A New ISO 10303 (STEP) Resource for Modeling Parameterization and Constraints

Michael J. Pratt
e-mail: mike@lmr.clara.co.uk
LMR Systems,
21 High Street, Carlton, Bedford, MK43 7LA, UK

ISO 10303 (STEP) is a standard for the electronic exchange of product life-cycle data. It has been continually expanded by the addition of new capabilities since its first release by ISO in 1994. Until now, the standard has not been capable of exchanging models with parameterization and constraints such as those commonly generated by modern CAD systems. This makes it difficult to modify a model for any purpose following an exchange. However, new STEP resources have been developed to rectify this situation, and the paper reports on the the first of them to reach International Standard status. It provides the capability to associate explicitly represented parameterization and constraint information with CAD models. The primary application areas are in the 2D profiles or sketches that are inherent in many CAD model constructional procedures, in the positioning and orientation of features in 3D models and in the positioning and orientation of assembly components. Other complementary new STEP resources are briefly discussed at the end of the paper. [DOI: 10.1115/1.1814383]

1 Introduction

The international standard ISO 10303 for the electronic exchange of product life-cycle data has been adequately described elsewhere (e.g., [1,2]). Work on it commenced in 1984, and 10 years were taken in developing the foundations for ISO's first computer-interpretable standard and building an appropriate structure on them before the first version of ISO 10303 was published in 1994.

Meanwhile important developments had occurred in CAD technology which, because of the nature of the ISO standardization process, could not be taken into account in the first release of the standard. In particular, the general adoption in CAD systems of procedural or construction history modeling, and the inclusion of parameterization and geometric constraint capabilities, made it much easier to edit models than had been the case in earlier systems. However, these new capabilities arrived too late to be included in the first release of STEP, because the technical content of an ISO standard is frozen at an early stage in its development. The result is that, while the standard as it currently exists apparently allows the exchange of CAD models with a high degree of success, the models as received after an exchange lack crucial information that enables them to be modified effectively in the receiving system. Although they correspond geometrically and topologically with the model in the originating system, they are essentially inert. In consequence, much operator time is now spent in trying to reconstruct ''design intent'' following the transfer of CAD models so that those models can be optimized for applications downstream of design.

2 Basic Principles of ISO 10303-108

The first basic principle to be observed in extending the standard to include the new CAD system capabilities mentioned above was that of upward compatibility with existing parts of ISO 10303—no changes could be made to those parts already published. ISO 10303-108 [3], the new resource part that is the primary focus of this paper, could make references into models defined using previous STEP resources, but references from such models to entities defined in the new resource were not permitted because that would have required forbidden changes.

The solution chosen was the provision of a ''wrapper'' around the nonparametric model. This outer layer contains details of explicitly defined parameters and their connections with elements of the interior model, and also of the explicitly defined constraint relationships between elements of that model.[1] The interior model itself is then regarded as a representative example of a parametric family of models, the one corresponding to the particular set of parameter values and constraints defined in the wrapper. It is referred to as the *current result*.

At the time when a model is exchanged, the information in the wrapper is redundant as far as its embedded component is concerned. It is only after the exchange is complete that this information comes into play, and its role is then to control the manner in which the model may be edited in the receiving system. The preferred method of editing is simply by changing the values of parameters. The wrapper gives details of all the explicitly defined parameters that are available for this purpose. If one or more of them is changed, then the embedded model must be regenerated, subject to the constraints that are also present in the wrapper.

Most of ISO 10303-108 is concerned with the capabilities of the wrapper, but an additional schema is also provided, specifying representations for the two-dimensional profiles or sketches that are widely used in CAD systems as the basis for constructing three-dimensional volumetric elements, typically by the use of sweeping operations such as extrusion or rotation about an axis. The following sections describe the contents of the five schemas comprising ISO 10303-108 in some detail.

The information modeling language used in ISO 10303 is called EXPRESS, and it is part of the standard itself [4]. No detailed knowledge of EXPRESS is needed for understanding the examples given in the paper; it is sufficient to know that the language provides supertypes and subtypes, and that these exhibit inheritance of attributes in a manner similar to that of object-orientated modeling. Entity data types and their attributes as modelled in EXPRESS are denoted in what follows by the use of **bold** type. Also, it should be noted that ISO 10303-108 frequently defines new entity data types as subtypes of entities defined in other resource parts of the standard. This is ''standard'' practice, related

Contributed by the Engineering Informatics (EIX) Committee for publication in the JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING. Manuscript received March 31, 2004; revised September 12, 2004. Associate Editor: R. Rangan.

[1]CAD models often also exhibit *implicit* parameters and constraints. An implicit parameter is an input argument to a constructional procedure (see Section 4), and an implicit constraint is a constraint inherent in the operation of such a procedure.

to the fact that, as stated in the Foreword to every published part of ISO 10303 "*The integrated generic resources and the integrated application resources [of the standard] specify a single conceptual product data model*."

3 ISO 10303-108 Schemas

3.1 parameterization_schema. The **parameterization_schema** is concerned with the definition of parameters and their association with quantities in the embedded model or current result. In fact the word "parameter" was already used with several different meanings in ISO 10303, and so the basic entity data type in ISO 10303-108 was named **model_parameter** to ensure a clear distinction from other usages. It is an *abstract* or non-instantiable supertype (i.e., no instance of it can occur in a model exchange file), but has the instantiable subtypes described below. An instance of a subtype of **model_parameter** is regarded as a mathematical variable; it has a domain of validity and a current value. Two instantiable subtypes are defined, **bound_model_parameter** and **unbound_model_parameter**.

An instance of **bound_model_parameter** is associated with an attribute of an entity data type instance in the embedded current result model. Its current value is the value of that attribute. The necessary association or binding is made via the entity data type **attribute_reference**, which itself has two attributes. The first is of type **representation_item**, defined in ISO 10303-43 [5], which allows it to point to any instance of an element used in building the current result. The second is a specialized text string referring to the name of the attribute of the specified instance to which the binding is made.

Consider, for example, the ISO 10303-42 [6] entity data type **circle**. This has the EXPRESS definition

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

The significance of this is fairly self-evident. An instance of a circle will appear in an ISO 10303-21 exchange file [7] in the form

```
#345 = CIRCLE('C1',#325,15.0);
```

The initial number is the reference number of the instance in the file. The first two values inside the parentheses are values of attributes that **circle** inherits from its supertypes; the first is just a textual name, and the second is a reference to an **axis_placement** or local coordinate system represented by instance #325 in the file (not shown). The third attribute value, 15.0, is the value of the radius attribute of the circle instance.

To associate a **bound_model_parameter** with the radius attribute of the circle instance, an instance of **attribute_reference** must first be created. As mentioned above this has two attributes; the first is an attribute name (a text string) and the second has the ISO 10303-43 type **representation_item**. In the present case the **attribute_reference** instance will occur in the exchange file as

```
#355 = ATTRIBUTE_REFERENCE('RADIUS',#345);
```

Here the first attribute is the name of the attribute of **circle** to which the parameter is bound, and the second is a reference to the intended specific instance of **circle** in the exchange file. This is a valid reference because **circle** is a subtype of **representation_item**.

Finally, the link between the parameter and the attribute is made through an instance of the entity data type **bound_parameter_environment** (the name does not intuitively indicate its significance, but was required for compatibility with previously existing resources). This entity has two attributes; the first refers to a variable and the second to its interpretation (in some sense). In the present case the variable is a **bound_model_parameter** instance and the interpretation an **attribute_reference** instance. The following fragment of an exchange file illustrates the principle. The **circle** instance occurs first. This is succeeded by instances of **bound_model_parameter**, **attribute_reference** and **bound_parameter_environment**, the last providing the association between the parameter and the referenced attribute of the **circle** instance.

```
#345=CIRCLE('C1',#325,15.0);
#350=BOUND_MODEL_PARAMETER(...,*);
#355=ATTRIBUTE_REFERENCE('RADIUS', #345);
#360=BOUND_PARAMETER_ENVIRONMENT (#350,#355);
```

Some attributes of the **bound_model_parameter** have been omitted, including a name, a reference to a domain of validity and a textual description. The one attribute shown, denoted by an asterisk, represents the value of the parameter, which cannot be given explicitly in practice. The actual value is of course 15.0, the value of the radius attribute of the circle, but the EXPRESS language provides no means for extracting that value and writing it into the appropriate location in #350. The asterisk in the file fragment above denotes that the value is "derived"—the process of derivation is the responsibility of the translation software.

In the example given above, the **circle** instance will belong to the embedded model or current result, but the **model_parameter** instance will belong to the wrapper that surrounds it. It provides subsidiary information concerning a valid way of modifying the model following its transfer into a receiving system.

An instance of **unbound_model_parameter**, as its name implies, is not bound to an entity instance attribute. Instead, it participates in a mathematical relationship with other parameters, some of which may be instances of **bound_model_parameter**. Provision for the specification of mathematical constraint relationships is made in the next schema to be described, the **explicit_constraint_schema**.

To illustrate the use of an **unbound_model_parameter** instance, we consider a rectangular block. Suppose that x, y and z represent bound model parameters associated with the length, width and height of the block. We may now introduce another parameter $t>0$, and define relationships $x=3t-2$, $y=t^2$ and $z=t^2-3$. The new parameter can be represented by an instance of **unbound_model_parameter**. It is not bound to any physical quantity in the model, but it is used to control the values of three instances of **bound_model_parameter**. A major distinction between a bound and an unbound **model_parameter** is that the value of the first must be written in the exchange file as "derived," for the reason explained above, but an explicit value can be provided for the second. In an actual model exchange this should, of course, be compatible with the values of the attributes it controls through relations with **bound_model_parameter** instances.

It should be noted that, although one attribute of a **model_parameter** is a name, that name has little significance in an ISO 10303 model. As shown above, in a file exchange a parameter instance is referred to by its reference number in the exchange file, and not by name. During the transfer of the model, the receiving system will generate its own internal identifiers for the elements composing the model. There would be little point in transferring the internal system identifier generated by the sending system, since the naming conventions of that system will be different from those of the receiving system. However, the name attribute of a parameter may be used to capture and transfer any descriptive name that the user of the sending system had associated with it.

A further entity data type defined in the **parameterization_schema** is the **fixed_instance_attribute_set**. This is simply a set of instances of **attribute_reference** as described above. The occurrence of an instance of **fixed_instance_attribute_set** is simply an assertion that the set of individual attributes it refers to all have fixed values. Equivalently, the domain of their range of validity has zero

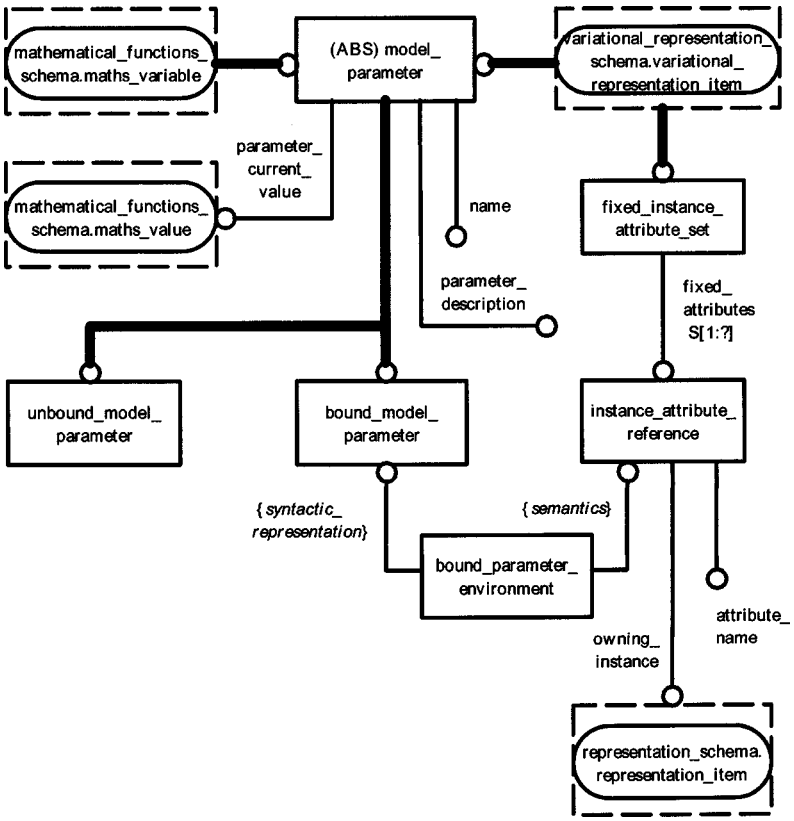

Fig. 1 ISO 10303-108 parameterization_schema

extent. Effectively, this is an instruction to the receiving system that the value of the indicated attributes should remain constant in any modification of the model following a transfer.

The ISO 10303-108 **parameterization_schema** is summarized in Fig. 1, which uses the EXPRESS-G graphical representation of the EXPRESS language. EXPRESS-G is fully defined in [4]. The diagram does not make use of all its capabilities, and some slight simplifications have been made in the interests of clarity. Its interpretation is as follows:

⇒ Rectangular boxes represent entity data types. In abstract (non-instantiable) cases the entity name is preceded by '(ABS).'

⇒ Thick lines denote subtype relationships, a subtype being distinguished by a small circle at the end of the link.

⇒ Thin lines show attribute relationships, the type of the attribute similarly being distinguished by a small circle. The attribute name is shown beside the link. L[x:y] or S[x:y] indicates that the attribute value is a list or a set of instances of the indicated type, and gives the upper and lower limits of the number of members of the aggregate ("?" is used for an indeterminate upper limit).

⇒ Attributes whose links terminate in small circles but do not point to entity boxes are essentially of basic types such as real, integer, Boolean or string.

⇒ Entity data types imported from other schemas are indicated by dashed boxes with the entity name, preceded by the name of the schema where it is defined, in an enclosed round-ended box. Some of those schemas are included in ISO 10303-108 and are described in this paper; in other cases the names of the referenced entities are fairly self-explanatory.

⇒ The code specifying EXPRESS entities may contain rules imposing restrictions on the validity of instances of those entities, but these are not represented in EXPRESS-G diagrams.

⇒ Simplified links are shown by italicised attribute names (note, however, that this is not standard EXPRESS-G practice).

3.2 explicit_constraint_schema. In CAD models, constraints can occur in two forms:

• An implicit constraint is built into the operation of a constructional operation. Example: an operation to create a rectangle on the screen of the system *automatically* creates a quadrilateral in which opposite sides are parallel and adjacent sides perpendicular.

• An explicit constraint, by contrast, is one that is deliberately added by the system user. Example: the user may create a quadrilateral by picking four corner points, and then subsequently apply two parallelism constraints as relationships between pairs of opposite sides. An angular constraint may then be applied to fix one corner angle of the resulting parallelogram as 60 deg. Most dimensions specified in a CAD model are cases of explicit constraints.

The ISO 10303-108 **explicit_constraint_schema**, as its name implies, is concerned only with explicit constraints. Implicit constraints are handled by the procedural representation capability mentioned later in the paper. An abstract (non-instantiable) supertype **explicit_constraint** is defined, which has two subtypes, **free_form_constraint** and **defined_constraint**.

The entity data type **free_form_constraint** is again abstract; it is the supertype of all constraints defined in terms of explicit mathematical relationships between instances of **model_parameter** subtypes (referred to simply as "parameters" in what follows). Its instantiable subtypes are

• **free_form_assignment**—this entity data type specifies the values of one or more parameters as being equal to the value of a mathematical expression involving other parameters. This type of constraint would be used to model the assignments of values to

parameters represented by x, y and z in terms of the unbound parameter t in the **unbound_model_parameter** example given in the last section.

• **free_form_relation**—this entity data type specifies a mathematical equation or inequality in which two or more parameters participate. These parameters are divided into two subclasses; one set is regarded as providing input values, and the other output values—it is members of the second subclass whose values are constrained, in terms of values of members of the first subclass.

As an example of the use of the **free_form_relation** constraint, consider the following relation between parameters represented by a, b, c and d:

$$a^2+b^2-(c^2+d^2)>0$$

Here a and b may be input or reference parameters, and the constraint used to control the values of c and d. However, a and b may themselves be output or constrained parameters in some other constraint relation, and this provides the possibility for hierarchies of constraints in a model.

The representation of mathematical relationships used in these constraints is defined by the resource ISO 10303-50 (''Mathematical constructs'') [8]. Such relationships are not represented in the form of strings, as in programming languages, but rather as tree structures built in terms of individual arithmetical and relational operators, standard functions and operands.

The entity data type **explicit_constraint** has a second subtype, **defined_constraint**. This is again abstract, and it has just one instantiable subtype in the **explicit_constraint_schema**. However, it has another very important subtype in the next schema to be discussed, the **explicit_geometric_constraint_schema**.

A **defined_constraint** is a constraint in which no explicit mathematical relationship is defined. It is intended for use in cases where the semantics of the constraint are well understood by both the sending and receiving system. In such a situation it is sufficient to send the constraint in a descriptive form, allowing the receiving system to formulate it mathematically in whatever manner is best suited to its internal functionality. An example is given in the next section. The single instantiable subtype of **defined_constraint** provided in the present schema is **equal_parameter_group**, which simply lists a set of parameters whose values are all constrained to be equal. This is such a common requirement that it is not felt necessary to spell out even such a simple mathematical relationship as equality between the values of the parameters concerned.

Finally, the **explicit_constraint_schema** provides the construct **simultaneous_constraint_set**, which allows the grouping together of a set of constraints that are required to hold simultaneously in the model. Mathematically, this implies the requirement for the simultaneous solution of a set of constraint relationships in the receiving system. If one or more of those relationships is nonlinear, such a system will in general have multiple solutions. The designer's choice of solution in the sending system will be indicated by the configuration of the embedded or current result model mentioned earlier.

It should be noted that ISO 10303-108 contains no representation for the sequential application of constraint relationships, except insofar as such sequences are implied by the hierarchical systems of constraints mentioned earlier in this section. Sequences in general imply time dependency, which is outside the scope of ISO 10303-108 but lies in the domain of the complementary resource ISO 10303-55 (''Procedural and hybrid representation'') [9], briefly discussed later in the paper.

The structure of the ISO 10303-108 **explicit_constraint_schema** is illustrated in EXPRESS-G form in Fig. 2. The explanatory notes to Fig. 1 apply, and in this diagram there is the additional feature that attribute definitions made at the supertype level are in some cases refined at the subtype level—this is denoted by ''(RT)'' before the name of the redefined attribute, the abbreviation standing for ''redefined type.''

3.3 explicit_geometric_constraint_schema. The constraint capabilities described in the preceding section have very general applicability, and can be employed in any kind of model. However, one very important application area is that of the shape models generated by CAD systems, in which many kinds of geometric constraints may be applied between the modeling elements. The **explicit_geometric_constraint_schema** specifies a range of constraints of this type. Such constraints are widely implemented in CAD systems, and it is therefore not necessary to define them mathematically—since the semantics of a constraint of this type will be common to the sending and receiving systems, a descriptive form of representation will be sufficient. The use of subtypes of the **defined_constraint** defined in the previous schema is therefore appropriate.

At the highest level, the **explicit_geometric_constraint_schema** defines an abstract supertype **explicit_geometric_constraint**, a subtype of **defined_constraint**. All constraints of this type share two attributes, one specifying a set of one or more *constrained elements* and the other a set of zero or more *reference elements*. These elements are required to be of the ISO 10303-42 type **geometric_representation_item**, which includes as subtypes such things as points, curves and surfaces, the individual geometric elements underlying shape models.

If reference elements are present in an instance of such a constraint, then the constraint is said to be *directed*; if not, it is *undirected*. Consider, by way of example, the case of two lines constrained to be parallel. The directed case may be stated as ''line A is parallel to line B,'' and in this case line A is constrained with respect to line B, the reference element. The undirected case corresponds to ''lines A and B are parallel,'' and here both lines play the same role in the constraint; there is no reference element. The two cases may alternatively be described as *asymmetric* and *symmetric*, because the two lines play different roles in the first case but equal roles in the second.

The **explicit_geometric_constraint_schema** defines a set of specific subtypes of **explicit_geometric_constraint**, all inheriting the two attributes discussed above. The first-level subtypes are briefly described in the following paragraphs. In every case where a ''basic form'' is mentioned there exists a second-level dimensional subtype with modified semantics. These further subtypes are described later.

3.3.1 **fixed_element_geometric_constraint**. There is no reference element. The constrained elements may belong to any type of **geometric_representation_item**, which includes high-level constructs such as **solid_model**, **surface_model** and **wireframe_model**. The members of the set of constrained elements are frozen and not available for editing following a model transfer.

3.3.2 **parallel_geometric_constraint**. Zero or one reference elements may be present. The constrained and reference elements are lines or planes. In the undirected case the members of the set of constrained elements are all parallel to each other; in the directed case they are all parallel to the reference element.

3.3.3 **point_distance_geometric_constraint**. The constrained elements are points. They are constrained to lie at the same (unspecified) distance from up to four reference elements, which may be points, curves or surfaces. This constraint may be used, for example, to constrain a point to lie equidistant from the four faces of a tetrahedron. In this basic form the constraint may only be instanced as a directed constraint.

3.3.4 **skew_line_distance_geometric_constraint**. The elements involved are two nonintersecting lines, one of which may be a reference element. The distance between the lines is specified, along their common normal.

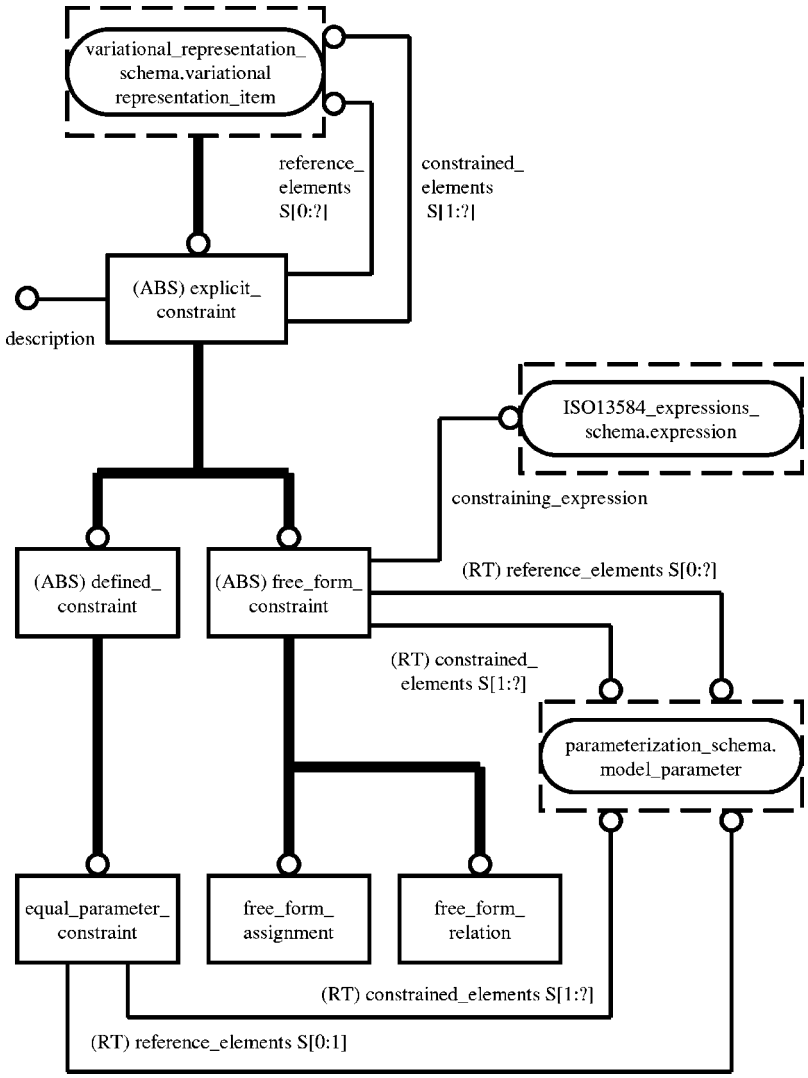

Fig. 2 ISO 10303-108 explicit_constraint_schema

3.3.5 **curve_distance_geometric_constraint**. The constrained elements are curves. In the basic form a single curve is constrained so that its minimum distances from up to three reference elements are equal.

3.3.6 **surface_distance_geometric_constraint**. The constrained elements are surfaces. In the basic form a single surface is constrained so that its minimum distances from up to three reference elements are equal.

3.3.7 **radius_geometric_constraint**. This is an undirected constraint, having no reference element. It constrains the radii of a set of elements to be equal, those elements being of any geometric type characterized by a single radius attribute, e.g., **circle**, **cylindrical_surface** or **spherical_surface** as defined in ISO 10303-42.

3.3.8 **curve_length_geometric_constraint**. This is an undirected constraint, having no reference element. It constrains the lengths of a set of elements to be equal, those elements being of the geometric type **bounded_curve** as defined in ISO 10303-42.

3.3.9 **parallel_offset_geometric_constraint**. This constraint is intended for use with approximate computed offset curves or surfaces. It can be used to constrain a set of two-dimensional curves in a plane, a set of three-dimensional curves, or a set of surfaces. These element types may not be mixed in a single instance of the constraint. The constraint may be used in undirected form to require all pairs of a set of constrained elements to possess the specified offset relationship. In its directed form the constraint requires a set of constrained elements all to have the same offset relationship with a single reference element.

3.3.10 **angle_geometric_constraint**. This type of constraint defines relationships between lines and/or planes. In its basic form it is a directed constraint, requiring all members of a set of such elements to make equal angles with a reference element.

3.3.11 **perpendicular_geometric_constraint**. Here the elements involved are lines or planes, and the constraint may be either directed or undirected. In the directed case the members of a set of constrained elements are required to be perpendicular to one or two reference elements of the same type (two reference elements are only allowed in 3D cases). In the undirected case the constrained elements may number two or three (three are only allowed in 3D cases), and they must all be either lines or planes, but not a mixture of the two.

3.3.12 **incidence_geometric_constraint**. This type of constraint requires one or more points, curves or surfaces to lie on (or be incident on) one or more reference elements of any of those

types. The inverse case, where the reference elements are included in one or more constrained elements, is also covered. There is also an undirected case, in which one of two elements is required to be incident on the other. Examples: the constrained elements might be a set of points, all required to lie on a specified reference curve. Alternatively, the points might be the reference elements, and the curve constrained to interpolate them.

3.3.13 **coaxial_geometric_constraint**. This constraint requires a set of elements with axisymmetric geometry to share the same axis. In the directed case, a single reference element is used to define that axis.

3.3.14 **tangent_geometric_constraint**. This constraint requires two curve or surface elements to be tangent to each other (an undirected relationship) or one or more curves or surfaces to be tangential to one or more specified reference elements of the same type (a directed relationship).

3.3.15 **symmetry_geometric_constraint**. This constraint requires two points, curves or surfaces to be symmetrically disposed with respect to a reference element. In 2D the reference element is a line, and in 3D it may be either a line or a plane.

3.3.16 **swept_point_curve_geometric_constraint**. This constraint relates to the sweeping of 2D profiles or sketches to generate 3D shapes. Each vertex point in the original profile sweeps out an edge curve of the 3D shape. The constraint requires that, if the profile is modified, the curves underlying the edges swept out by the motion of its vertices are appropriately recomputed according to the type of sweep motion specified.

3.3.17 **swept_curve_surface_geometric_constraint**. This constraint also relates to the sweeping of 2D profiles or sketches to generate 3D shapes. Each edge curve in the original profile sweeps out a face of the 3D shape. The constraint requires that, if the profile is modified, the surfaces underlying the faces swept out by the motion of its vertices are appropriately recomputed according to the type of sweep motion specified.

3.3.18 **curve_smoothness_geometric_constraint**. ISO 10303-42 allows the specification of the level of smoothness of each transition between the individual curve segments making up a composite curve. The simplest possibilities include, in order of increasing smoothness, positional continuity, tangent continuity and curvature continuity. The ISO 10303-42 capability allows the levels of smoothness to be indicated at the time of model transfer, but does not constrain them to be maintained if the received model is edited in the receiving system. The **curve_smoothness_geometric_constraint** provides this additional capability.

3.3.19 **surface_smoothness_geometric_constraint**. ISO 10303-42 also allows the specification of the level of smoothness of each transition between the individual surface patches making up a composite surface. Again, the ISO 10303-42 capability allows the levels of smoothness to be indicated at the time of model transfer, but does not constrain them to be maintained if the received model is edited in the receiving system. The **surface_smoothness_geometric_constraint** provides this additional capability.

As mentioned earlier, several of the constraint types described above have dimensional subtypes. These are given below.

3.3.20 **parallel_geometric_constraint_with_dimension**.This subtype of **parallel_geometric_constraint** provides for the specification of a dimension between parallel lines and/or planes.

3.3.21 **point_distance_geometric_constraint_with_dimension**. This subtype of **point_distance_geometric_constraint** provides for specification of a dimension locating one or more points with respect to other geometric elements.

3.3.22 **curve_distance_geometric_constraint_with_dimension**. This subtype of **curve_distance_geometric_constraint** provides for specification of a dimension locating one or more curves with respect to other geometric elements.

2.2.12 **surface_distance_geometric_constraint_with_dimension**. This subtype of **surface_distance_geometric_constraint** provides for specification of a dimension locating one or more surfaces with respect to other geometric elements.

3.3.24 **radius_geometric_constraint_with_dimension**. This subtype of **radius_geometric_constraint** constrains the members of a set of circular, cylindrical or spherical elements to have the same specified radius.

3.3.25 **curve_length_geometric_constraint_with_dimension**. This subtype of **curve_length_geometric_constraint** constrains all members of a set of bounded curves to have the same specified length.

3.3.26 **parallel_offset_geometric_constraint_with_dimension**. This subtype of **parallel_offset_geometric_constraint** requires an offset curve or surface to lie at a specified distance from the element from which it is offset.

3.3.27 **angle_geometric_constraint_with_dimension**. This subtype of **angle_geometric_constraint** constrains the value of the angle between line or plane elements.

All the constraints briefly described above have a dimensionality attribute, which is required to be consistent with the dimensionality of all the geometric elements involved in the constraint. The primary application areas for such constraints are

- **2D profiles or sketches**—in this case all the geometric elements, and any constraints between them, will have dimension 2.
- **Inter-feature relationships**—this covers situations where the position and/or orientation of one feature is constrained with respect to another feature. The constraints will be 3D in this case, and will usually be applied between datums or reference elments associated with the features concerned.
- **Assembly modeling**—in this case constraints between geometric elements of different part models, used to position and orient them in an assembly model, will usually have dimension 3. However, certain diagrammatic representations, e.g. of gear trains, may use 2D representations.

The structure of the ISO 10303-108 **explicit_geometric_constraint_schema** is shown in Fig. 3, using the EXPRESS-G notation. Details of the attributes of the various subtypes of **explicit_geometric_constraint** are omitted for clarity, but these were specified above in the descriptive subsections 3.3.1–3.3.27. It should be noted that the names of the dimensional subtypes as actually specified in ISO 10303-108 contain abbreviations as shown in the diagram, though these names were spelled out in full in the foregoing descriptions.

Bettig and Shah [10] have published a very comprehensive set of geometric constraint specifications based on the geometric definitions provided in ISO 10303-42 [6]. However, the approach taken in ISO 10303-108 was to provide only the most frequently used classes of geometric constraints. If others are required, the more general capabilities of ISO 10303-108 may be employed. For example, no direct means are provided for constraining the minor radius of a toroidal surface. That effect may be achieved by association of a **model_parameter** with the **minor_radius** attribute of an instance of **toroidal_surface** (as defined in ISO 10303-42) and then constraining the parameter value in whatever way is desired.

3.4 Variational Representation Schema. A *variational model* is characterized by the presence of explicitly represented variable parameters and constraints such as those described earlier. Such a model may be considered as representing a family of

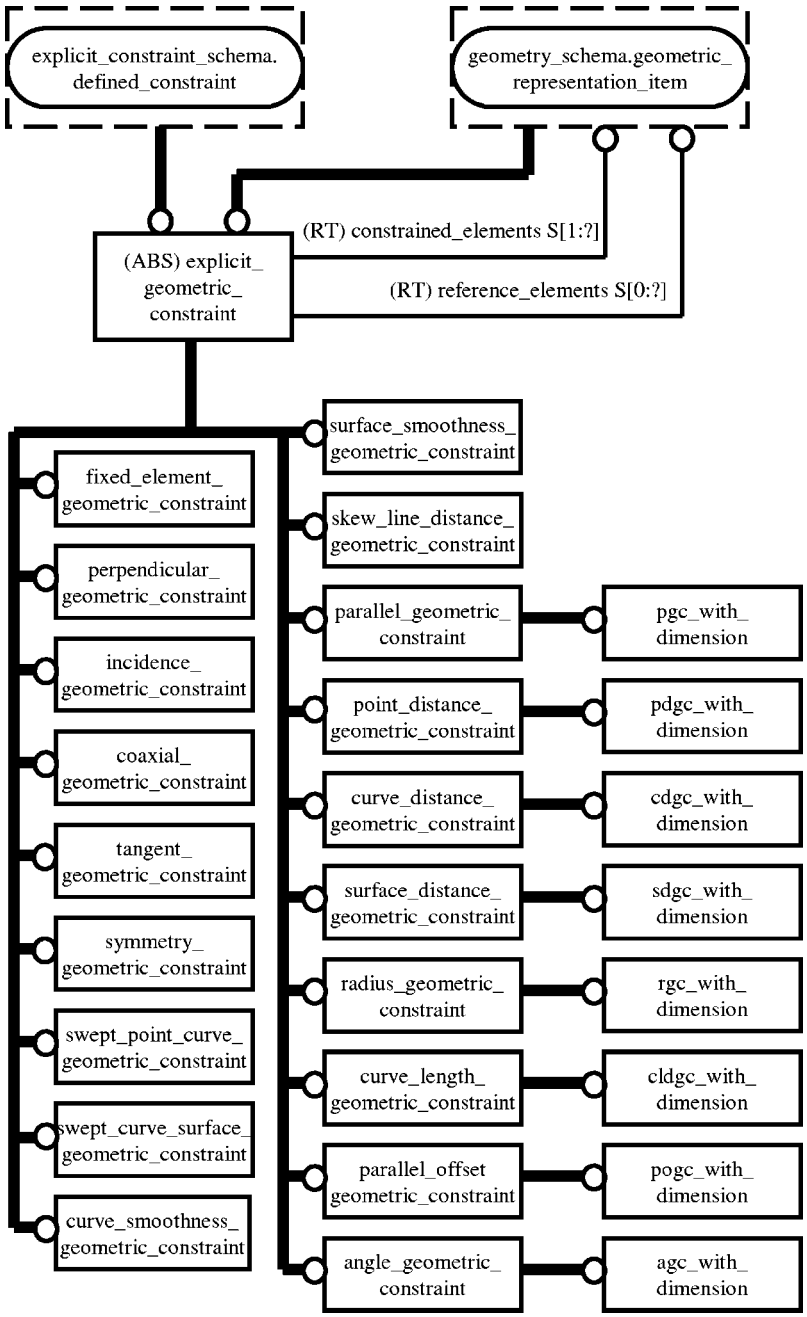

Fig. 3 ISO 10303-108 explicit_geometric_constraint_schema

related nonvariational models. ISO 10303-108 associates a ''current result'' with a variational model, namely that member of the represented family corresponding to the current values of all the parameters. Different members of the family can be derived from it by variation of the associated parameter values, subject to any imposed constraints.

The current result may be an explicit model represented in terms of its constituent elements, or alternatively it may be a procedural model defined in terms of a sequence of constructional operations (see Section 4). In the case of a procedural model the operations specified frequently require the provision of supporting elements; if these are included explicitly in the exchange file, then explicit constraints can be imposed upon them.

By way of an example, consider the Boolean union of two cylinders. The essential operations are the creation of the two cylinders and the union procedure. But the creation of each cylinder requires specification of a line (the axis) and a point (defining the position of one planar face), together with two numerical arguments defining the radius and length of the cylinder. The axial lines will normally be supplied in the exchange file as standard ISO 10303-42 instances of **line**, in which case it is possible to impose (for example) a perpendicularity constraint between them.

The word ''representation'' is used in ISO 10303 documentation as synonymous with ''model.'' An ISO 10303 **representation** has as one of its attributes a list of **representation_item** instances, and these are the top-level elements of which the representation or model is composed. These entity data types are defined in ISO 10303-43, ''Representation structures'' [5]. The top-level representation items usually reference other supporting items at a lower level, which are therefore only indirectly referenced by the **representation** itself but nevertheless constitute essential elements in its specification.

The **variational_representation_schema** defines the relationship between the current result model and its controlling variational information. The relationship between the variational representation and the current result is in fact very simple—the second is entirely contained within the first. Alternatively, the information regarding explicitly defined parameters and constraints may be considered to constitute a wrapper surrounding the current result. This schema contains definitions for only four entity data types:

- **auxiliary_geometric_representation_item**: this defines a subtype of **geometric_representation_item** that is not intended for use as an element of a shape model, but which may be used in a constraint that affects elements of that model. An example is provided by the mid-plane of a slot feature. This is not part of the geometry of the slot itself, but may participate in a constraint that positions and orients the slot in a part model.
- **variational_representation_item**: this subtype of **representation_item** is declared as the supertype of **model_parameter**, **fixed_instance_attribute_set** and **explicit_constraint**. These entities can only be used in the wrapper of a **variational_representation**. They provide the control needed for effective editing of the model following a transfer.
- **variational_representation**: this subtype of **representation** defines the overall model, comprising the current result (which is a **representation** in its own right) and the wrapper containing the variational information. The current result is not allowed to contain any instances of **variational_representation_item**. The containment relationship is modeled simply by requiring all instances of **representation_item** belonging to the current result to belong also to the **variational_representation**.
- **variational_current_representation_relationship**: this captures the nature of the relationship between the containing **variational_representation** and the embedded current result **representation**. It is a subtype of the ISO 10303-43 entity data type **representation_relationship**, and it specifies several restrictions intended to ensure that the overall representation is self-contained and internally self-consistent.

Figure 4 depicts the relationships between entities in the **variational_representation_schema**, using the EXPRESS-G notation.

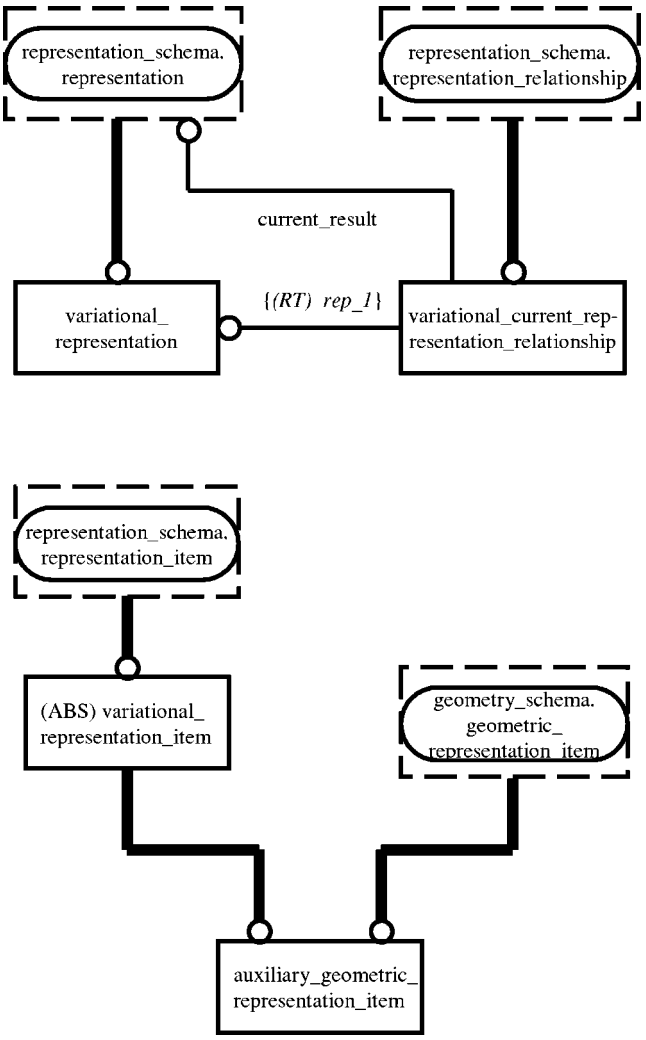

Fig. 4 ISO 10303-108 variational_representation_schema

3.5 sketch_schema. The **sketch_schema** provides representations for 2D profiles or sketches, which may be defined in several ways.

First, the schema defines an entity **neutral_sketch_representation**, a subtype of **shape_representation**. This may be thought of as a self-contained geometric model in its own right, existing in some 2D coordinate system other than model space, and possibly stored in a library. The geometry of such a sketch is explicit, consisting of a collection of 2D curve segments and points. An instance of **neutral_sketch_representation** may have a transformation applied to it to position and orient it in 3D model space, when it may be used as the basis of a sweep operation or as a cross-section for some surface construction operation. An instance of **neutral_sketch_representation** may have variational data associated with it, in which case it is also required to belong to the type **variational_representation**, with all that implies (see Section 3.4).

Second, the schema specifies the entity **positioned_sketch**. This is defined in 3D model space, and since it will normally participate in a part or assembly model it is defined as a subtype of **geometric_representation_item** rather than as a representation in its own right. Thus it does not have variational and non-variational forms, but may have parameters and constraints associated with it if it participates in an instance of **variational_representation**. An instance of **positioned_sketch** specifies explicit geometry for the sketch in terms of curve segments and points, all of which are required to lie in the plane of the sketch.

Third, the entity **repositioned_neutral_sketch** is provided. This gives a representation of a sketch in model space by reference to an instance of **neutral_sketch_representation** and a transformation. Although the geometry of the original neutral sketch is explicit, here it is only implicitly defined through the transformation provided. This entity is a subtype of **geometric_representation_item**.

Fourth, a further entity **implicit_explicit_positioned_sketch_relationship** has been defined which relates an instance of **positioned_sketch** to its corresponding instance of **repositioned_neutral_sketch**.

These four entity types cater to the following commonly arising situations:

- A sketch is initially created in some neutral 2D coordinate system and then positioned and oriented in 3D model space. If that is done, a CAD system will normally create the geometry of the 3D sketch explicitly, and this geometry may be captured as an instance of **positioned_sketch**. However, if a corresponding instance of **repositioned_neutral_sketch** is also captured, this will convey the details the transformation that was applied. The implicit and explicit representations of the 3D sketch will be related by an instance of **implicit_explicit_positioned_sketch_relationship**.

• Conversely, a sketch may be defined initially in 3D, with explicit geometry, and captured by an instance of **positioned_sketch**. Then it may be desired to transform that sketch into 2D to be stored in a library for later reuse elsewhere in the CAD model under construction. In that case also the capture of the corresponding instance of **repositioned_neutral_sketch** will occur, the transformation expressing the inverse of the the one actually applied since the mapping is in the reverse direction to that of the previous case. The relationship between the implicit and explicit forms of the 3D sketch will also be recorded, as before.

An instance of **neutral_sketch_representation** may of course be used to generate multiple positioned sketches, each differently positioned and oriented in the overall model.

The entity data type **positioned_sketch** has an attribute **sketch_basis**, which selects between three possibilities. The basis of such a sketch is either (i) a planar curve, (ii) an instance of the Part 42 entity **curve_bounded_surface** or a planar face of an existing model. These are the three types of geometric element for which sweep operations are defined in ISO 10303-42. In every case the basis is required to have dimensionality 3.

The **sketch_schema** additionally allows the association of what is called *imported geometry* with an instance of **positioned_sketch**. This is additional geometry, defined in the plane of the sketch, but not part of the sketch itself. A typical use of imported geometry is for defining the shape of an enclosure for a packaging application. The application of geometric constraints between elements of the imported geometry and elements of the sketch then allows the sketch to be optimized through the use of parametric variation while not violating the boundaries of the enclosure.

Imported geometry consists of points and curves that may be defined in several ways in terms of geometric elements lying outside the plane of the sketch. The **sketch_schema** provides the following possibilities for imported points and curves, all of them subtypes of **auxiliary_geometric_representation_item**, under the general heading of *auxiliary elements*:

- **implicit_planar_intersection_point**—a point defined by the intersection of the sketch plane with a curve in model space;
- **implicit_planar_projection_point**—a point defined by the projection of an external point onto the sketch plane in a specified direction;
- **implicit_planar_intersection_curve**—a curve defined by the intersection of the sketch plane with a surface in model space;
- **implicit_model_intersection_curve**—a (normally composite) curve defined by the intersection of the sketch plane with a surface model or boundary representation solid model;
- **implicit_projected_curve**—a curve defined by the projection of an external curve onto the sketch plane in a specified direction;
- **implicit_silhouette_curve**—a (normally composite) curve defined by the shadow or silhouette projected onto the sketch plane by a surface model or boundary representation solid model illuminated in a specified direction.

Finally, the **sketch_schema** provides a definition for **subsketch**, which as its name implies is part of a sketch. It is specified in terms of a subset of the geometry of its owning sketch. This will allow applications to define different treatments for different parts of a sketch, in specializations of this schema. Only one such treatment is specified in the schema itself; the entity **rigid_subsketch** defines part of a sketch that is required to act like a rigid body under transformation within the sketch plane, i.e., the distance and angular relationships between all pairs of elements of the subsketch are required to remain invariant under the transformation.

Figure 5 shows the primary structure of the ISO 10303-108 **sketch_schema** as modeled in EXPRESS-G. The representations of **subsketch** and the auxiliary geometric elements listed above have not been included. In two places the possibility of a selection between different attribute types has been shown by a round-ended rectangle containing the available choices, all of which are defined in ISO 10303-42 (ISO 2000a). This is not standard EXPRESS-G notation, but has been adopted here in the interests of clarity and conciseness.

4 Status of ISO 10303-108 and its Companion Resources in the Standardization Process

At the time of writing, ISO 10303-108 has passed its Draft International Standard (DIS) ballot among the 20 member countries of ISO TC184/SC4. No country voted negatively, though a total of 15 comments was received from three of those countries. Most of these were minor editorial comments, but some detail technical changes to the Part 108 **sketch_schema** were also requested in the interests of better harmonization with other related standards. The final version of the document has now been sent to ISO in Geneva for publication, which should occur before the end of 2004.

At the time of writing, a companion document, ISO 10303-55 (''Procedural and hybrid representation'') [9] has also passed its DIS ballot and been sent for publication. It is therefore appropriate, before concluding, to give a brief overview of this additional new ISO 10303 resource.

Most CAD systems use a procedural representation as their primary internal description of a model. This captures the operations used to build the model, rather than the constituent elements of the model when built. CAD systems also generate a secondary model, of the explicit boundary representation type, which is primarily used for screen display, for geometric calculations and for user interaction (e.g., for the picking of elements from the screen display by the user). What ISO 10303 currently transfers is this secondary explicit model.

The procedural model is usually a hybrid, containing some explicitly defined elements. These may include explicit 2D sketches, which are sometimes models in their own right that can be reused more than once in a CAD design. As has been explained, such sketches may have a variational nature, and one of the primary purposes of ISO 10303-108 is to capture the explicit parameter and constraint information associated with sketches that currently gets lost in an ISO 10303 transfer.

While ISO 10303-108 can represent an important part of what is known as ''design intent'' information, the capture and transfer of procedural models will provide a further major advance, because such models are inherently very easy to modify—it is simply a matter of changing the values of input arguments of constructional operations and rerunning the constructional procedure. This is, in fact, the primary reason why such representations became widely implemented in CAD systems. ISO 10303-55 is intended to provide basic mechanisms for the exchange of models of this type. It is a comparatively short document, its brevity being made possible by the use of ISO 10303 entity data type specifications as *constructors*, i.e., as invocations of internal operations in the receiving CAD system for the actual creation of model elements. This usage is permitted by ISO 10303-11 (''The EXPRESS language reference manual'') [4], though the provision of constructors was originally designed for other purposes.

The approach outlined allows the immediate specification of constructional procedures involving any elements for which EXPRESS specifications already exist in the various parts of ISO 10303. What is needed is some method for sequencing them, for dealing with the importation of explicit elements such as sketches into a constructional sequence, and for handling elements that were picked from the screen by the user in the sending system. ISO 10303-55 provides exactly these capabilities. Additional details are given in two recent papers [11,12], and a fully detailed account of ISO 10303-55 will be published when the standardization process for this new resource is complete.

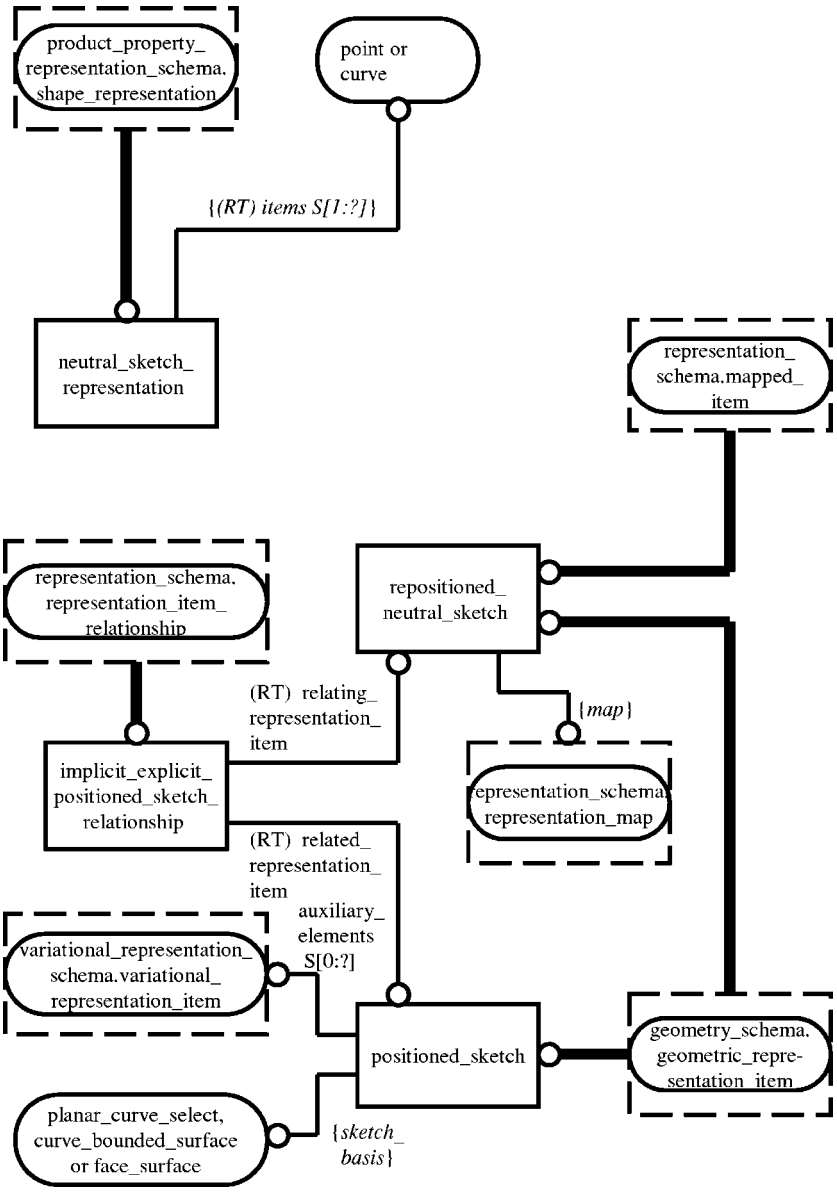

Fig. 5 ISO 10303-108 sketch_schema

Although ISO 10303-55 can be used to build models composed of existing ISO 10303 entities, there remains the problem that no EXPRESS representations are yet available for the high-level design features widely used in modern CAD practice. A project is currently under way to generate the necessary specifications. The part number ISO 10303-111 has been allocated to the design features resource, whose development is an international endeavour involving Korea, the UK and the USA. It has recently passed its Committee Draft (CD) ballot, the first stage on the road to standardization.

4.1 Experimental Verification of ISO 10303-108. Various organizations have verified the capabilities of ISO 10303-108 by writing and testing experimental translators. These include the US National Institute of Standards and Technology (NIST) and the US-based industry/government consortium PDES Inc. No fundamental problems have so far been encountered. Several reports on these experiments are known to be in preparation, and one is already available [13]. This happens to deal with the most comprehensive tests carried out, based on the use of ISO 10303-55, ISO 10303-108 and a draft version of ISO 10303-111.

The report covers a 14-month project funded by the US Office of Naval Research under an initiative concerned with improving efficiency in the supply-chain. The project was administered by PDES Inc. Six engineering companies were involved, two major aerospace companies and four of their suppliers, and these companies suggested a range of real engineering parts to be used as test models. The CAD systems used were Dassault Systems' CATIA, Parametric Technology's ProEngineer and PLM Systems' Unigraphics. The results obtained were good by previous standards of ISO 10303 tests of new translators. The most serious difficulties encountered were due to the limited scope of ISO 10303-111, which is still in the development process. This project indicated that the major groundwork has now been laid for the full transfer of parametric models using the STEP standard [14].

5 Example of the Use of ISO 10303-108 in Conjunction with ISO 10303-55

This section provides an example illustrating the use of the capabilities defined in ISO 10303-108 in modeling the shape of a product. The basic use of ISO 10303-55 is also shown, though this

resource is not described in detail—it will be the subject of a future paper, as mentioned above. The modeled shape is an L-shaped block with a hole in it. The current result model is an ISO 10303-203 (see [2]) model that precedes the procedural representation in an ISO 10303-21 [7] exchange file. Entity data types defined in other parts of ISO 10303 are used as constructors.

```
/* From current result model file*/
#840 = ADVANCED_BREP_SHAPE_REPRESENTATION(.....);
/* Now start the procedural representation:*/
#1010 = EXPLICIT_PROCEDURAL_SHAPE_REPRESENTATION_RELATIONSHIP
    (‘‘,$,#1020,#840);
#1020 = PROCEDURAL_SHAPE_REPRESENTATION(‘FINAL_OBJECT’,
    (#1280),.....);
#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((#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),( ),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 explicit current result representation of the model is summarized by the **advanced_brep_shape_representation** instance #840. Instance #1010 defines the dual model relationship between the explicit and procedural representations of the L-block.

In the file fragment instance #1030 is an occurrence of **procedural_solid_representation_sequence** as defined in ISO 10303-55. This sequence contains an ordered list of entities that are to be interpreted as constructors by the receiving system. The list specifies only the primary elements created; secondary elements involved in their definitions are, for purposes of clarity in this example, listed immediately following the operation sequence. For example, instance #1040 represents an operation for creating an instance of **extruded_face_solid** as defined in ISO 10303-42, to generate the base shape of the L-block. One of the attributes of that instance is the face to be extruded. The face concerned 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** as defined in ISO 10303-55. In this case a single element is selected, whose distinction as a selected element determines its appropriate treatment by 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.

The following instances in the above example are of entity data types defined in ISO 10303-42, whose names are mostly self-explanatory:

- ⇒ **axis1_placement** (a type of local axis system);
- ⇒ **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 ISO 10303-108 and de-

scribed earlier in this paper. 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, a reference to the solid on which the blend is to be defined, a reference to the specific edge to be blended, and the blend radius. The emerging ISO 10303 resource ISO 10303-111 will contain specifications representing this and many other high-level feature-based modelling elements.

As explained earlier, ISO 10303-108 defines the concept of a **variational_representation**, a type 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.

The following fragment of an ISO 10303-21 [7] 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®EFERENCE
    ('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 **bound_model_parameter**, in the file segments #1300–#1330, #1340–#1370, #1400–#1430 and #1440–#1470, respectively. Each of these segments defines

- ⇒ A domain of validity for values of the parameter (in all cases, a finite range of non-zero reals);
- ⇒ 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;
- ⇒ A specific attribute of an instance in the file to which the model parameter is bound, and whose value it represents;
- ⇒ 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

- ⇒ The radius of the cylinder forming the shaft of the hole;
- ⇒ The base radius of the cone forming the bottom of the hole;
- ⇒ The height of the cylinder forming the shaft of the hole;
- ⇒ The distance from the top center 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 deg and the cone base radius is equal to the cylinder radius, this sum gives the total depth of the hole. If the L-block model is enhanced with the above set of parameters and constraints, the hole in the solid may be edited simply by changing the value of the radius of its cylindrical component, in which

case other dimensions in the model will change automatically so that the resulting hole shape will remain characteristic of a drilled hole with a conical base.

The two final instances in the file fragment define an instance of **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.

All parameterization and constraint instances in this example are of entity data types defined in ISO 10303-108, with the exception of two entity data types defined in ISO 13584-20. That resource is used by ISO 10303-108 for the representation of mathematical expressions and relationships. The relevant entities are:

⇒ **comparison_equal**;

⇒ **plus_expression**.

They define relationships between mathematical expressions or elements of such expressions.

6 Summary

The paper has described the capabilities of ISO 10303-108, a new part of the international standard ISO 10303 (STEP). It allows the inclusion of parameterization and constraint information in the intersystem exchange of explicit models, particularly geometric models of the boundary representation and closely related types. It permits the transmission, with any such exchanged model, of the scheme of explicit parameterization and constraints that was imposed by its original creator, information that has hitherto been lost in ISO 10303 information transfers. The primary application areas for such parameters and constraints in CAD models are 2D sketches or profiles, interfeature relationships and interpart relationships in 3D assembly models.

ISO 10303-108 has been designed to operate compatibly with the additional new ISO 10303 resource ISO 10303-55 (''Procedural and hybrid representation''), which will provide the remaining basic mechanisms needed for the full transfer of parametric models using ISO 10303. A further resource, ISO 10303-111, will specify a set of design features corresponding to those provided by major CAD systems, and will interoperate with the two earlier mentioned documents.

Acknowledgments

Many people from the ISO 10303 development community (ISO TC184/SC4) have been involved in the work described in this paper. Not all of them can be mentioned, but major contributions have been made during the lifetime of the ISO 10303-108 project by, in particular, Bill Anderson (Advanced Technology Institute, USA), Noel Christensen (Honeywell, USA), Ray Goult (LMR Systems, UK), Tom Kramer (NIST, USA), Philip Kraushar (Boeing, USA), Akihiko Ohtaka (Nihon Unisys, Japan), Guy Pierra (LISI/ENSMA, France) and Chia-Hui Shih (Pacific STEP, USA).

The author gratefully acknowledges support from the US National Institute of Standards and Technology to enable his participation in the work described in the paper.

References

[1] Owen, J., 1997, *STEP: An Introduction*, 2nd ed., Information Geometers, Winchester, UK.

[2] Pratt, M. J., 2001, ''Introduction to ISO 10303—The STEP Standard for Product Data Exchange,'' ASME J. Comput. Inf. Syst. Eng., **1**, No. 1, pp. 102–103.

[3] ISO, 2004, ''Industrial automation systems and integration—Product data representation and exchange—Part 108: Integrated application resource: Parameterization and constraints for explicit geometric product models,'' International Standard ISO 10303-108, International Organization for Standardization, Geneva.

[4] ISO, 1994, ''Industrial automation systems and integration—Product data representation and exchange—Part 11: Description methods: The EXPRESS language reference manual,'' International Standard ISO 10303-11, International Organization for Standardization, Geneva.

[5] ISO, 2000, ''Industrial automation systems and integration—Product data representation and exchange—Part 43: Integrated generic resource: Representation structures,'' International Standard 10303-43, International Organization for Standardization, Geneva.

[6] ISO, 2000, ''Industrial automation systems and integration—Product data representation and exchange—Part 42: Integrated generic resource: Geometric and topological representation,'' International Standard 10303-42, International Organization for Standardization, Geneva.

[7] ISO, 1994, ''Industrial automation systems and integration—Product data representation and exchange—Part 21: Implementation methods: Clear text encoding of the exchange structure,'' International Standard 10303-21, International Organization for Standardization, Geneva.

[8] ISO, 2002, ''Industrial automation systems and integration—Product data representation and exchange—Part 50: Integrated generic resource: Mathematical constructs,'' International Standard 10303-50, International Organization for Standardization, Geneva.

[9] ISO, 2004, ''Industrial automation systems and integration—Product data representation and exchange—Part 55: Integrated generic resource: Procedural and hybrid representation,'' International Standard ISO 10303-55, International Organization for Standardization, Geneva.

[10] Bettig, B., and Shah, J. J., 2000, ''Derivation of a Standard set of Geometric Constraints for Parametric Modeling and Data Exchange,'' Comput.-Aided Des., **33**, No. 1, pp. 17–33.

[11] Pratt, M. J., 2001, ''Extension of the STEP Standard for Parametric CAD Models,'' ASME J. Comput. Inf. Syst. Eng., **1**, No. 3, pp. 269–275.

[12] Pratt, M. J., 2002, ''Parametric STEP for Concurrent Engineering,'' in *Advances in Concurrent Engineering* (Proc. 9th ISPE International Conference on Concurrent Engineering, Cranfield, UK, July 2002) R. Gonçalves, R. Roy, and A. Steiger-Garção eds., Balkema Publishers, Lisse, Netherlands, pp. 933–940.

[13] Stiteler, M., 2004, ''Construction History And ParametricS: Improving Affordability through Intelligent CAD Data Exchange,'' CHAPS Program Final Report, Advanced Technology Institute, 5300 International Boulevard, North Charleston, SC 29418, USA.

[14] Pratt, M. J., Anderson, W. D., and Ranger, T., 2004, ''Towards the Standardized Exchange of Parameterized Feature-based CAD Models,'' in press, Computer Aided Design.