it did not previously exist;

explicit selected element: (as explained in the present clause). This is used for the identification, in the receiving system, of an element that has already been generated in that system, and that corresponds to a specified explicit element in the sending system.

This schema provides an entity data type **user_selected_elements** (see clause 4.3.5) that is used to distinguish explicit selected elements from other elements of a procedural or hybrid model. This has a subtype **indirectly_selected_elements** (see clause 4.3.6) for use in cases where the directly selected element is representative of some other element or elements. Examples of the use of indirect selection in a shape modelling context are given in clause 4.3.6.

4.2.4 Dual models

As explained in the Introduction, most CAD systems generate not only a procedural model but also a secondary explicit model, which plays a vital role in the interaction of the user with the modelling system. The explicit model is displayed on the screen, providing visual feedback to verify the results of modelling operations. It also allows the selection, from the screen, of modelling elements to be used as the basis for further modelling procedures.

NOTE 1 In a CAD system, the secondary explicit model is usually a manifold boundary representation solid model, as defined in ISO 10303-42 (the relevant entity data type is **manifold_solid_brep**). Earlier parts of ISO 10303 have made provision for the representation and exchange of explicit models of this and related types. This part of the standard adds the basic mechanisms needed for the representation and exchange of procedural representations of CAD and other kinds of models.

NOTE 2 In most CAD systems the secondary model is ephemeral, in the sense that it is discarded whenever a change is made to the primary model, and a new explicit model is then generated that takes the change into account.

Additionally, the explicit form of the model may be used for computational purposes.

EXAMPLE 1 If the model is a geometric model, the explicit form is necessary for the determination of such things as an edge curve defined by the intersection of two surfaces. As already explained, the primary (procedural) model does not contain the explicit elements needed to enable calculations of this type.

The importance of the interplay between primary and secondary models demands a representation for a *dual model*, a relationship between two representations, one procedural or hybrid, and the other explicit. The entity data type **explicit_procedural_representation_relationship** is defined in this part of ISO 10303 for that purpose (see clause 4.3.1). It is assumed that the two components of a dual model are always consistent alternative models or representations of the same physical entity.

There are also circumstances where it is possible, and may be desirable, to associate procedural and explicit model elements at the **representation_item** level. For this purpose the additional entity data type **explicit_procedural_representation_item_relationship** is provided (see clause 4.3.2).

This part of ISO 10303 is based on the further assumption that a procedural model exchanged between systems may be accompanied by an explicit model, in which case what is exchanged is a dual model. The primary purpose of this is to enable comparison, in the receiving system, of the explicit model reconstructed there by evaluation of the transmitted procedural model with the explicit model as it existed in the sending system. This will enable the detection of any gross errors occurring in the exchange. Also, in certain cases ambiguities can arise in the reconstruction of a hybrid model, and it may be possible to resolve these in the receiving system by reference to the explicit component of the dual model, which exhibits the choices made by the creator of the model in the sending system.

EXAMPLE 2 A representation of a CAD model may include a set of nonlinear constraints whose corresponding equations have multiple solutions. In this case the explicit model exhibits the choice of solution made by the original designer.

As explained in clause 4.2.1, a procedural or hybrid model is parametric in nature, and the explicit component of a dual model is an example from the represented parametric family. Specifically, the explicit component is the example corresponding to the current values of the parameters in the procedural model at the time of model transfer. Ideally, therefore, evaluation of the procedural model with those parameter values in the receiving system should give an explicit model that is identical with the explicit component of the dual model in the sending system. However, differences between the internal representations of modelling systems will prevent the achievement of this ideal.

EXAMPLE 3 Two hypothetical CAD systems have the following characteristics:

System A: — Internal numerical tolerance for coincidence of points is 10^{-4} units;

— A 360° circular cylindrical face is subdivided into three subfaces each subtending a $120°$ angle.

System B: — Internal numerical tolerance for coincidence of points is 10^{-7} units;

— A 360° circular cylindrical face is represented as single face, two of whose edges coincide along a *seam curve* lying along a generator of the cylinder.

Such differences ensure that the precise details of the transmitted and regenerated explicit models will rarely be in total agreement, though they will normally be close enough for practical purposes.

4.2.5 Representation of constructional operations in procedural models

An ISO 10303 model or **representation** is composed of instances of **representation_item** (see ISO 10303-43). The occurrence of such an instance in an exchange file or shared database was originally intended to be *declarative*, that is, to indicate the actual presence of the item in the model being transferred. However, ISO 10303-11 specifies (in clause 9.2.5) that 'When an entity is declared [in a schema], a constructor is also implicitly declared. The constructor identifier is the same as the entity identifier. . . The constructor, when invoked, shall return a partial complex entity value for that entity data type to the point of invocation. . . '. This capability was intended primarily for use in local or global rules in schemas, but it is used in this part of ISO 10303 for the representation of constructional operations for instances of the entity data types concerned *to be performed in the receiving system following a transfer*. Thus, whereas the 'point of invocation' of the constructor was originally envisaged to be during rule checking by an ISO 10303 translator, in the transfer of construction history models it will be during model regeneration in the receiving system. The treatment of attribute values as parameters passed to constructors is spelled out in detail in the cited clause of ISO 10303-11.

EXAMPLE Consider the following instance in an ISO 10303-21 [2] exchange file:

```
#210 = CIRCLE('C1', #150, 6.0);
```

here the attribute values represent, respectively, the name of the circle, a reference to its axis placement defined elsewhere in the file, and its radius. If this instance is transferred as an element of an explicit model, the receiving system will be expected to rewrite the definition of the circle in its native internal format and to build it into the data structure of an explicit model of the boundary representation or similar type. By contrast, if the instance is specified as part of an operation sequence in a construction history, it will invoke in the receiving system a procedure for the *ab initio* creation of the specified circle. Thus in the first case the process is conceptually one of translation, and in the second it is one of generation.

The quotation from ISO 10303-11 given above implies that procedural representations may be generated in terms of instances of *any* subtypes of **representation_item** for which EXPRESS definitions exist. A wide selection of such entity data types is therefore immediately available for use with this part of ISO 10303.

Two consequences of the use of entity data type definitions as construction operations are the following:

a) It is necessary to distinguish between instances that are used as constructors and instances that represent explicitly transferred elements forming part of a hybrid model;

b) It is also vital to capture the appropriate sequencing of instances used as constructors.

NOTE ISO 10303-21 [2] states that the sequence of instances in an ISO 10303-21 exchange file has no significance.

The entity data type **procedural_representation_sequence** has been defined in clause 4.3.4 of this part of ISO 10303 to capture the ordering of instances representing constructional operations in a procedural model. It also serves to distinguish those instances that are to be interpreted as operations from those that are not, which are excluded from such sequences.

A further aspect of the use of entity data type instances to represent operations is that local and global rules applying to such instances are interpreted as constraints on the generated elements in the receiving system, and not as validity conditions requiring to be checked as they are in the exchange of explicit models using ISO 10303.

4.2.6 Implicit and explicit constraints

A hybrid model may contain both implicit and explicit constraints. Implicit constraints are constraints in a model that arise automatically from the operation of a constructional procedure. Explicit constraints are explicitly represented relationships between elements of an evaluated model.

EXAMPLE ISO 10303-42 defines an entity data type **block**, representing a rectangular parallelepiped. If **block** is used as a constructional operation in a construction history model, the created block will have three pairs of parallel faces, because this is inherent in the definition of the shape concerned. For this case the parallelism constraints are implicit. If the dimensional parameters of the block are changed, the regenerated block will always have three pairs of parallel faces.

On the other hand, consider a pure boundary representation of a block of the same type. This will contain six faces, twelve edges and eight vertices. The parallelism of opposite faces results from the specific combination of points, lines and planes associated with the topological elements of the representation. There is nothing in such a model to prevent a modification of one or more of those geometric elements in such a way that the resulting shape loses the property of parallelism of opposite faces. This situation may be rectified by imposing explicit constraints of the type ‘Face A is parallel to Face B’, and requiring these constraints to continue to hold in any modification of the model. ISO 10303-108 provides representations for a range of such explicit constraints that can be applied to elements of a part model, and ISO 10303-109 [3] defines representations of explicit constraints between the surfaces of parts in an assembly model.

Finally, consider a hybrid model defined by a linear sweep operation on an explicitly defined planar sketch. The elements of the sketch may have explicit constraints defined upon them. The sweep operation, on the other hand, generates implicit constraints. One of these ensures that the planar faces corresponding to the initial and final positions of the swept sketch are parallel. Upon evaluation of the hybrid model it would be possible to evaluate these implicit constraints as well as the implicitly defined geometry, and to define an explicit parallelism constraint in the evaluated boundary representation model. However, such evaluation of implicit constraints does not generally occur in existing CAD systems.

4.2.7 Suppression of constructional operations

Most CAD systems provide a capability for capturing procedural models in which some constructional operations are specified as being suppressed. In such a case, the implication is that the complete design model is excessively complex for some application that will be based on it.

EXAMPLE It may be desired to suppress certain small features on a CAD model before using it as input to a finite element mesh generator. This could be the case if it were judged that these features would have no significant effect on the results of the finite element calculations, for example if they are small holes in lightly stressed regions. If these features were not suppressed their presence would lead to the generation of a more complex mesh model, resulting in longer computation times and possible loss of accuracy in the computed results.

This part of ISO 10303 allows the exchange of general procedural models with suppressed operations through the provision of an optional attribute **suppressed_items** in the entity data type **procedural_representation_sequence**. If present, this simply specifies the set of items in that sequence that were marked as suppressed in the sending system.

4.2.8 Exchange of procedural and hybrid models

The basic assumption made in this part of 10303 concerning the exchange of procedural and hybrid models is that a received model will be evaluated in the receiving system during or immediately following the input of the ISO 10303 model. As mentioned earlier, if the procedural or hybrid model is transmitted as one component of a dual model, then the accompanying explicit model may be used as the basis for comparison with the reconstructed explicit model generated by the receiving system, to ensure that the exchange has been satisfactory. If procedural or hybrid models expressed in ISO 10303 form are used for archival purposes it is anticipated that they will be similarly re-evaluated following transfer from the archive into an appropriate application system.

4.2.9 Variational cases of procedural and hybrid models

The entity data type **variational_representation** is defined in ISO 10303-108 as a type of **representation** that contains explicitly modelled parameters and constraints. Such a representation may be thought of as the association of an explicit 'current result' model with those parameters and constraints, which enable it to be edited, following a transfer, in conformance with the designer's original intent. By contrast, the parameters in a procedural model may be thought of as input arguments to constructional operations, and any constraints are implicit (unless they occur in explicit hybrid elements). However, it is also possible to associate explicit parameters and constraints, as defined in ISO 10303-108, with elements in an exchange file that provide supporting information for the definition of operations in the operation sequences of a procedural or hybrid model. An example is given in annex E.3. In such a case the resulting procedural representation is required (by a WHERE rule of **variational_representation_item** as defined in ISO 10303-108) to have the additional type **variational_representation**. The current result may then have both procedural and explicit aspects, the first being the procedural representation evaluated with the current values of all its input arguments, and the second (in the case when a dual model is being exchanged) being the associated explicit model. The two are in principle equivalent in the sense that the second is the result of evaluating the first. The intended usage is that the variational representation shall refer to its associated explicit current result via the procedural form, as illustrated in annex E.3.

4.3 Procedural model entity definitions

4.3.1 explicit_procedural_representation_relationship

The **explicit_procedural_representation_relationship** entity data type is a type of **representation_relationship** as defined in ISO 10303-43. It specifies an association between a procedural or hybrid representation and a corresponding explicit representation (known as the 'current result'). A procedural representation is essentially parametric, representing a family of models. The explicit current result model is therefore regarded as a representative example of a model from that family, indicative of some aspects of its creator's intentions in constructing it. A WHERE rule is defined to ensure that the associated explicit representation is of some fully explicit type, belonging to neither of the types **procedural_representation** or **variational_representation**.

EXPRESS specification:

```
*)
ENTITY explicit_procedural_representation_relationship
  SUBTYPE OF (representation_relationship);
  SELF\representation_relationship.rep_1 : procedural_representation;
WHERE
  WR1 : (NOT ('PROCEDURAL_MODEL_SCHEMA.PROCEDURAL_REPRESENTATION'
    IN TYPEOF(SELF\representation_relationship.rep_2))) AND
    (NOT ('VARIATIONAL_REPRESENTATION_SCHEMA.VARIATIONAL_REPRESENTATION'
    IN TYPEOF(SELF\representation_relationship.rep_2)));
  WR2 : SELF\representation_relationship.rep_1.context_of_items :=:
    SELF\representation_relationship.rep_2.context_of_items;
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation_relationship.rep_1: the procedural representation defining the parameterized family of models.

SELF\representation_relationship.rep_2: a representative explicit (non-parametric) model from the family of models defined by the related procedural (parametric) model.

Formal propositions:

WR1: The representation referenced by the attribute **SELF\representation_relationship.rep_2** shall not be an instance of **procedural_representation** or of **variational_representation**.

WR2: The two representations related by an instance of **explicit_procedural_representation_relationship** shall share a common representation context.

Informal propositions:

IP1: The explicit model shall be the result of evaluating the procedural model with the current values of all its parameters.

4.3.2 explicit_procedural_representation_item_relationship

The **explicit_procedural_representation_item_relationship** entity data type is a type of **representation_item_relationship** as defined in ISO 10303-43. It defines an association between two instances of **representation_item**, one procedurally defined and the other explicitly defined, both representing the same modelling element.

NOTE In general, **representation_item** instances occurring in a procedural model do not have counterpart instances in the current result. However, in cases where this does happen it may be useful to establish relationships at the **representation_item** level in addition to those defined at the **representation** level.

EXAMPLE Consider the case of a nut and a bolt as modelled in a CAD system. ISO 10303 does not enforce the modelling of these two items as separate products, and the system user may have created them as discrete elements of a single **shape_representation**. Each element will be a **representation_item**. In this case the explicit shape representation will contain two instances of **representation_item** of subtype (for example) **manifold_solid_brep**, while the procedural shape representation will contain two instances of **representation_item** of type **procedural_shape_representation_sequence**. Then the association between the explicit and procedural models of each element can be captured through the use of **explicit_procedural_representation_item_relationship** instances, though in this case the subtype **explicit_procedural_geometric_representation_item_relationship**, defined in clause 5.4.2 will be appropriate.

EXPRESS specification:

```
*)
ENTITY explicit_procedural_representation_item_relationship
  SUBTYPE OF (representation_item_relationship);
  SELF\representation_item_relationship.relating_representation_item :
    procedural_representation_sequence;
WHERE
  WR1 : NOT ('PROCEDURAL_MODEL_SCHEMA.PROCEDURAL_REPRESENTATION_SEQUENCE'
    IN TYPEOF(
      SELF\representation_item_relationship.related_representation_item));
  WR2 : SIZEOF(QUERY(q <* using_representations(
    SELF\representation_item_relationship.related_representation_item) |
    item_in_context(
      SELF\representation_item_relationship.relating_representation_item,
      q.context_of_items))) > 0;
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation_item_relationship.relating_representation_item: an instance of **procedural_representation_sequence** defining the procedural representation of a model element.

SELF\representation_item_relationship.related_representation_item: an explicitly defined model element corresponding to the result of evaluating the procedural definition.

Formal propositions:

WR1: The instance of **representation_item** referenced by the attribute **SELF\representation_item_-relationship.related_representation** shall not be an instance of **procedural_representation_sequence**.

WR2: The two related instances of **representation_item** shall share at least one common **representation_context**.

Informal propositions:

IP1: The explicit model element shall be the result of evaluating the procedurally defined model element with the current values of all its parameters.

4.3.3 procedural_representation

The **procedural_representation** entity data type is a type of **representation** as defined in ISO 10303-41. It defines a model in terms of a set of **procedural_representation_sequence** instances. Each such sequence represents a full or partial model. The definition of a procedural representation in terms of a set rather than a list of instances of **procedural_representation_sequence** allows for cases where the representation sequences are independent of each other.

EXAMPLE An assembly may include components that were originally designed independently of each other for some different purpose. In such a case there is no necessity for ordering the design sequences for those components in any particular way, and the use of a set of sequences is appropriate.

On the other hand, the case where the design of one component is dependent on the prior existence of the design for another component can be handled by embedding their two design sequences in the appropriate order in a higher-level design sequence, as illustrated in clauses E.2 and E.4 of annex E.

EXPRESS specification:

```
*)
ENTITY procedural_representation
  SUBTYPE OF (representation);
  SELF\representation.items :
    SET[1:?] OF procedural_representation_sequence;
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation.items: the set of **representation_item** instances of the **representation** supertype, in this case a set of **procedural_representation_sequence** instances.

NOTE A WHERE rule of the entity data type **procedural_representation_sequence** (see clause 4.3.4) ensures that all instances of **representation_item** involved in an instance of such a sequence shall share the **representation_context** associated with its owning **procedural_representation**. In complex cases where operation sequences are embedded within each other in an instance of **procedural_representation** (for example, see instance `#1290` in clause E.2 of annex E), this rule ensures compatibility of **representation_context** at all levels of embedding.

4.3.4 procedural_representation_sequence

The **procedural_representation_sequence** entity data type is a type of **representation_item** as defined in ISO 10303-43. It represents a list of constructional operations for the creation of a partial or complete model. The operations shall be performed, following a model transfer, in the order in which they appear in the list. The fact that this entity data type is defined as a subtype of **representation_item** allows its instances to be included (or *embedded*) in the operation sequences of higher-level instances of the same type. This allows the sequencing of partial designs in an overall design when later design stages are dependent on earlier ones, as illustrated in clauses E.2 and E.4 of annex E.

NOTE 1 The correct ordering of constructional operations is crucial if the transmitted model is to be correctly reconstructed in a receiving system. In general, the same set of operations performed in a different order will generate a different model.

This entity data type optionally records a set of items belonging to the sequence that are to be treated as suppressed following the transfer of a model.

NOTE 2 Such suppression is usually applied in order to simplify a model, for example by elimination of fine detail, for some application purpose downstream of design.

The intended usage of the **suppressed_items** attribute is as follows:

— Following receipt of the transferred procedural model in the receiving system, the complete model shall be evaluated, with no operations suppressed.

— Once this has been done, a further explicit model shall be generated, using the same underlying instances of **procedural_representation_sequence** but with omission of the operations specified by the attribute **suppressed_items**.

— The current result transmitted with the procedural model shall represent the full model with no operations suppressed. Comparison of the regenerated explicit model with the current result shall therefore take place after the first, full, evaluation. Suppression of operations should therefore not be used in the sending system for the correction of errors, but only for simplification purposes.

EXPRESS specification:

```
*)
ENTITY procedural_representation_sequence
  SUBTYPE OF (representation_item);
  elements         : LIST[1:?] OF representation_item;
  suppressed_items : SET[0:?] OF representation_item;
  rationale        : text;
WHERE
  WR1: SIZEOF(QUERY(q <* suppressed_items | NOT (q IN elements))) = 0;
END_ENTITY;
(*
```

Attribute definitions:

elements: the list of **representation_item** instances to be created in the receiving system.

suppressed_items: the set of **representation_item** instances that, regarded as operations, are to be omitted in the generation of the simplified explicit representation in the receiving system.

rationale: a textual description of the design rationale underlying the represented sequence of operations.

Formal propositions:

WR1: No item shall occur in the set of **suppressed_items** that is not present in the sequence of operations.

NOTE 3 No restriction is imposed in this schema on the range of subtypes of **representation_item** that may participate in an instance of **procedural_representation_sequence**. It is therefore possible, in principle, to specify operation sequences comprising elements of widely different types. However, for compatibility with CAD system capabilities, several subtypes of this entity data type are provided in clause 5 for the represetation of specific types of CAD model (see clauses 5.4.5, 5.4.6 and 5.4.7).

4.3.5 user_selected_elements

The **user_selected_elements** entity data type is a type of **representation_item**, representing a reference to one or more explicit elements that were picked from the screen of the sending system.

EXAMPLE In a CAD context, the set of selected elements may be a connected collection of edges and vertices of a solid model that are to be rounded with a specified radius.

The occurrence of an instance of this entity in a **procedural_representation_sequence** is intended to initiate a search in the receiving system for the model element(s) matching the specified selected element(s). Any matching elements correspond to elements that were selected by the user of the sending system, as explained in clause 4.2.3. In appropriate uses of this entity data type, a **representation_item** in the sending system that is referenced by the **user_selected_elements** should always have a matching counterpart in the explicit model under construction in the receiving system.

EXPRESS specification:

```
*)
ENTITY user_selected_elements
  SUBTYPE OF (representation_item);
  picked_items : SET[1:?] OF representation_item;
END_ENTITY;
(*
```

Attribute definitions:

picked_items: a set of **representation_item** instances that are to be matched with corresponding elements in the explicit model regenerated from the transferred construction history model.

Informal propositions:

IP1: Any instance in the set of selected elements shall correspond to a matching element of the explicit representation resulting from evaluation of the transferred procedural model up to the point immediately before the occurrence of **user_selected_elements** containing that instance.

NOTE 1 An element referenced by an instance of this entity data type will not in general be present in the current result model, as in the case of a selected edge of a shape model which, when filleted, is replaced by the resulting fillet face. Such superseded elements must therefore be 'remembered' by the sending system and available for transmission in the transfer of procedural models.

NOTE 2 The intended usage of any instance of this entity data type, or its subtypes, is that the instance shall appear in an operation sequence immediately before the constructional operation in which the selected elements are used. The instance of **user_selected_elements** will then mark the occurrence of the selection operation at the appropriate point in the sequence. This recommended usage is illustrated in clause E.2 of annex E.

4.3.6 indirectly_selected_elements

The entity data type **indirectly_selected_elements** is a type of **user_selected_elements** indicating that the explicitly selected element(s) are intended to be representative of some other related element or elements. It therefore provides an additional attribute, used to specify a set of **representation_item** instances that are referenced indirectly.

EXAMPLE 1 In a CAD context, two edges may be selected to identify a specific face of a solid model (a single edge does not suffice, because it is shared by two faces).

EXAMPLE 2 A face may be selected to represent an entire solid model that is to be repositioned in an assembly.

EXAMPLE 3 A solid model may be selected as an indirect reference to the set of all its edges and vertices if these are to be rounded with a specified radius.

When this entity data type is instanced it is intended that the *indirectly* referenced elements as well as those directly referenced shall be matched with elements in the explicit model under construction in the receiving system. Although the indirectly referenced elements are the ones that will subsequently be used in the constructional procedure, the transfer of the directly referenced elements constitutes part of the exchange of design intent.

EXPRESS specification:

```
*)
ENTITY indirectly_selected_elements
  SUBTYPE OF (user_selected_elements);
  indirectly_picked_items : SET[1:?] OF representation_item;
END_ENTITY;
(*
```

Attribute definitions:

indirectly_picked_items: a set of **representation_item** instances that are to be matched with corresponding elements in the explicit model regenerated from the transferred construction history model.

Informal propositions:

IP1: Any instance in the set of indirectly selected elements shall correspond to a matching element of the explicit representation resulting from evaluation of the transferred procedural model up to the point immediately before the occurrence of **indirectly_selected_elements** containing that instance.

NOTE An indirectly selected element referenced by an instance of this entity data type will not in general be present in the current result model, because it may have been modified or deleted as a result of subsequent modelling operations. Such superseded elements must therefore be ‘remembered’ by the sending system and available for transmission in the transfer of procedural models.

EXPRESS specification:

```
*)
END_SCHEMA; -- procedural_model_schema
(*
```

5 Procedural shape model

The following EXPRESS declaration begins the procedural shape model schema and identifies the necessary external references.

EXPRESS specification:

```
*)
SCHEMA procedural_shape_model_schema;

REFERENCE FROM product_property_representation_schema   -- ISO 10303-41
  (shape_representation);

REFERENCE FROM geometry_schema                          -- ISO 10303-42
  (geometric_representation_item);

REFERENCE FROM topology_schema                          -- ISO 10303-42
  (topological_representation_item);

REFERENCE FROM geometric_model_schema                   -- ISO 10303-42
  (edge_based_wireframe_model,
   face_based_surface_model,
   shell_based_surface_model,
   shell_based_wireframe_model,
   solid_model);

REFERENCE FROM representation_schema                    -- ISO 10303-43
  (representation,
   representation_item_relationship,
   representation_relationship);

REFERENCE FROM procedural_model_schema;                 -- ISO 10303-55
(*
```

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

`product_property_representation_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
`procedural_model_schema`	clause 4 of this part of ISO 10303

NOTE 2 See annex D, Figures D.2 and D.3, for a graphical presentation of this schema.

5.1 Introduction

The subject of the **procedural_shape_representation_schema** is the representation or modelling of shape in terms of constructional operations.

5.2 Fundamental concepts and assumptions

This schema provides representation methods for the following:

— The specification of shape representations in terms of sequences of constructional operations;

— The use of shape modelling entity data types from other parts of ISO 10303 to represent constructional operations for instances of those entity data types.

These resource constructs are of general utility in the representation, exchange and sharing of procedurally defined and hybrid shape models. The entity data types defined in the **procedural_shape_model_schema** are specializations of corresponding entity data types in the **procedural_model_schema** (see clause 4), and the introductory descriptive clauses of that schema apply also in the context of the present schema.

5.2.1 Procedural shape models

This schema introduces the entity data type **procedural_shape_representation**, a subtype of **procedural_representation** that represents a shape (see clauses 5.4.3 and 4.3.3 respectively). In this case the constructional operations defining the model may generate instances of the following entity data types defined in ISO 10303-42:

a) **geometric_representation_item**, which includes low-level entities such as points, curves, surfaces and associations of such geometry with topological elements. It also covers certain types of model built from such low-level elements, in particular solid, surface and wireframe models;

b) **topological_representation_item**, which includes pure topological entities such as vertices, edges and faces, together with their combinations that form loops, shells etc.

CSG models, which are specific cases of solid models, are intrinsically procedural, being defined in terms of Boolean operations on volumetric elements. ISO 10303-42 also specifies other procedurally defined shape elements, including certain types of swept surfaces and volumes.

NOTE 1 Further high-level constructional operations, in general absent from ISO 10303-42, are used by CAD systems to generate entire features of a shape model, whose constituent elements would otherwise need to be generated through the use of multiple low-level operations. Representations for CAD design features will be provided in a future part of ISO 10303.

NOTE 2 The generation of parameterized part and assembly models from procedural representations will require the use of ISO 10303-108 (‘Parameterization and constraints for explicit geometric product models’) and ISO 10303-109 (‘Kinematic and geometric constraints for assembly models’) [3]. These define a wide range of **geometric_representation_item** subtypes applicable to parametric models.

NOTE 3 Relationships between assembly constituents, as defined in ISO 10303-109 [3] and its associated application modules, are specified as relationships between geometric elements of those constituents at the level of **geometric_representation_item**. This allows such relationships to be captured in a natural way in a procedural assembly model. The appropriate method in most cases will be to capture the geometric elements concerned as selected elements (since they will not normally be present explicitly in a procedural model) in order to create the relationship between them. Relationships between assembly consitituents as defined in ISO 10303-44 are specified at a higher level, and their representation is independent of the manner of representation of the individual constituents.

The parameters involved in the constructional operations of a **procedural_shape_model** will include, for example, numerical values specifying dimensions, orientations or locations of elements to be created in the model. In most CAD systems, the primary model representation is procedural but the secondary model actually displayed on the screen is the explicit ‘current result’. The explicit model is usually discarded (wholly or partially) when the primary model is edited, and a new version generated from the modified constructional sequence.

The basic element of a procedural shape model is the entity data type **procedural_shape_representation_sequence** (see clause 5.4.4). Specialized subtypes of **procedural_shape_representation_sequence** are provided for the generation of solid, surface and wireframe models.

5.2.2 Hybrid shape models

A *hybrid shape model* contains a combination of explicit geometric, topological or geometric model elements, together with a set of constructional operations appropriate to shape models. An example was given in clause 4.2.2.

5.2.3 Explicit selected elements in a shape model

The use of explicit selected elements is very important in the exchange of procedural models, as explained in clause 4.2.3, where an example application was given. The screen display of a shape modelling system is usually generated from an explicit version of the model, and the user's selected elements will typically be vertices, edges or faces of that model.

This schema provides an entity data type **user_selected_shape_elements**, a subtype of **user_selected_elements** as defined in clause 4.3.5, that is used for the capture and transfer of selected instances of shape model elements. This has a subtype **indirectly_selected_shape_elements**, defined in clause 5.4.9, for use in cases where (for example) a directly selected shape element is representative of some other shape element or elements.

EXAMPLE An instance of **indirectly_selected_shape_elements** may be used when an edge of a solid model is selected as being representative of the entire solid model it belongs to. In this case the edge is directly selected and the solid model is indirectly selected.

5.2.4 Dual shape representations

In the case of a dual shape representation, the operations defined in the primary procedural model all create instances of **geometric_representation_item** or **topological_representation_item**. The secondary explicit model will be one of the explicit forms of shape representation. The present schema provides an entity data type **explicit_procedural_shape_representation_relationship**, a subtype of **explicit_procedural_representation_relationship** as defined in clause 4, to capture the appropriate association between the procedural and explicit components of a dual shape representation.

There are also circumstances where it is possible, and may be desirable, to associate procedural and explicit shape model elements at the **geometric_representation_item** level. For this purpose the additional entity data type **explicit_procedural_geometric_representation_item_relationship** is provided.

5.2.5 Design rationale for shape models

At the level of an individual feature of a shape model, design rationale may be determined by simply querying the feature on the CAD system screen, in which case its current parameter values are displayed and may be modified interactively. The interaction in this case is with the explicit component of the dual model. However, at deeper levels of modelling the rationale underlying major segments of the constructional history may be very obscure and impossible to elucidate by querying the explicit model. In such cases it may be necessary to try to interpret the procedural form of the shape model directly. Complex procedural representations of CAD models are notoriously difficult for human operatives to comprehend, and this part of ISO 10303 therefore makes provision for the hierarchical structuring of construction histories and for the inclusion of textual material explaining the design rationale behind each structural component of the model. This material plays an analogous role in a procedural model to comment in a computer program. At present there is no known method for the automatic capture of design rationale information, but it is hoped that such developments may occur in the future, possibly as an outcome of the increasing use of knowledge-based methods in engineering design.

5.3 Procedural shape model type definitions

5.3.1 shape_representation_item

The **shape_representation_item** type allows a selection between two subtypes of **representation_item** that may be used to define constructional operations in a **procedural_shape_representation**. The underlying entity data type **representation_item** is defined in ISO 10303-43

EXPRESS specification:

```
*)
TYPE shape_representation_item = SELECT
  (geometric_representation_item,
   topological_representation_item);
END_TYPE;
(*
```

5.4 Procedural shape model entity definitions

5.4.1 explicit_procedural_shape_representation_relationship

The **explicit_procedural_shape_representation_relationship** entity data type is a type of **explicit_procedural_representation_relationship** as defined in clause 4.3.1. It associates a procedural or hybrid shape representation with a corresponding explicit current result. The procedural representation is in general parametric, representing a family of shapes. The explicit current result is regarded as a representative example of a shape from that family, indicative of some aspects of its creator's intentions in constructing it.

EXPRESS specification:

```
*)
ENTITY explicit_procedural_shape_representation_relationship
  SUBTYPE OF (explicit_procedural_representation_relationship);
  SELF\representation_relationship.rep_1 : procedural_shape_representation;
  SELF\representation_relationship.rep_2 : shape_representation;
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation_relationship.rep_1: the procedural representation defining the parameterized family of shapes.

SELF\representation_relationship.rep_2: a representative explicit (non-parametric) model from the family of shape models defined by the related procedural (parametric) representation.

NOTE A WHERE rule on the supertype **explicit_procedural_representation_relationship** ensures that the instance referenced by the attribute **SELF\representation_relationship.rep_2** is not of type **procedural_shape_representation**.

5.4.2 explicit_procedural_geometric_representation_item_relationship

The **explicit_procedural_geometric_representation_item_relationship** entity data type is a type of **explicit_procedural_representation_item_relationship** as defined in clause 4.3.2. It associates two instances of **geometric_representation_item**, one procedurally defined and the other explicitly defined, both representing the same shape modelling element.

NOTE 1 In general, **geometric_representation_item** instances occurring in a procedural shape model do not have counterpart instances in the current result. However, in cases where this does happen it may be useful to establish relationships at the **representation_item** level in addition to those defined at the **representation** level.

NOTE 2 An example of the use of this entity data type was given in clause 4.3.2.

EXPRESS specification:

```
*)
ENTITY explicit_procedural_geometric_representation_item_relationship
  SUBTYPE OF (explicit_procedural_representation_item_relationship);
  SELF\representation_item_relationship.relating_representation_item :
    procedural_shape_representation_sequence;
  SELF\representation_item_relationship.related_representation_item :
    geometric_representation_item;
WHERE
  WR1 : NOT (
    'PROCEDURAL_SHAPE_MODEL_SCHEMA.PROCEDURAL_SHAPE_REPRESENTATION_SEQUENCE'
    IN TYPEOF(
      SELF\representation_item_relationship.related_representation_item));
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation_item_relationship.relating_representation_item: an instance of **procedural_shape_representation_sequence** defining the procedural representation of a model element.

SELF\representation_item_relationship.related_representation_item: an explicitly defined shape model element corresponding to the result of evaluating the procedural definition.

Formal propositions:

WR1: The instance of **representation_item** referenced by the attribute **SELF\representation_item_relationship.related_representation** shall not be of type **procedural_shape_representation_sequence**.

Informal propositions:

IP1: The explicit model element shall be the result of evaluating the procedurally defined model element with the current values of all its parameters.

5.4.3 procedural_shape_representation

The **procedural_shape_representation** entity data type is a type of **shape_representation** (as defined in ISO 10303-41) that defines a shape in terms of a set of **procedural_shape_representation_sequence** instances. It is also a type of **procedural_representation** as defined in clause 4.3.3. Each sequence specified by the attribute **SELF\representation.items** defines a full or partial shape, and partial shapes may be combined into more complex shapes by themselves serving as elements in further sequences.

EXPRESS specification:

```
*)
ENTITY procedural_shape_representation
  SUBTYPE OF (procedural_representation, shape_representation);
  SELF\representation.items :
    SET[1:?] OF procedural_shape_representation_sequence;
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation.items: the set of representation items of the **representation** supertype, in this case a set of **procedural_shape_representation_sequence** instances.

NOTE 1 A WHERE rule of **procedural_representation_sequence** (see clause 4.3.4) ensures that all instances of **representation_item** involved in a specific instance of **procedural_representation** are associated in a common **representation_context**. A **procedural_shape_representation** is composed of instances of **procedural_shape_representation_sequence**, each of which in turn contains a list of **shape_representation_item** instances. These are either **geometric_representation_item** instances or **topological_representation_item** instances. Unless the representation is purely topological it will contain one or more instances of **geometric_representation_item**, which requires that the associated **representation_context** shall be a **geometric_representation_context** as defined in ISO 10303-42. This subtype of **representation_context**, applicable to shape representations, has an attribute **dimension_count** defining the dimensionality of the shape representation owning the context. Thus all instances of **geometric_representation_item** occurring in an instance of **procedural_shape_representation** are required to have compatible dimensionality.

NOTE 2 No requirement is imposed in this part of ISO 10303 for all members of the set of representation items of an instance of **procedural_shape_representation** to have full type compatibility. It is therefore possible, in principle, to specify a shape model comprising elements of different types, for example a solid model with associated wireframe extensions. Other parts of ISO 10303 using or specializing this resource may impose additional restrictions in that respect.

5.4.4 procedural_shape_representation_sequence

The **procedural_shape_representation_sequence** entity data type is a type of **geometric_representation_item** that represents a list of constructional operations for the creation of a shape model. It is also a type of **procedural_representation_sequence** as defined in clause 4.3.4. The operations shall be performed, following a model transfer, in the order in which they appear in the list.

NOTE The correct ordering of constructional operations is crucial if the transmitted shape model is to be correctly reconstructed in a receiving system. In general, the same set of operations performed in a different order will generate a different shape model.

EXPRESS specification:

```
*)
ENTITY procedural_shape_representation_sequence
  SUBTYPE OF (geometric_representation_item,
              procedural_representation_sequence);
WHERE
  WR1 : SIZEOF(QUERY(q <* SELF\procedural_representation_sequence.elements
    | NOT ('PROCEDURAL_SHAPE_MODEL_SCHEMA.SHAPE_REPRESENTATION_ITEM'
    IN TYPEOF(q)))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: Each model element to be created in the receiving system shall have one of the types belonging to the SELECT type **shape_representation_item**.

5.4.5 procedural_solid_representation_sequence

The **procedural_solid_representation_sequence** entity data type is a type of **procedural_shape_representation_sequence** that represents a solid shape. Its TYPE list is required to include **solid_model** as defined in ISO 10303-42.

It is not necessary that the result of each individual operation in the sequence is a solid model, only that the result of the final operation can be interpreted as a solid model. The same principle applies to the further subtypes of **procedural_shape_representation_sequence** defined in clauses 5.4.6 and 5.4.7. The following examples illustrate two of the possibilities that can arise in the procedural representation of a solid.

EXAMPLE 1 The occurrence of a selection operation in the sequence neither creates a new solid nor changes an existing solid model. Rather, it is an indication that an action occurred in the sending system that was preparatory to an ensuing operation giving rise to a change in an existing solid model.

EXAMPLE 2 The use of Boolean operations in an instance of **procedural_solid_representation_sequence** will lead to the occurrence of **boolean_result** instances in the sequence. The entity data type **boolean_result** is defined in ISO 10303-42, and it is *not* a subtype of **solid_model**. It represents a collection of zero, one or more solids that are regarded as in some sense intermediate results in a construction process. ISO 10303-42 provides an entity data type **csg_solid** which may be used to characterize the final result in a Boolean construction process as an instance of **solid_model**. However, its use is not necessary in the circumstances described at the beginning of this example, because the sequence itself represents a final result and is itself defined as a subtype of **solid_model**.

EXPRESS specification:

```
*)
ENTITY procedural_solid_representation_sequence
  SUBTYPE OF (procedural_shape_representation_sequence);
WHERE
  WR1 : 'GEOMETRIC_MODEL_SCHEMA.SOLID_MODEL' IN TYPEOF(SELF);
END_ENTITY;
(*
```

Formal propositions:

WR1: The type list of an instance of **procedural_solid_representation_sequence** shall include **solid_-model**.

Informal propositions:

IP1: The operations listed in an instance of **procedural_solid_representation_sequence** shall be concerned with the creation or modification of a solid model, and the result of the final operation in the sequence shall be interpretable as an instance of **solid_model**.

5.4.6 procedural_surface_representation_sequence

The **procedural_surface_representation_sequence** entity data type is a type of **procedural_shape_-representation_sequence** that represents a surface model. Its TYPE list is required to include one of the SELECT types of **surface_model** as defined in ISO 10303-42, namely **shell_based_surface_model** or **face_based_surface_model**.

EXPRESS specification:

```
*)
ENTITY procedural_surface_representation_sequence
  SUBTYPE OF (procedural_shape_representation_sequence);
WHERE
  WR1 : ('GEOMETRIC_MODEL_SCHEMA.FACE_BASED_SURFACE_MODEL' IN TYPEOF(SELF))
    XOR
    ('GEOMETRIC_MODEL_SCHEMA.SHELL_BASED_SURFACE_MODEL' IN TYPEOF(SELF));
END_ENTITY;
(*
```

Formal propositions:

WR1: The type list of an instance of **procedural_surface_representation_sequence** shall include either **face_based_surface_model** or **shell_based_surface_model**.

Informal propositions:

IP1: The operations listed in an instance of **procedural_surface_representation_sequence** shall be concerned with the creation or modification of a surface model, and the result of the final operation in the sequence shall be consistent with the type of surface model specified in the type list of the parent instance.

5.4.7 procedural_wireframe_representation_sequence

The **procedural_wireframe_representation_sequence** entity data type is a type of **procedural_shape_representation_sequence** that represents a wireframe model. Its TYPE list is required to include one of the SELECT types of **wireframe_model** as defined in ISO 10303-42, namely **shell_based_wireframe_model** or **edge_based_wireframe_model**.

EXPRESS specification:

```
*)
ENTITY procedural_wireframe_representation_sequence
  SUBTYPE OF (procedural_shape_representation_sequence);
WHERE
WR1 : ('GEOMETRIC_MODEL_SCHEMA.EDGE_BASED_WIREFRAME_MODEL' IN TYPEOF(SELF))
    XOR
    ('GEOMETRIC_MODEL_SCHEMA.SHELL_BASED_WIREFRAME_MODEL' IN TYPEOF(SELF));
END_ENTITY;
(*
```

Formal propositions:

WR1: The type list of an instance of **procedural_wireframe_representation_sequence** shall include either **edge_based_wireframe_model** or **shell_based_wireframe_model**.

Informal propositions:

IP1: The operations listed in an instance of **procedural_wireframe_representation_sequence** shall be concerned with the creation or modification of a wireframe model, and the result of the final operation in the sequence shall be consistent with the type of wireframe model specified in the type list of the parent instance.

5.4.8 user_selected_shape_elements

The **user_selected_shape_elements** entity data type is a type of **user_selected_elements** as defined in clause 4.3.5. It provides a reference to one or more explicit shape modelling elements that were picked from the screen of the sending system. The occurrence of an instance of this entity in a **procedural_shape_representation_sequence** is intended to initiate a search in the receiving system for shape elements that match those referenced by the **user_selected_shape_elements** instance. The matching elements correspond to the elements that were selected by the user of the sending system, as explained in clause 4.2.3. The shape elements referenced by the **user_selected_shape_elements** should exist in the explicit shape model as reconstructed in the receiving system at the time the selected element is referenced.

EXPRESS specification:

```
*)
ENTITY user_selected_shape_elements
  SUBTYPE OF (user_selected_elements);
WHERE
  WR1 : SIZEOF(QUERY(q <*
    SELF\user_selected_elements.picked_items | NOT
    ('PROCEDURAL_SHAPE_MODEL_SCHEMA.SHAPE_REPRESENTATION_ITEM'
    IN TYPEOF(q)))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: The type list of each selected element shall include **shape_representation_item**.

5.4.9 indirectly_selected_shape_elements

The **indirectly_selected_shape_elements** entity data type is a type of both **user_selected_shape_elements**, as defined in clause 5.4.8, and **indirectly_selected_elements**, as defined in clause 4.3.6. The use of this subtype is appropriate when the explicitly selected shape element(s) are intended to be representative of some other related shape element or elements. It therefore defines an additional attribute, used to specify a set of **shape_representation_item** instances that are referenced indirectly. Examples were given in clause 4.3.6.

When this entity data type is instanced it is intended that the *indirectly* referenced shape elements as well as those directly referenced shall be matched with elements of the explicit shape model under construction in the receiving system. Although the indirectly referenced elements are the ones that will subsequently be used in the constructional procedure, the transfer of the directly referenced elements constitutes part of the exchange of design intent.

EXPRESS specification:

```
*)
ENTITY indirectly_selected_shape_elements
  SUBTYPE OF (indirectly_selected_elements,
              user_selected_shape_elements);
WHERE
  WR1 : SIZEOF(QUERY(q <*
    SELF\indirectly_selected_elements.indirectly_picked_items
    | NOT ('PROCEDURAL_SHAPE_MODEL_SCHEMA.SHAPE_REPRESENTATION_ITEM'
    IN TYPEOF(q)))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: The type list of each indirectly selected element shall include **shape_representation_item**.

EXPRESS specification:

```
*)
END_SCHEMA; -- procedural_shape_model_schema
(*
```

Annex A
(normative)

Short names of entities

Table A.1 provides the short names of entities specified in this part of ISO 10303. Requirements on the use of short names are found in the implementation methods included in ISO 10303.

Table A.1 – Short names of entities

Entity data type names	Short names
explicit_procedural_geometric_representation_item_relationship	EPGRIR
explicit_procedural_representation_item_relationship	EPRIR
explicit_procedural_representation_relationship	EPRR
explicit_procedural_shape_representation_relationship	EPSRR
indirectly_selected_elements	INSLEL
indirectly_selected_shape_elements	ISSE
procedural_representation	PRC0
procedural_representation_sequence	PRRPSQ
procedural_shape_representation	PRSHRP
procedural_shape_representation_sequence	PSRS
procedural_solid_representation_sequence	PSR0
procedural_surface_representation_sequence	PSR1
procedural_wireframe_representation_sequence	PWRS
user_selected_elements	USSLEL
user_selected_shape_elements	USSE

Annex B
(normative)

Information object registration

B.1 Document identification

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

```
{ iso standard 10303 part(55) version(1) }
```

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

B.2 Schema identification

B.2.1 procedural_model_schema identification

To provide for unambiguous identification of the procedural-model-schema in an open information system, the object identifier

```
{ iso standard 10303 part(55) version(1) schema(1)
            procedural-model-schema(1) }
```

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

B.2.2 procedural_shape_model_schema identification

To provide for unambiguous identification of the procedural-shape-model-schema in an open information system, the object identifier

```
{ iso standard 10303 part(55) version(1) schema(1)
         procedural-shape-model-schema(2) }
```

is assigned to the **procedural_shape_model_schema** (see clause 5). The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

Annex C
(informative)

Computer interpretable listings

This annex references a listing of the EXPRESS entity data type names and corresponding short names as specified in this part of ISO 10303. It also references a listing of each EXPRESS schema specified in this part of ISO 10303, without comments or other explanatory text. These listings are available in computer-interpretable form, and can be found at the following URLs:

Short names:

`http://www.tc184-sc4.org/Short_Names/`

EXPRESS:

`http://www.tc184-sc4.org/EXPRESS/`

If there is difficulty accessing these sites contact ISO Central Secretariat or contact the ISO TC184/SC4 Secretariat directly at `sc4sec@tc184-sc4.org`.

NOTE The information provided in computer-interpretable form at the above URLs is informative. The information that is contained in the body of this part of ISO 10303 is normative.

Annex D
(informative)

EXPRESS-G diagrams

The diagrams in this annex correspond to the EXPRESS schemas specified in this part of ISO 10303. The diagrams use the EXPRESS-G graphical notation for the EXPRESS language. EXPRESS-G is defined in annex D of ISO 10303-11.

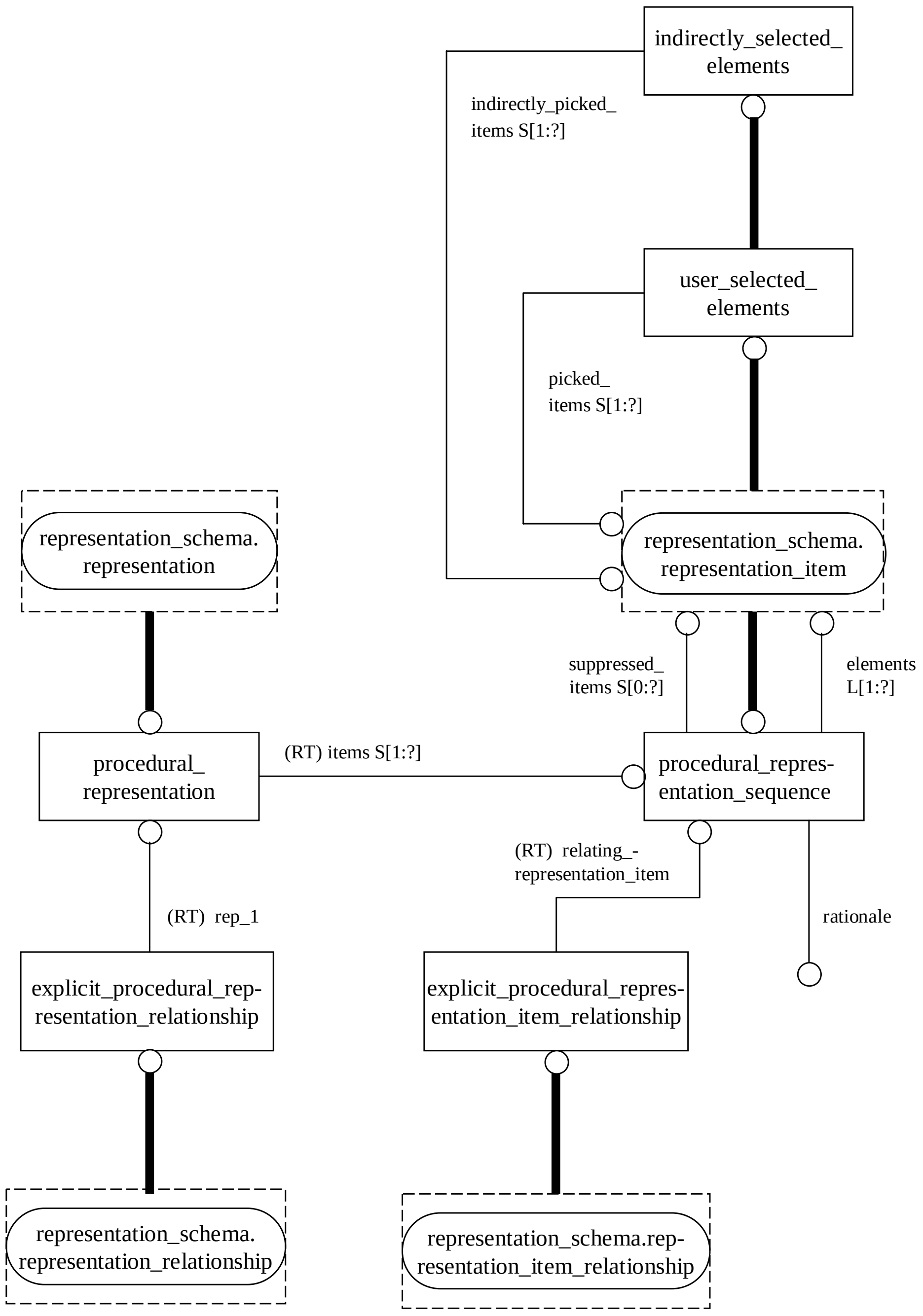
indirectly_selected_ elements
indirectly_picked_ items S[1:?]
user_selected_ elements
picked_ items S[1:?]
representation_schema. representation
representation_schema. representation_item
suppressed_ items S[0:?]
elements L[1:?]
procedural_ representation
(RT) items S[1:?]
procedural_repres- entation_sequence
(RT) relating_- representation_item
(RT) rep_1
rationale
explicit_procedural_rep- resentation_relationship
explicit_procedural_repres- entation_item_relationship
representation_schema. representation_relationship
representation_schema.rep- resentation_item_relationship

Figure D.1 – procedural_model_schema – EXPRESS-G diagram 1 of 1

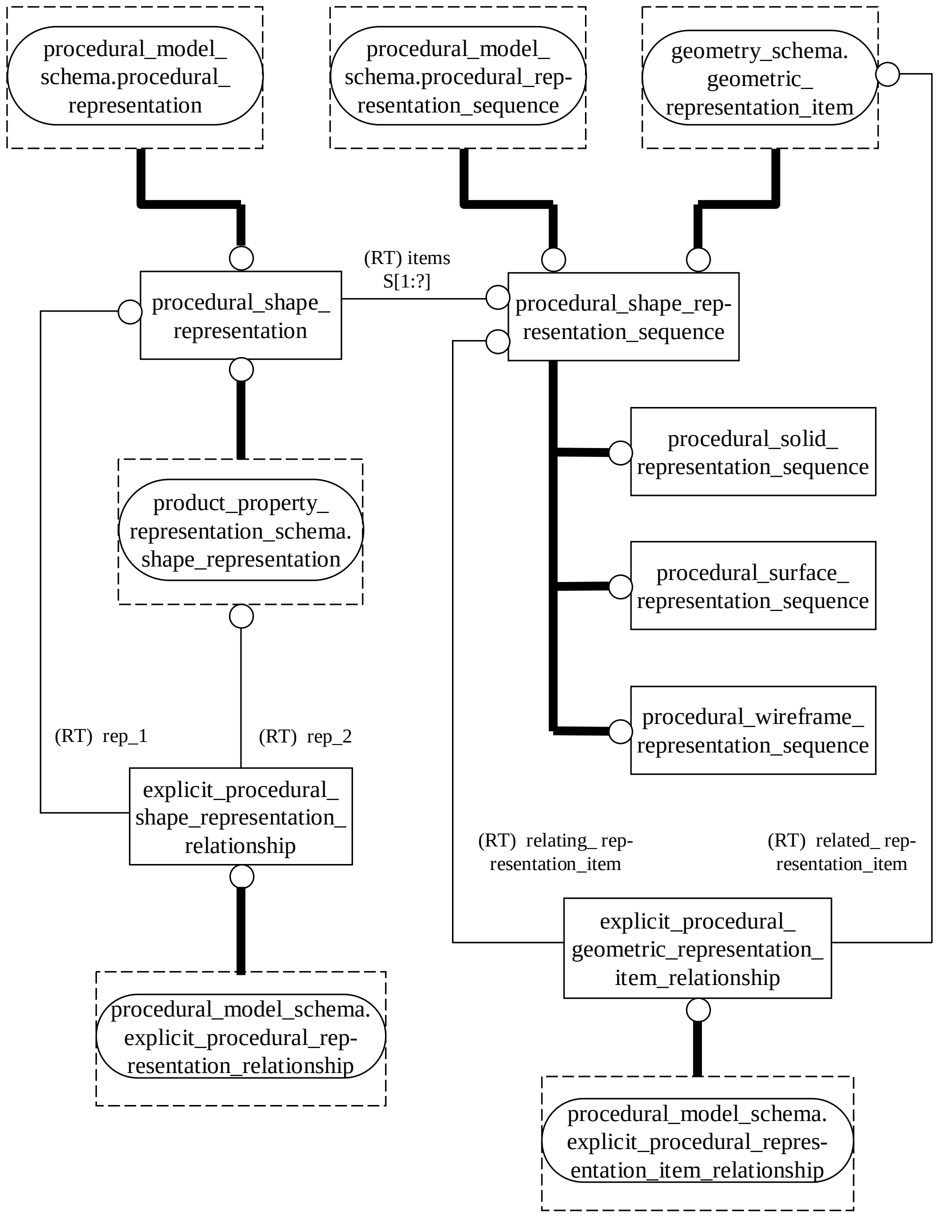
procedural_model_ schema.procedural_ representation
procedural_model_ schema.procedural_rep- resentation_sequence
geometry_schema. geometric_ representation_item
(RT) items S[1:?]
procedural_shape_ representation
procedural_shape_rep- resentation_sequence
procedural_solid_ representation_sequence
product_property_ representation_schema. shape_representation
procedural_surface_ representation_sequence
procedural_wireframe_ representation_sequence
(RT) rep_1
(RT) rep_2
explicit_procedural_ shape_representation_ relationship
(RT) relating_ rep- resentation_item
(RT) related_ rep- resentation_item
explicit_procedural_ geometric_representation_ item_relationship
procedural_model_schema. explicit_procedural_rep- resentation_relationship
procedural_model_schema. explicit_procedural_repres- entation_item_relationship

Figure D.2 – procedural_shape_model_schema – EXPRESS-G diagram 1 of 2

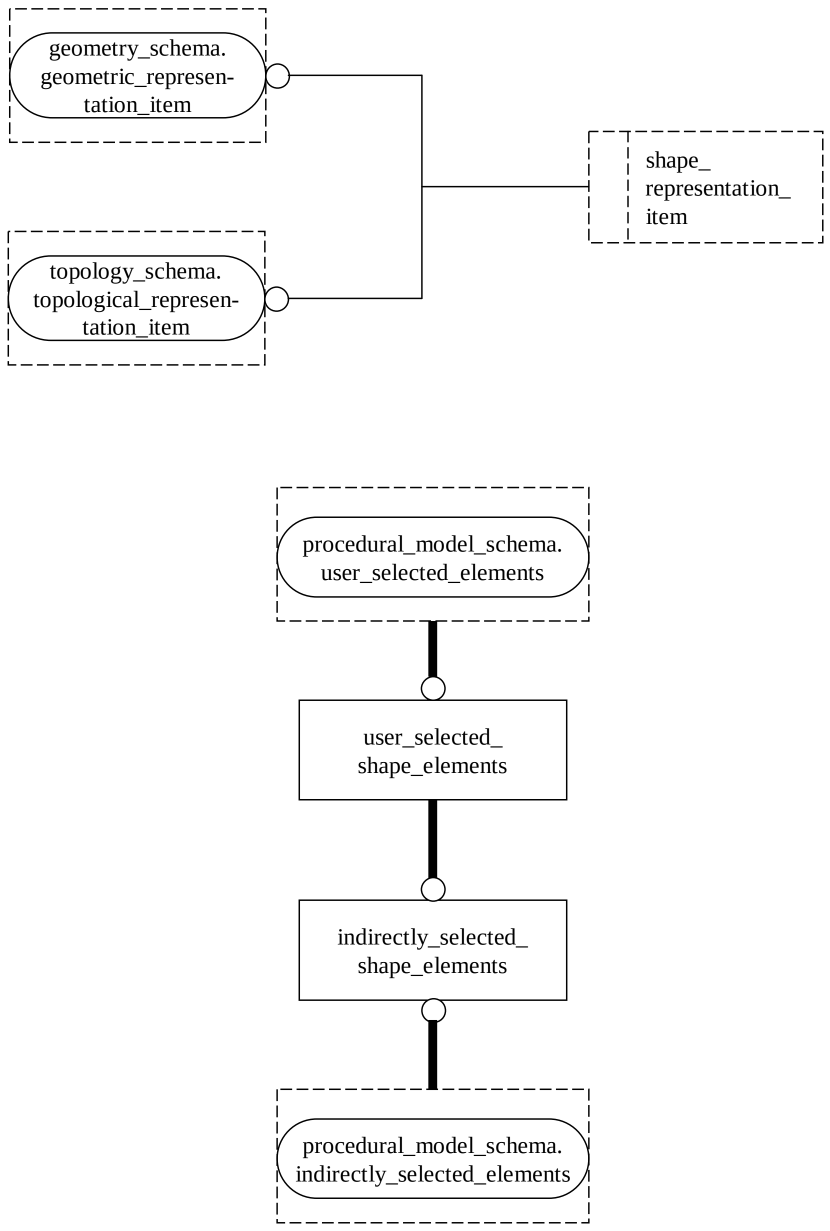

Figure D.3 – procedural_shape_model_schema – EXPRESS-G diagram 2 of 2

Annex E
(informative)

Examples of the use of this part of ISO 10303

E.1 Example of non-geometric application of procedural modelling

One example of a non-geometric procedural representation is that of a chess game specified in terms of its individual moves. The corresponding explicit representation could take the form of an 8×8 matrix with vector-valued elements specifying (i) the state of occupancy of the square on the chessboard corresponding to the matrix element, and (ii) if the square is occupied, the type and colour of the occupying piece (for example white pawn, black bishop).

This application well illustrates the difference between the nature of procedural and explicit models. An explicit model provides a snapshot of an evolving physical situation at some moment in time, whereas the corresponding procedural model gives details of the evolution of the model up to that point.

E.2 Example of intended usage of the procedural_shape_model_schema

This clause contains an example illustrating the intended use of the capabilities defined in this part of ISO 10303 for modelling the shape of a product. The current result model is assumed to be an ISO 10303-203 [4] model that precedes the procedural representation in an ISO 10303-21 [2] exchange file. The example shows the use of entity data types defined in other parts of ISO 10303 as constructors. Although most of these are defined in integrated resource parts, they are used here, for illustrative purposes, as though they were instantiable entities from an application protocol. The modelled form is an L-shaped block with a hole in it.

```
/* From current result model file */

#840 = ADVANCED_BREP_SHAPE_REPRESENTATION(.....,#850);
#850 = GEOMETRIC_REPRESENTATION_CONTEXT(.....);
...

/* Now start the procedural representation:  */

#1010 = EXPLICIT_PROCEDURAL_SHAPE_REPRESENTATION_RELATIONSHIP
        ('',$,#1020,#840);
#1020 = PROCEDURAL_SHAPE_REPRESENTATION('FINAL_OBJECT',
        (#1280),#850);

#1030 = PROCEDURAL_SOLID_REPRESENTATION_SEQUENCE('BASIC L-BLOCK',
        (#1040,#1050,#1060),(),'RATIONALE: TEXT...');
#1040 = EXTRUDED_FACE_SOLID('L-SOLID',#1070,#1080,8.);
#1050 = USER_SELECTED_SHAPE_ELEMENTS('SELECTED_EDGE',(#1120));
#1060 = CONSTANT_RADIUS_EDGE_BLEND('BLEND1',#1040,#1120,2.);

#1070 = FACE_SURFACE('L-FACE', .....);
#1080 = DIRECTION('EXTRUSION_DIRECTION',(0.,0.,1.));
/* supporting information for the extruded_face_solid #1040 */
.....
```

```
#1120 = EDGE_CURVE(.....);
/* supporting information for the edge_curve #1120 */
.....
#1180 = PROCEDURAL_SOLID_REPRESENTATION_SEQUENCE('HOLE-VOLUME',
        (#1190,#1200,#1210),(#1190,#1200,#1210 ),
        'RATIONALE: TEXT...');
#1190 = RIGHT_CIRCULAR_CYLINDER('HOLE-SHAFT',#1220,2.,1.);
#1200 = RIGHT_CIRCULAR_CONE('HOLE-BASE',#1250,1.,0.,45.);
#1210 = BOOLEAN_RESULT('HOLE-VOLUME',.UNION.,#1190,#1200);

#1220 = AXIS1_PLACEMENT('CYL-AXIS',#1230,#1240);
#1230 = CARTESIAN_POINT('FACE-CENTRE',(7.5,4.,4.));
#1240 = DIRECTION('CYL-AXIS-DIR',(0.,-1.,0.));
#1250 = AXIS1_PLACEMENT('CONE-AXIS',#1260,#1270);
#1260 = CARTESIAN_POINT('CONE-APEX',(7.5,1.,4.));
#1270 = DIRECTION('CONE-AXIS-DIR',(0.,-1.,0.));

#1280 = PROCEDURAL_SOLID_REPRESENTATION_SEQUENCE('FINAL-VOLUME',
        (#1290),(),'RATIONALE: TEXT...');
#1290 = BOOLEAN_RESULT('L-BLOCK-WITH-HOLE',.DIFFERENCE.,#1030,
        #1180);
```

It is assumed that the procedural model entries are preceded by an explicit representation of the model, summarized by the **advanced_brep_shape_representation** instance #840. The instance of **representation_context** associated with the explicit model is #850.

Instance #1010 defines the dual model relationship between the explicit and procedural representations of the L-block. Note that both representations share the same instance of **representation_context** (in this case that context is geometric), as required by a WHERE rule on **explicit_procedural_representation_relationship**. The file fragment contains three occurrences of **procedural_solid_representation_sequence** composing the procedural representation, instances #1030, #1180 and #1280. These sequences specify only the primary elements created; secondary elements involved in their definitions are, for purposes of clarity in this example, consistently listed after the instance of **procedural_solid_representation_sequence** in which they are used.

EXAMPLE Instance #1040 represents an operation for creating an **extruded_face_solid**, the base shape of the L-block. One of the attributes of that instance is the face to be extruded. That face is listed after the operation sequence, as instance #1070, followed by all the lower-level elements involved in its definition.

After instance #1040 has created the extrusion forming the basic shape of the L-block, the concave edge of the resulting volume is selected and a fillet operation performed on it. Note that the selection operation is indicated in the sequence of operations by an instance of **user_selected_shape_elements**. This refers to the selected element, and determines the appropriate use of that element in the succeeding operation.

Instance #1180 represents the shape of the hole in the block, which consists of a cylindrical portion and a conical tip; the individual volumes are created separately and fused together by means of a Boolean union operation. Instance #1280 defines the final model, which is obtained by Boolean subtraction of the hole shape from the L-block shape. Note that the two preceding operation sequences are reused as input to the third sequence, which illustrates the possibilities for embedding operation sequences at various levels in the representation of structured designs.

NOTE 1 The following instances in the above example are of entity data types defined in ISO 10303-42:

— **axis1_placement**;

— **boolean_result**;

— **cartesian_point**;

— **direction**;

— **edge_curve**;

— **extruded_face_solid**;

— **face_surface**;

— **right_circular_cone**;

— **right_circular_cylinder**.

All other instances in the procedural part of the example, except one, are of entity data types defined in this part of ISO 10303. The exception is instance `#1060` of a fictitious entity data type **constant_radius_edge_blend**, defining a constant radius blend feature on a single edge. Its attributes are a name (assumed to be inherited from its supertype **geometric_representation_item**, a reference to the solid on which the blend is to be defined, a reference to the specific edge to be blended, and the (real) blend radius.

NOTE 2 Primitive volumes for Boolean operations are positioned and oriented as they actually occur in the current result. It is believed that sending systems operating by creation in a default position followed by transformation can easily generate the required information. In most cases the primitives will not be represented explicitly in the current result.

NOTE 3 The use of three instances of **procedural_solid_representation_sequence** is purely for illustrative purposes — it demonstrates the embedding of operation sequences within each other. In the present case, the three final Boolean operations could have been replaced by two difference operations added to the end of the first sequence. However, as modelled here the hole volume could be separately instanced and used in a pattern feature, for example.

NOTE 4 As mentioned in clause 5.4.5, instances `#1210` and `#1290`, although not themselves instances of any subtype of **solid_model**, are deemed to define a result of type **solid_model** by virtue of their presence in a **procedural_solid_representation_sequence**.

NOTE 5 In the second instance of **procedural_solid_representation_sequence**, `#1180`, the attribute **suppressed_items** of the supertype **procedural_representation_sequence** contains three items. This has the effect of suppressing all the operations creating the volume of the hole in the L-block, leaving a null volume. When the model is first imported, all operations will be performed, but the receiving system will then be expected to reevaluate the model omitting these operations. The initial full evaluation permits the complete received model to be checked against the accompanying current result before generation of the simplified version.

E.3 Example of the use of variational (parameterization and constraint) information with a procedural model

The present part of ISO 10303 provides for the association of a procedural representation with an explicit representation, the *current result*, through the use of **explicit_procedural_representation_relationship** as defined in clause 4.3.1. In the context of this part of ISO 10303, the current result is what is obtained when the procedural model is evaluated. It is an invariant model, often of the boundary representation type.

ISO 10303-108 defines the concept of a **variational_representation**, a subtype of **representation** containing explicitly represented model parameters and constraints. Any instance of **variational_representation** is required to have an associated current result, a member of the parametric product family represented, specifically the one with current values of all the parameters involved. Use of the ISO 10303-108 entity data type **variational_current_representation_relationship** provides the necessary association.

An instance of **variational_representation** should refer to the current result via its procedural equivalent if one exists. The current result may then be regarded as a representative of the variational family that is subject to parametric modification through changes not only in explicit parameters defined in the model but also in implicit parameters occurring as arguments of constructional operations. Similarly, parametric modifications should conform not only to explicit constraints in the variational representation but also to implicit constraints inherent in constructional operations.

Four possibilities arise for combinations of procedural, explicit and variational representations. The three most important are

a) procedural and explicit alone: This case is straightforward. It requires the association of the explicit model with the procedural form by the use of **explicit_procedural_representation_relationship** as defined in clause 4.3.1 of this part of ISO 10303, or some specialized subtype of it.

b) variational and explicit alone: This case is also straightforward. The variational representation is effectively the explicit representation with an additional layer of parameter and constraint data associated with it to control the way it may be modified in a receiving system following a transfer. The association between the two models in this case is defined by the use of **variational_current_representation_relationship** as defined in ISO 10303-108. This type of association is appropriate for variational instances of **neutral_sketch_representation** as defined in ISO 10303-108.

c) procedural, variational and explicit: As might be expected, this is the most complex case. It is characterized by the direct association of variational information with elements involved in the procedural representation. The detailed example following this introductory material illustrates such a situation. A frequently occurring case will be that where the variational content of a hybrid model is confined entirely to the explicit elements involved, which will frequently be sketches with parameterized and/or constrained elements. The association of the procedural model with the explicit model is, as before, represented by the use of **explicit_procedural_representation_relationship**. However, in this case the use of **variational_current_representation_relationship** is also required, and here the current model must be referenced via its *procedural* form rather than its explicit form.

NOTE 1 In case (c), it will not be appropriate to associate the variational information directly with corresponding elements of the explicit current result model, because in general the elements created and subjected to constraints in the the procedural model may be modified or even deleted by the effects of subsequent operations, and therefore be absent from the final model corresponding to the current result. This is why the relationship of the variational representation with the current result must be via the procedural form in this case.

There is a fourth possibility, case (d), a combination of a procedural and a variational model in the absence of an explicit representation. This is possible; no instance of **explicit_procedural_representation_relationship** would be needed, and the two models present would be related by the use of **variational_current_representation_relationship**. However, the disadvantages of the lack of an explicit model make it unlikely that this option will be widely used.

NOTE 2 The relationships between the three types of representation for cases (a), (b) and (c) discussed above are illustrated in Figure E.1. The figure is intended to show relationships between specific instances rather than entity data types. The diagram for case (d) will be as for case (c) with the two leftmost instances omitted.

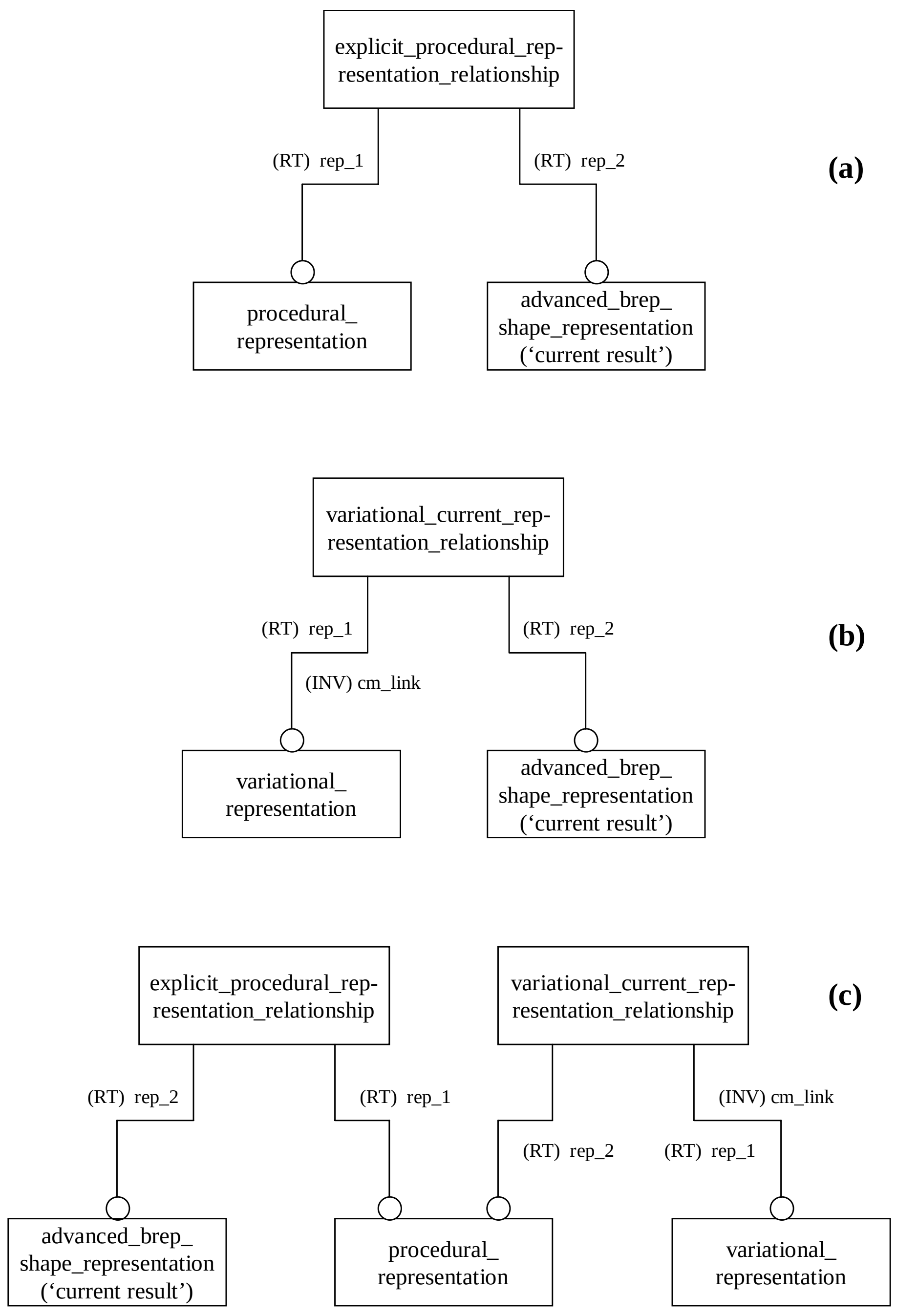
explicit_procedural_rep-resentation_relationship
(RT) rep_1
(RT) rep_2
(a)
procedural_ representation
advanced_brep_ shape_representation ('current result')
variational_current_rep-resentation_relationship
(RT) rep_1
(RT) rep_2
(b)
(INV) cm_link
variational_ representation
advanced_brep_ shape_representation ('current result')
explicit_procedural_rep-resentation_relationship
variational_current_rep-resentation_relationship
(c)
(RT) rep_2
(RT) rep_1
(INV) cm_link
(RT) rep_2
(RT) rep_1
advanced_brep_ shape_representation ('current result')
procedural_ representation
variational_ representation

Figure E.1 – Relationships between instances of procedural, variational and explicit models for the cases of (a) no variational model, (b) no procedural model, and (c) all three models.

The following fragment of an ISO 10303-21 [2] transfer file shows how the example of the previous clause can be extended by the inclusion of variational information:

```
/* Continuation of the example in the previous clause */

#1300 = FINITE_REAL_INTERVAL(0.,.OPEN.,2.,.CLOSED.);
#1310 = BOUND_MODEL_PARAMETER('R1',#1300,*,'HOLE-RADIUS',*);
#1320 = INSTANCE_ATTRIBUTE_REFERENCE
        ('GEOMETRIC_MODEL_SCHEMA.RIGHT_CIRCULAR_CYLINDER.RADIUS',
        #1190);
#1330 = BOUND_PARAMETER_ENVIRONMENT(#1310,#1320);


#1340 = FINITE_REAL_INTERVAL(0.,.OPEN.,2.,.CLOSED.);
#1350 = BOUND_MODEL_PARAMETER('R2',#1340,*,'CONE-BASE-RADIUS',*);
#1360 = INSTANCE_ATTRIBUTE_REFERENCE
        ('GEOMETRIC_MODEL_SCHEMA.RIGHT_CIRCULAR_CONE.RADIUS',#1200);
#1370 = BOUND_PARAMETER_ENVIRONMENT(#1350,#1360);


#1380 = EQUAL_PARAMETER_CONSTRAINT('',$,(#1310,#1350),());
#1390 = PDGC_WITH_DIMENSION('','TOTAL-HOLE-DEPTH',(#1260),(#1230),3.);
/* (PDGC stands for 'point distance geometric constraint') */


#1400 = FINITE_REAL_INTERVAL(0.,.OPEN.,5.,.CLOSED.);
#1410 = BOUND_MODEL_PARAMETER('H',#1400,*,'HOLE-SHAFT-DEPTH',*);
#1420 = INSTANCE_ATTRIBUTE_REFERENCE
        ('GEOMETRIC_MODEL_SCHEMA.RIGHT_CIRCULAR_CYLINDER.HEIGHT',
        #1190,);
#1430 = BOUND_PARAMETER_ENVIRONMENT(#1410,#1420);


#1440 = FINITE_REAL_INTERVAL(0.,.OPEN.,6.,.CLOSED.);
#1450 = BOUND_MODEL_PARAMETER('D',#1440,*,'TOTAL-HOLE-DEPTH',*);
#1460 = INSTANCE_ATTRIBUTE_REFERENCE
        ('EXPLICIT_GEOMETRIC_CONSTRAINT_SCHEMA.PDGC_WITH_DIMENSION.
        DISTANCE_VALUE',#1390,);
#1470 = BOUND_PARAMETER_ENVIRONMENT(#1450,#1460);


#1480 = FREE_FORM_RELATION('','DEPTH-RELATION',
        (#1450),(#1310,#1410),#1490);
#1490 = COMPARISON_EQUAL(#1450,#1500);
#1500 = PLUS_EXPRESSION(#1310,#1410);


#1510 = VARIATIONAL_REPRESENTATION('',(#1280,#1310,#1350,#1380,#1390,
        #1410,#1450,#1480),#850);
#1520 = VARIATIONAL_CURRENT_REPRESENTATION_RELATIONSHIP
        ('',$,#1510,#1020,#1020);
```

The file fragment provides a parameterization of the hole in the L-block object. It creates four instances of the ISO 10303-108 entity data type **bound_model_parameter**, in the file segments `#1300 – #1330, #1340 – #1370, #1400 – #1430` and `#1440 – #1470`, respectively. Each of these segments defines

a) A domain of validity for values of the parameter (in all cases, a finite range of non-zero reals);

b) The model parameter itself, which has a name, a reference to its domain and a description. The final attribute, shown as derived, is the value of the parameter, derived from the attribute to which the parameter is bound;

c) A specific attribute of an instance in the file to which the model parameter is bound, and whose value it represents;

d) A **bound_parameter_environment** instance, which provides the essential link between the model parameter and the attribute it is bound to.

The attributes that have parameters associated with them are

e) The radius of the cylinder forming the shaft of the hole;

f) The base radius of the cone forming the bottom of the hole;

g) The height of the cylinder forming the shaft of the hole;

h) The distance from the top centre of the hole shaft to the apex of the conical bottom surface.

Two explicit constraint instances are present. The first is an instance of **equal_parameter_constraint**, which simply requires the cylinder radius and the base radius of the cone to have equal values. The second is a constraint on the distance between the top and bottom points of the hole, which is required to be equal to the sum of the height and the radius of the cylinder. Because the cone apex angle is 45° and the cone base radius is equal to the cylinder radius, this sum gives the total depth of the hole.

The two final instances define a **variational_representation** and its link to the corresponding current result, in this case via the equivalent procedural form. The variational representation refers to a set of instances of **representation_item** that includes all those referred to by the (procedural equivalent of) the current result and also all the instances of explicitly represented parameters and constraints defining the variational aspect of the model. The fact that the procedural form of the current result is referenced twice by the instance of **variational_current_representation_relationship** is a requirement of the EXPRESS definition of that entity, and stems from the need to apply a local uniqueness rule.

NOTE 3 The capture and transfer of parameterization and constraint data are discussed in more detail in ISO 10303-108, which also provides further examples.

NOTE 4 All instances in this example are of entity data types defined in ISO 10303-108, with the exceptions of two entity data types that are defined in ISO 13584-20 [5]. That resource is used by ISO 10303-108 for the representation of mathematical expressions and relationships. The relevant entities are:

— **comparison_equal**;

— **plus_expression**.

E.4 Example of the embedding of operation sequences and the recording of design rationale

The entity data type **procedural_representation_sequence** specifies a sequence of instances of **representation_item** to be interpreted as constructional operations, and is itself defined as a subtype of **representation_item**. An instance of **procedural_representation_sequence** may therefore be included in the operation list of another instance of the same type. Thus embedding of operation sequences within each other is possible, and this facility may be used to build complex structures reflecting various simultaneous or sequential activities in an actual design process. More particularly, the same is true of the entity data type **procedural_shape_representation_sequence** and its subtypes, appropriate for the capture of the procedurally defined geometric models generated by CAD systems. A simple example of the embedding of geometric design sequences was given in clause E.2 of this annex.

In order to improve the readability and re-usability of a procedural design sequence or operation history, it is important to capture the rationale underlying the design process represented. The operational history itself records important aspects of the design intent, including the parameterization scheme and design constraints. The design rationale provides the underlying justification for the design intent inherent in the transferred procedural model.

The following schematic example illustrates a possible structure for the design of the cylinder head of an automobile engine. The abbreviations **OP** (short for ‘operation’) and **OPSEQ** (short for ‘operation sequence’) are used to denote an instances of **geometric_representation_item** and **procedural_shape_representation_sequence**, respectively.

```
OPSEQ_1 (name: complete cylinder head design,
          rationale: set of assertions);
  OP_11;
  OP_12;
  OPSEQ_11 (name: exhaust port,
            rationale: set of assertions);
    OP_111;
    OP_112;
    ...
    OP_11l;
  ENDSEQ;
  OP_13;
  OPSEQ_12 (name: water jacket,
            rationale: set of assertions);
    OP_121;
    OP_122;
    ...
    OP12m;
  ENDSEQ;
  OP_14;
  OP_15;
  OPSEQ_13 (name: main cylinder head,
            rationale: set of assertions);
    OP_131;
    OP_132;
    ...
    OP_13n;
```

  ENDSEQ;
  OP_16;
  OP_17;
ENDSEQ.

In the above example the abbreviation **ENDSEQ** denotes the end of an operation sequence. There is a main operation sequence, which contains three embedded operation sequences together with some individual operations. The embedded sequences are concerned with the design of specific elements of the overall cylinder head. Only one level of embedding is shown above, but in principle arbitrarily many levels of embedding are permitted. The order of embedding of subsidiary sequences within the main sequence reflects the order of creation of the various elements of the cylinder head, which may in turn reflect dependencies of the later parts of the design on details of the earlier parts.

The assertions made in the design rationale give the reasons why particular design choices are made. Those choices may to some extent be represented by the application of constraints and the assignment of parameter values and ranges of validity, together with relations between parameters. Such information may be included implicitly (through the choice of operations and their input arguments) or explicitly (as illustrated in the previous clause). The design rationale information may include textual specifications of design criteria, design constraints and design confirmation procedures that cannot be captured by the ISO 10303 capabilities currently available. It may also be used to record key results from analysis (for example, finite element calculations) and to store them for later reuse.

EXAMPLE 1 The following types of design constraints are illustrative of those that can be represented in a procedural model using the capabilities of ISO 10303-108:

— The distance between specified surfaces of two objects shall lie between L_1 and L_2;

— The values of model parameters P_1, P_2 and P_3 shall be related by $P_3 = 5P_1 + P_2$.

EXAMPLE 2 Other types of design constraints cannot currently be represented in a procedural model using existing ISO 10303 capabilities. These include such cases as

— 'The number of supporting ribs shall have a specified dependency on the value of parameter H_4'. This is not possible because it requires the generation of a number of ribs that depends upon the result of some prior calculation. ISO 10303 currently provides no means for the incorporation of the necessary control structures such as IF...THEN...ELSE or REPEAT...UNTIL in procedural models;

— 'It shall be verified that there is sufficient space for engine vibration, and then ensured that resonator capacity is within acceptable limits, with tolerance T_1'. These are complex tasks requiring the use of analytical capabilities not currently covered by ISO 10303.

The best that can be done in these last two cases is to provide a textual description of what is required.

The intention is that, following an ISO 10303 exchange of a model, the responsibility for confirmation of design intent can be transferred to the receiving system. Validation of constraints may occur after a single design operation or after multiple operations. For example, a simple design constraint may be checked for continued validity after a single modification operation. On the other hand, a complex analytical check such as verification of structural integrity using finite element analysis is normally only possible for a fairly complete model defined in terms of many operations.

Finally, it is emphasized that that the capability for capturing design rationale information in textual form, as provided in this part of ISO 10303, is regarded as minimal, and that extensions may be made in this respect in future versions of the document. One possibility would be to include computer-interpretable pointers to external documents.

Bibliography

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

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

[3] ISO 10303-109, *Industrial automation systems and integration — Product data representation and exchange — Part 109: Integrated application resource: Kinematic and geometric constraints for assembly models.*

[4] ISO 10303-203, *Industrial automation systems and integration — Product data representation and exchange — Part 203: Application protocol: Configuration controlled design.*

[5] ISO 13584-20, *Industrial automation systems and integration — Parts library — Part 20: Logical resource: Logical model of expressions.*

[6] HOFFMANN, C.M., *Geometric and Solid Modeling*, San Mateo, CA: Morgan Kaufmann, 1989.

[7] HOSCHEK, J.; DANKWORT, W. (editors). *Parametric and Variational Design*, Stuttgart: Teubner-Verlag, 1994.

[8] RAPPAPORT, A. *Breps as Displayable-Selectable Models in Interactive Design of Families of Geometric Objects*, in *Geometric Modeling: Theory and Practice* (edited by W. Strasser, R. Klein and R. Rau); Berlin: Springer-Verlag, 1997.

[9] SHAH, J. J.; MÄNTYLÄ, M. *Parametric and Feature-based CAD/CAM*, New York: Wiley, 1995.

Index

ICS 25.040.40

Price based on 48 pages