A Design/Constraint Model to Capture Design Intent

Chia-Hui Shih, Ph.D.
Pacific STEP
cshih@pacbell.net

Bill Anderson, Ph.D.
SCRA
Advanced Technology Group
anderson@scra.org

1. INTRODUCTION

1.1 Background

ENGEN (**E**nabling **N**ext **GEN**eration Mechanical Design) is a program jointly sponsored by **DARPA** (Defense Advanced Research Projects Agency) and PDES, Inc., an industry consortium to accelerate the development and implementation of ISO 10303, known informally as **STEP** (**ST**andard for **E**xchange of **P**roduct Model Data). This research program is an 24 month effort (Sept. 95 - Aug. 97) to prove the feasibility of capturing certain key aspect of design intent in a neutral data model as represented by constraints, parametrics, design history, and features. The model will utilize the EXPRESS modeling language and be independent of any particular CAD system. In 1997 a demonstration is scheduled to show exchange of design intent among dissimilar CAD systems. The principal program deliverables are the ENGEN Data Model (EDM), and a pilot demonstration showing exchange of design intent using Ford's work flow process with the systems: Computervision (CV) - CADDS 5®, Structural Dynamics Research Corporation (SDRC) - I-DEAS®, and Parametric Technology Corporation (PTC) - Pro/E®. Organizations and companies participating in ENGEN are Ford Motor Company, SCRA, PDES, Inc., CV, SDRC, PTC, International TechneGroup, Inc. (ITI), Pacific STEP, Purdue University, and Arizona State University.

The main focus of this paper is to discuss the use and representations of constraints within the framework of STEP methodology, in particular, the conceptual modeling language EXPRESS and the STEP physical file. A parametric module is built on top of the design and constraint modules. Current parametric/variational/feature-based CAD systems have the powerful capability to associate parameters with a design, so that a change in a parameter(s) results in an automatic update to the design. For example, the diameter of a hole in a block may be ½ the length of the block, so that a change in the length explicitly changes the diameter. Some of the parameters provide insight into the intentions of the designer.

The initial release of STEP in 1994 successfully handles the exchange of 'static' design, providing no mechanism to capture design intent. This is a barrier to the exchange of more intelligent product information between mechanical design applications. Thus, commercial STEP translators have focused primarily on lower level geometric and topological information for solids, surfaces, and curves. The ISO Parametrics Group has begun to develop a model to address the area of parametrics [1]. EDM [3] is a major component in the STEP effort to develop parametrics capability ([2]).

In our context, the design, rather than just being a passive drawing or a flat file, becomes more active in the sense that it enables the downstream users to modify or regenerate the design from explicit information in the exchange file. The information provided in the exchange file are generated by the sender or upstream designer. Moreover, additional requirements at later stages can feedback to the earlier designers to refine the design. Constraints in predefined or free form formulations capture the logical inter-relationship as well as the inter-dependence of design/shape data. The design intent is an accumulation of these interactions of design/shape and constraints. A file or database containing design intent information is more automatically "editable" than one which contains nothing but a snap shot of a design. Because EDM is a conceptual model that captures design plus constraints, which are the main aspect of parametric technology, we can view ENGEN as an application of the parametric technology with a special objective, namely, to convey design intent.

Solid Modeling '97 Atlanta GA USA

1.2 Development platform

The EDM approach uses STEP as its development environment/platform. STEP is developed as a group of 'parts' [8], each of which represents an International .Standard. The initial release of STEP in 1994 contained 12 parts. We utilize the EXPRESS [4] modeling language (ISO 10303-11 or Part 11), and a data/information exchange mechanism known as the physical file exchange format (ISO 10303-21 or Part 21). Unfortunately, once a language is adopted, its style and grammar take away some freedom in solution; hence, EDM is subject to the power as well as the limitation inherited from EXPRESS. Furthermore, as the EDM model is instantiated in a Part 21file (for brevity, STEP file), additional limitations are present due to the mapping rules between the EXPRESS grammar and the Part 21 grammar. Next is an example of the EXPRESS definition of a widget entity followed by an instance in a STEP file.

```
ENTITY widget;
  a1    : INTEGER;                -- A
  a2    : STRING;                 -- B
  a3    : BOOLEAN;                -- C (F, T)
  a4    : LOGICAL;                -- D (F, U, T)
  a5    : SET [1:2] OF REAL;      -- E
  a6    : REAL;                   -- F
  a7    : point;                  -- G (An entity type )
END_E NTITY;
```

Note: F = False, U = Unknown, T = True.

```
#1 =   POINT (..);
identifier   entity name
#2 =  WIDGET  (99, 'ABC',  .T.,  .UNKNOWN.,
                    |     |     |        |
                    A     B     C        D
(9.000, 1.2345),  -7.8, #1 );
        |      |     |
        E      F     G
```

Thus, the tight coupling of EXPRESS and STEP file on the one hand offers efficient mapping and ease for implementation; yet on the other hand causes restrictions on expressions for both modeling and instantiation.

1. 3 Terminology

In order to clarify the usage of several ordinary terms in this discourse the following definitions are used:

Constraints: limitations on an item's permissible values within itself or with respect to other elements in the design, such assize, shape, position, orientation, dimension, logical relationships (such as identical, distinct, set relation), as well as engineering constraints.

Design History: a record of the design activities or events in the creation of the design.

Entity type: A major construct of EXPRESS that represents a conceptual unit that is used in depicting a design. Examples of entity types are line, transformation, and B-spline curve. Entity types may have hierarchical relationships among each other, that is, one entity type may be a subtype of one or more other entity types (called supertypes) by inheriting the supertypes' attributes. One entity type may also contain attributes (as part of its representation) of entity types. Every entity type is associated naturally with a construct function.

Model: a collection of logically related entity types and rules that the application designer can use as low level templates, (e.g., line as a pair of direction and point) in order to create an intermediate or final product. A model is a term at the conceptual level; its analogy in STEP is a SCHEMA.

Module: a subset of a model containing types that are grouped together based on some commonality such as common characteristics or functionality. Structurally, there is no distinction between a 'model' and a 'module', the difference lies in their scopes and relative domains. We use 'module' when the collection is intended to support a part rather than the whole for the discourse of interest, where 'model' is to stand for the universe of an application. For example, in ISO 10303-203 [5] advanced boundary representation solids may be considered a module. Thus, module is a relative term.

Parametrics: the use of equations and constraints to augment (or replace) the explicit creation of product shape on CAD systems.

Entity instance: An instance of an entity type appearing as a data record in a STEP file or a database. A STEP pre-processor outputs a STEP file and takes as input either a CAD system database or possibly its command log. This is because not all CAD systems store the constraints in their databases, but only use them to compute the solutions.

1.4 High level architecture

In this paper, we describe one major model (EDM) which contains three modules: Design[1], Constraint, Parametric Module. The Design module is based on ISO 10303-42. The short term approach requires no change of the existing STEP schema(s) whereas the intermediate term approach will require some extensions on schema(s). The Parametric Module supports both parametric and variational technology in declarative representations. Another module in EDM, called Spatial Configuration, deals with assembly of parts appearing as constrained shapes in a common coordinate system will not be discussed in detail here. The Feature Module is under development and currently contains a small number of transition features ([3]). The Design module is self-contained; Constraint module points to Design; Parametric points to Design and Constraint. The architecture is depicted in the following figure.

[1] Design contains shape of the part without any external constraints. Shape and design will be used interchangeably.

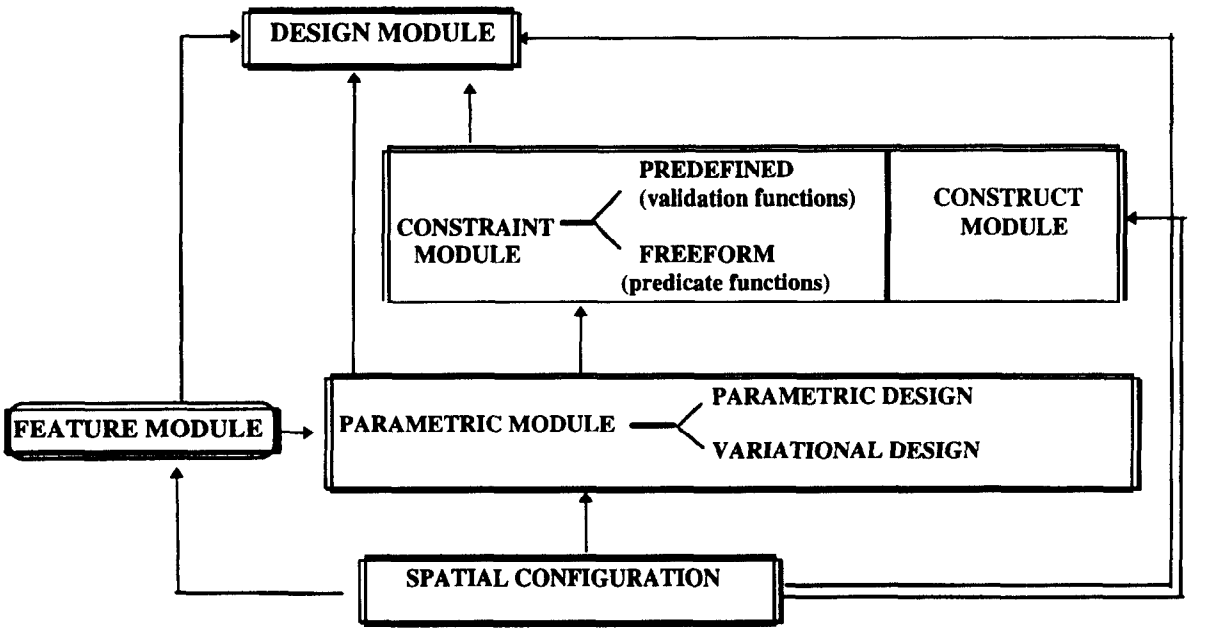

Figure 1: EDM high level architecture

The purpose is to have an architecture whose base is built on the traditional design models such as geometry, topology, or configuration control by adding new modules in an upward compatible manner. In order to make any extension upward compatible, we must have the new modules pointing to the existing ones, but not vice versa. In other words, the existing one is self sufficient without knowing any new modules referencing it. In the following sections, we give a brief introduction of each module in EDM.

2. INTERACTION OF DESIGN AND CONSTRAINT

2.1 Nomenclature

For the sake of this discussion, a specification stands for a requirement at the higher conceptual design, while a constraint stands for a requirement or a condition at the detailed design level, using one or more mathematical equations. A specification is usually broken into many low level constraints through a sequence of design activities. Thus, for a design to satisfy even a single specification it may require the validation of many mathematical equations, engineering equations, or logical conditions at different phases of the design cycle. For example, suppose there is an automobile design specification to have a seat belt that can sustain a certain impact force. Meeting that specification involves many low level detailed design requirements in shape, assembly, and material. For each aspect, there may be many equations to compute, solve, and verify. For our purpose, the parametric modeling paradigm does not deal with high level specifications, but rather deals with low level constraints most represented by mathematical formulations.

A realized design is a collection of related instances of standard entity types that together yield the desired outcome at that design stage. When equipped with sufficient constraining information, a 'skeleton' of such a design can be extracted by leaving out some or all attributes values. Hence, a potential design rather than a realized design can be preserved and shared. Such a skeleton has true parametric characteristics. For example, a cube can be decomposed into a collection of six planar faces, twelve straight edges and eight vertex points in such a way that parallelism and dimensional constraints are imposed on the faces and edges. Such a cubic skeleton may have only one parameter, e.g., the dimension of an edge, to be realized. By using a parametric model along with an assembly mechanism (spatial configuration) we can place it in a larger design. In the ENGEN program, we do not handle such a macro or template entity, because of the limitation imposed by the physical file that all attribute values must be given. In other words, any complex template/macro must be declared as entity types in the schema. In a sense, a macro could be a user defined (likely complex) entity type; which cannot be handled effectively by the first release of STEP.

2.2. Effects of constraints on design

Constraints can be viewed in three aspects:

1. As restrictions and limitations on parameters that define the design. This is the intuitive understanding.
2. As freedom for modification of the design. This sounds like a total contradiction. First of all, by 'freedom', we do not mean tolerance in dimensioning or precision for real numbers. When

a design without any constraint information is passed down the development cycle, it represents a fixed snap shot yielding no freedom to the receiver to alter. Unless the constraint equation or equations yield a unique result, in that case, the intention is clear, the possibility of more than one acceptable solution indicates a freedom to modify. For example, two lines are passed along with the constraint that they are parallel. A broad interpretation is that both lines can move and the distance between them can change so far as they remain parallel. A single constraint is generally not deterministic; but a group of them can be.

3. As a design generating tool subject to #1 and #2. We assume that the receiving system has a 'solver' which uses the information as input and generates a result satisfying given constraints.

In particular, parametric modeling deals with constraints that are expressible in terms of basic data types such as integers, real numbers or strings, and entity types, such as line or ellipse. We would prefer to think of a constraint as a mathematical or engineering condition expressible in a system of equations F(p1, p2,...,pn) = 0. The result of each developing stage feeds into the next phase as parameters (with value already assigned) and additional conditions may be imposed. The iterative process may require the change of a design or even some constraints; hence it is very important to identify the elements that control the design outcome and the constraints they are subject to. This facilitates the modification of a design with limited intervention while enabling the preservation of design intent. In the simplest case, a few attribute values can be changed as long as the constraints are not violated; whereas in the extreme case, a new design may need to be re-generated by the downstream solver based on the parametric conditions that are presented in the exchange file. The EDM conceptual model itself does not preclude over-constrained or under-constrained design. The protocol of these various situations must be understood before any data sharing/exchange takes place.

3. ENGEN DATA MODEL (EDM)

The EDM modules introduced earlier are now discussed.

3.1 Design Module

The Design module consists of data structures and functions for defining shape information, primarily from STEP Part 42. The scopes of many supertypes are greatly reduced to include only a few needed subtypes. Geometric entities included are points, vectors, lines, conics (circle and ellipse only), and swept area solids (extrusion and revolution) plus Boolean operations for combining solids. The test parts for ENGEN are a Ford connecting rod and crankshaft, which will not require free form curves (B-splines in STEP). They are expected to be required in a future version of EDM for modeling parts such as automobile manifolds.

There is also one basic revision of these SCHEMAs summarized as follows:

Each attribute in an EXPRESS ENTITY of a basic data type such as REAL, INTEGER, or STRING is mapped into a STEP file as a constant with no identifier. This EXPRESS to STEP file mapping inhibits treating such attribute as a parameter because they are not assigned with identifiers. In order to handle attributes as parameters rather than static data, we introduce an entity type for REAL, INTEGER, or STRING as well as for each user defined data types of these types. These identifiable constants can then be used in the place of constants in a traditional STEP file. For example, instead of assigning 5.0 as the radius for a circle, alternatively we may assign R to the radius and assign 5.0 to R, where R is a persistent name. In this manner, we can assign constraint(s) on R, e.g. , $4 < R < 5$, so that it becomes much more versatile to modify the design.

3.2 Constraint Module

It is very important to point out the difference of the concept of rule in EXPRESS and the constraints of EDM. EXPRESS contains two types of rules: local and global. A local rule (key word: WHERE) pertains to a given entity type and applies to all instances of that entity type; for example: the radius of a circle shall be greater than one. A global rule (keyword: RULE) involves one or multiple types and applies to all instances of those involved types in a given instantiation of the schema; for example, the Euler's equation of topological elements for a manifold. However, there is no need to exchange these at the instance level, because they are part of the definitions. On the contrary, the constraints of interest in this paper are user created requirements. It contains constraints associated with the entity types in the Design module and they are contingent rules against the instances in the design process, not against the conceptual model; for example, 'the distance of the center of a particular circle must be at least 1 mm from the closest boundary curve', or 'Stress(A1) + Stress(A2) < 105k psi, where A1 and A2 are cross sections of a given part model'. These rules are not part of the conceptual model.

A pre-defined or explicit constraint may be applied in a shape model composed of geometrical and/or topological elements. It usually asserts a relationship between two or more elements present in the model, and is also usually external in the sense that it does not affect the shape or other static properties of the representation. The purpose of such a constraint is to capture a modest but important aspect of the designer's intent in creating the model; that is, the requirement that certain characteristics of its shape should remain invariant if any design modifications are subsequently made. Thus, it is only after transfer of the model into a receiving system that explicit geometric constraints have any effect. They then restrict the design freedom available for modification, the intention being that such restrictions ensure the continued functionality of the design throughout any changes that are made.

A shape model may contain two parallel linear edges. The parallelism may be expressed by the fact that the lines on which the edges lie share the same direction. This is part of the representation of the model (as it currently exists). We may say that the lines sharing the same direction instance are constrained by 'internal' or 'implicit' constraint. However,

ambiguity in interpretation between the sender and the receiver may exist. Is the usage of a shared direction instance an indication of an intentional constraint or is it simply for reducing the file size when the two edges are 'incidentally' parallel? Now suppose that a parallelism constraint is applied to the same pair of edges. This causes no changes to the geometry or topology of the shape model, as stated above. It asserts something that is already true, but additionally it expresses the requirement that the condition should remain true after reconstruction in a receiving system where further modifications may be made to the model. The external or explicit constraint is therefore not part of the shape description of the model as transferred, but governs what changes may subsequently be made to it.

Explicit geometric constraints are not necessarily used in the reconstruction of the model in the receiving system --- this will often be possible in terms of geometry and topology alone. However, it is envisaged that the constraint management capability of the receiving system will be used to check the model for compliance with the explicit geometric constraints transferred with it.

Some constraints have predefined connotations and are easily put in template forms. A constraint is either a condition(s) on a given design entity, or to establish a relationship among multiple design entities. They can be one-to-one, many-to-one, one-to-many, or many-to-many.

There are two ways to convey such a relationship. One way is to define pure design/shape elements (as in PART 42) and pure conceptual constraints (e.g., the concept of parallelism) separately and define an 'action' which chooses the constraint element to act on the certain design elements. The other way is to make the constraint elements as built-in characteristics of the design elements so that an instantiation of such a (relation) type conveys the whole action. An example of the first approach is: given a line and two points, the constraint is for the points to lie on the line or the line to pass through the points. An example of the second approach is to create a line passing through two points. EDM models the design-constraint relationship in both ways. The first way is depicted by the composition of Design Module and Constraint Module, and the second by Construct Module.

In each constraint type we classify the design elements in two classes: target and referent. The implication is that in a given constraints, the parameters do not necessarily play symmetric roles in the design process, but have a dependency relationship. The independent ones are included in the referent and the dependent ones in the targets. The purpose is to simulate a command during the design process as well as to convey the logical asymmetric relationship, if any.

A partial design history can be extract from the relationship by interpreting that the target instance is created after the referent element. This history is only abstract, because it is derived from the exchange file. The editability of the data file depends on the classification of the target and the referent set. EDM's Parametric Module contains an entity called 'design history' which explicitly captures the design history by listing the sequence of events. An event is a triad of (design instances, constraints, design instances). In short, the triad represents one design step seen by the user. It could be as atomic as a single design-constraint operation, or as molecule as a collection of them. Within the collection, the design-constraint instances can be sequential or concurrent.

There are basically three classes of constraints from the aspect of representation:

1. **Predefined** : Contains constraints that have well defined semantics with well known parameters (variables) in established formulations. Every entity type is basically a template. For example: parallel (for lines), axial symmetry (for curves or points), point on curve.

2. **Free Form** : Contains constraints that do not have a pre-conceived semantics and frame work. For example: The sum of the length and width of a rectangle = 10.0;

3. **Construct** : Contains built-in constraints in the process of creating the design element. For example, in STEP a standard line is represented by a point and a vector. Suppose we wish to impose the condition for a line to pass through another point. Instead of using a predefined constraint: 'curve passing through a point' to the line instance, in this module we have an entity type "construct-line-through-two-points" which is a subtype of the line type. In other words, this entity type inherits the standard attributes ('start' and 'dir') and has an additional attribute second_point. Many standard construction methods can be included as entity types in this module.

In EDM most constraints that are predefined can be characterized as geometric constraints with the exception of logical constraints, and the free form as algebraic constraint; but that distinction is, though intuitive, not precise. The analogy of "predefined vs. free-form" constraints in the area of design is "standard vs. user-defined" entity types. The latter has no pre-established notion or formulation, and is represented by means of composing basic, generic mathematical or other type of symbols. For predefined constraints, we will use the EXPRESS construct ENTITY to capture the parameters needed; even though conceptually the EXPRESS reserved word RULE is a more appropriate keyword for constraint type elements. This is because at present, all RULEs are observed at the conceptual level, and there is neither need nor mechanism to instantiate RULEs in STEP files. For the same reason, there is no direct and clean way to instantiate free form constraints, because of the lack of support in STEP files for anything but ENTITY. We shall handle them by means of a special type of STRING.

3.2.1 Predefined Constraint

Most predefined constraints in EDM are geometric in nature with a few logical/algebraic in nature. Figure 2 shows the pre-defined constraints included in EDM.

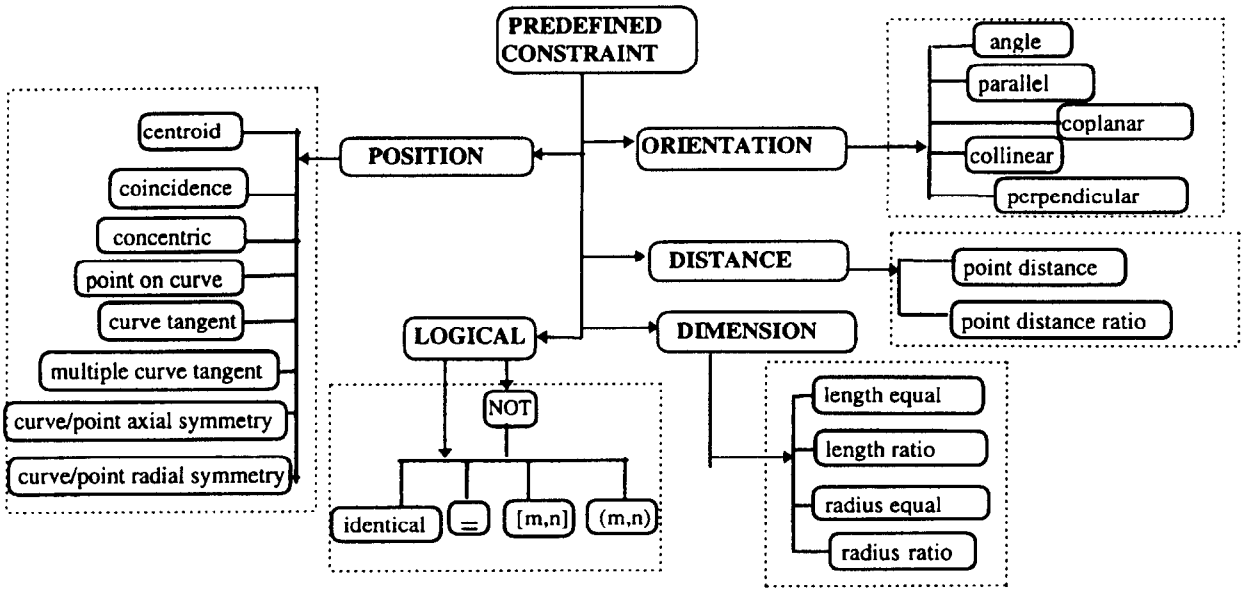

Figure 2: Hierarchy of Constraint Module

Every predefined constraint type contains also one or two attributes indicating the linear accuracy, the angular accuracy, or both. In addition, every constraint is associated with a constraint validation function which returns TRUE if the computation results in numbers that are less then the prescribed accuracy values.

3.2.2 Freeform Constraints

Freeform constraints are in the form of equations (expression). There are at least two approaches to represent an expression. One of the approaches is to define the basic ingredients that play roles in a general equation, such as variable, constants, operations, mathematical or numerical relationships, etc. as entity types, then define an expression also as an entity type that contains adequate attributes which facilitate the construction of an expression using these basic ingredients [6]. The advantage of this approach is that we can accomplish the goal by strictly remaining in the scope of STEP. These could be done because the construct ENTITY TYPE of EXPRESS lacks any pre-existing semantics so that very general concepts can be declared as an entity type. Thus, to define an expression, though tedious and complex, is safely within the power of EXPRESS. The disadvantage is that it is tedious and complex and non-intuitive. The exchange file has to re-construct a simple expression such as a + b = 5 by putting together the 'jigsaw puzzle' containing 'a', 'b', '+', '=', and '5' together. For example, let us assume that 'real_variable', 'real_constant', 'add', and 'equal' are entity types with appropriate attributes, five records are needed to create a single expression for a + b = 5 as follows:

```
#1 = REAL_VARIABLE ('A');
#2 = REAL_VARIABLE ('B');
#3 = REAL_CONSTANT (5);
#4 = ADD( #1,#2);
#5 = EQUAL(#4,#3);
```

The postprocessor would have to put them together into an expression which is recognizable by the CAD system; but the effort may be justified by not requiring an extension to EXPRESS and the existing STEP standards.

Another approach is to use STRING. This needs one additional assistance beyond the common usage of STRING. STRING allows us the freedom to put any collections of characters inside a string, even foreign characters, with a few precautions. The content in a STRING is for human readability. In order for a computer to read the value of a string, the string must have a grammar that can be processed by a computer. A freeform constraint in EDM is represented by a STRING that obeys the EXPRESS grammar on expressions. An 'EXPRESS based STRING' is signaled by an escape symbol '\e'. A user defined grammar can be used in a STRING with an escape symbol '\u'. The latter situation is based on the private agreement of the data sharing parties.

Finally, in order to facilitate expressions, besides all the available functions in EXPRESS, we included in EDM some functions which we call 'predicate functions'. For example, distance (point, point): REAL; These functions can be used in the place of an identifier in a legitimate expression. This approach would require an extension to the EXPRESS production rule. For example, for the equation above, we will have the following record:

```
#1000 = CONSTRAINT_FREEFORM (target_id,
referent_id,'<esc> A+B = 5','comment');
```

3.2.3 Construct Module

This module contains entity types that depict construction procedures. In the purest sense of a standard, a schema of STEP should contain a unique representation for each entity type. For example, if 'point' and 'vector' are chosen to represent a line, then two points should not be used to represent a line, no matter how common and useful it is.

However, for practical reason, there exists a handful of SUBTYPEs in the STEP parts that actually indicate enhanced representations for some entities. The EXPRESS SUPERTYPE/SUBTYPE feature provides extended entity representation in a controlled manner. We note that a SUBTYPE always inherits the attributes of its SUPERTYPE, hence it retains its 'standard representation'. Every entity type defined in the Construct Module is a SUBTYPE of some standard entity type. The additional attributes in the construct type is to serve the purpose of 'parameters' as they are used to solve the solution. Naturally, a construct entity type is over-constrained in that all the attribute values are not independent. The extra attributes of the construct type are different from the derived attribute of EXPRESS in that a derived attribute is a computational result from the standard attributes, while for a construct entity type, the standard attribute values are fully or partially derived from the given attributes in the construct entity type. For example, a line-through-two-point instance contains a start point, a vector, and another point. The direction of the line is computed from the points, but the magnitude of the vector is an independent information.

The Construct Module contains sufficient declarations of such nature as to provide additional methods in creating and modifying standard entity types. There are two reasons to choose ENTITY over other EXPRESS keywords such as PROCEDURE or FUNCTION, even though the nature of the latter is more compatible with the concept of constructor.

A. There is no mapping of FUNCTION to the physical file. In order to work within the existing STEP framework, ENTITY is the only available EXPRESS construct.

B. For EXPRESS FUNCTION, we need to include the actual algorithm for the function, not just the signature or prototype of the function. Since our purpose is NOT to tell HOW the constraints are solved, but WHAT they are, we use ENTITY to declare the functional argument. The attributes of an ENTITY are basically the arguments of a function.

In the future when the representation capability of STEP file is extended, more sophisticated constraints in the form of procedure can be exchanged by means of user defined procedures or functions. That will help the communication among dissimilar systems. At the present, since only 'WHAT' is represented, we have to leave alone the issue concerning solver incompatibility. There are several points worth noting concerning the Construct Module.

1. Every construct entity is a subtype of the original type. It is open-ended how many construct entity can be declared.

2. In each construct entity type, there is an attribute called "fixed" which indicates which attribute values, if any, are treated as constants (i.e., fixed constraints not subject to change).

3. The attributes in a construct entity may not yield a unique result, hence it is not exactly the same as the 'constructor' in programming language such as C++. We note that a construct entity is a subtype of the original type, hence it inherits the set of attributes that uniquely define an instance. The intent of a construct entity is to make the constraints a 'built-in' feature of the entity.

The following table shows the current content of the Construct Module of EDM.

ENTITY TYPE	METHOD
point	intersection, bisect, tangent, centroid, displaced, extruded
vector	two points, cross product, sum, difference, scalar multiplication, tangent vector, extruded
direction	two points, cross product, sum, difference, tangent
line	two points, parallel through a point, perpendicular, angled, tangent, extruded
circle	three points, concentric, tangent to line and through a point
plane	three points, parallel through point

Figure 3: Construct Entities

3.3 Parametric Module

Parametric Module contains mainly collections of individual design and constraint elements. EDM uses it to support parametric technology in which the input and output parameters are design data and the constraints behave like 'operations' on the parameters. We can imagine the parametric model as a vending machine to which one or multiple 'parameters' are dropped into the slots, either order sensitive or order insensitive, and some 'product design', either singular or complex, comes out of the output spout.

Parametric Module contains entity types that represent the various 'ways' the user inputs the 'parameters' and chooses the operations. Hence, it uses the individual design types and constraints defined in the Design Module (they are like the token or coins we drop into the machine) and Constraint Module (they are like the different choices) as its basic structure. The Parametric Module and its relations with other modules can be depicted by the following diagram:

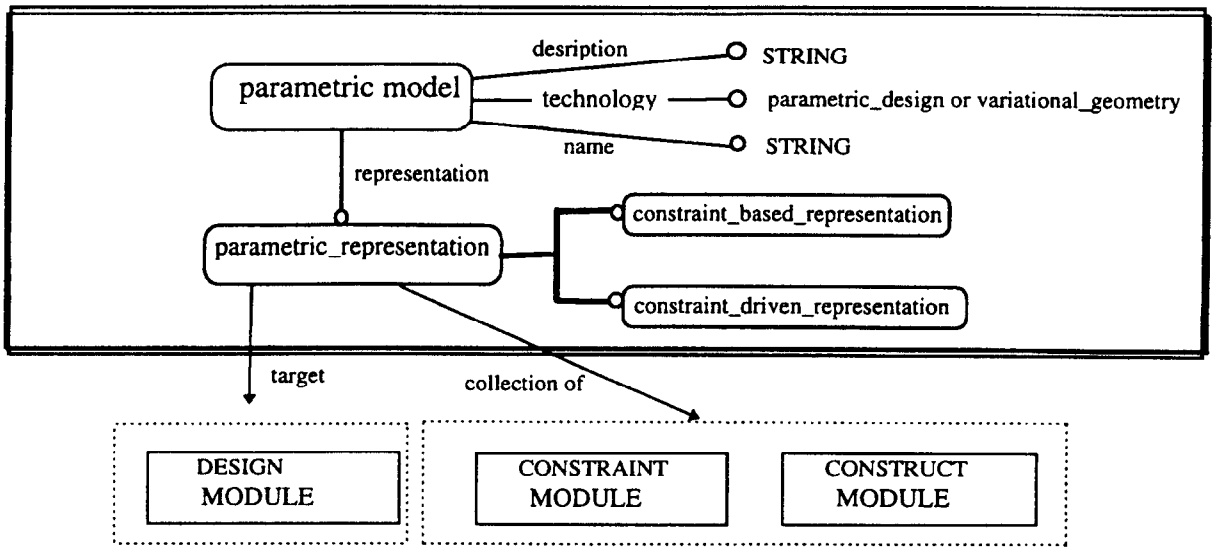

Figure 4: Architecture of Parametric Module

Based on Solver characteristics:

1. **Parametric Design** : Stands for the type of solver which requires an explicit order for the constraints/conditions to compute the solution stepwise. Such technology can be viewed as showing HOW. The process is represented by a list (i.e. ordered) of (design, constrain) pair or a list of construct entity types. Recall that a construct entity type is an entity type with built-in constraints.

2. **Variational Geometry** : Stands for solver which does not require orders for the constraints/conditions, and it is irrelevant on how to solve them. Such technology can be viewed as showing WHAT.

Further discussion can be in found in [9].

Based on representations:

1. **Constraint Based design** : A constraint based parametric model is basically a collection of either unordered (concurrent constraints) or ordered (sequential constraints) constraints.

2. **Constraint Driven Design** : The constraints take a more active role so that the design is computed out of the constraints from solvers. It is a collection of either ordered or unordered generators. A generator is a triad of (input, operator, output), where input and output are design elements and operator is a choice of constraint as defined in the Constraint module or the Construct module.

4. TECHNOLOGY APPLICATION

4.1 Demonstration

A work-in-process demonstration was conducted for the PDES, Inc. Technical Advisory Committee which showed exchange of some geometric constraints (parallel, perpendicular, point on, equal length) for a connecting rod test part. Suppliers participating were SDRC (IDEAS), CV (CADDS 5), and PTC (Pro/E). Figure 5 in the following provides the scenario of the demonstration. The connecting rod had been designed on CADDS 5 with the geometric constraints captured. The CV pre-processor generated an ENGEN Part 21 (STEP) file containing the constraints. PTC ran its post-processor on the file and brought the connecting rod into Pro/E with the geometric constraints displayed. The connecting rod had also been created in SDRC, and the pre-processor output the ENGEN file with the geometric constraints. The post-processor to transfer the ENGEN file into CADDS 5 was in-work as indicated, but has subsequently been completed. The 'Mesh' arrow shows a direct transfer of a finite element mesh from Pro/E to IDEAS. A future extension of the ENGEN technology might incorporate this capability.

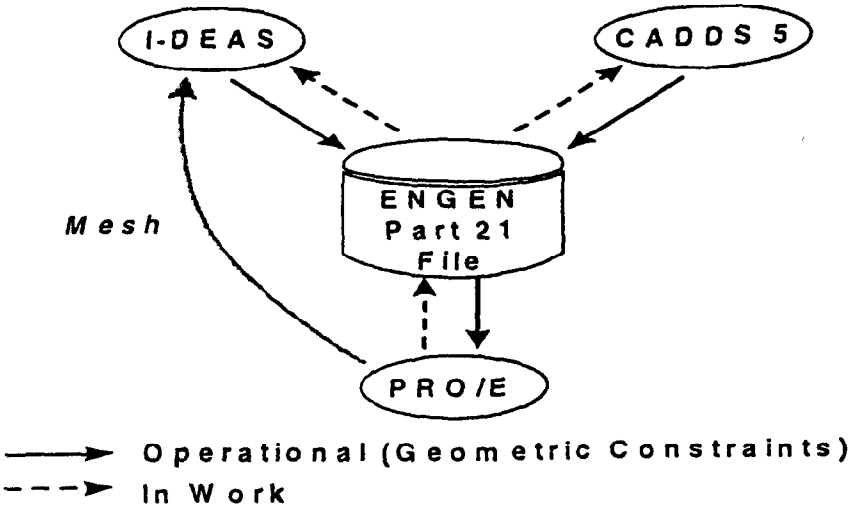

Figure 5: Work-in-Process Demonstration

This demonstration generated additional interest from PDES, Inc. member companies All three CAD vendors are working on input and output translators for geometric constraints in EDM. Figure 6 shows some of the geometric constraints handled by the CAD vendors. The constraints are on the profile and a solid model of the connecting rod is obtained by extruding the profile.

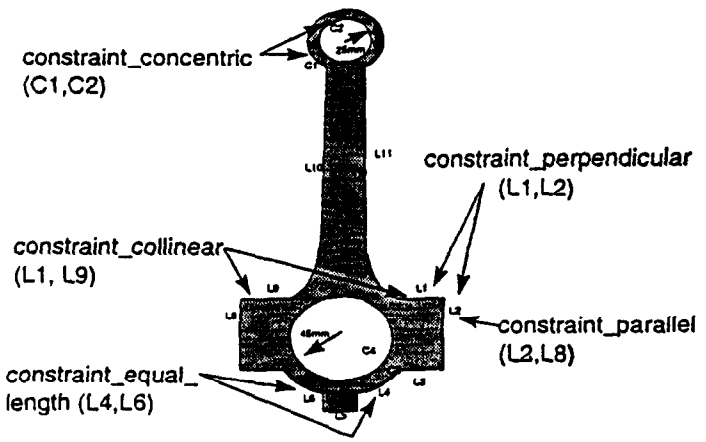

Figure 6: Geometric Constraints

5. Implementation difficulties and summary

The EDM has developed a foundation for capturing some key aspects of design intent and will be extended to include free form curves and some transition features. Vendors in the ENGEN program are developing pre- and post-processors to exchange design and constraints. The main obstacles so far encountered were:

1. Achieving agreement on which constraints in the EDM that the vendor systems can exchange; one might use an explicit constraint (line 1 parallel to line 2) or an implicit constraints (line 1 and line 2 use same direction vector); explicit constraints were decided on in the exchange format since they more accurately convey design intent.

2. CAD system developers differ in their opinion as to what constitutes design intent. Some use explicit constraints where others use implicit constraints (as mentioned above) or topological relationships. Thus some systems generate many

topological relationships. Thus some systems generate many more constraints in the exchange file, and this has led to problems of compatibility of interpretation.

3. The same effect can be achieved through the use of different but equivalent constraint sets. This should be no worse a problem than the use of different but equivalent shape representations in solid modeling, but it will take time to perfect the necessary translations. The actual choice of constraints transmitted in the neutral file should be transparent to the user.

4. Understanding the relationship between the modules, with primary focus on DESIGN and CONSTRAINT Modules so far, has been a challenge for the team, and particularly for vendors writing software; the model has been updated several times based on feedback from the implementers.

5. Maintaining an aggressive schedule with technical resources distributed throughout the U.S. and Italy; language differences of technical team whose native languages are: Chinese, Russian, Italian, and `Texas English' provide an interesting challenge for communications.

A Pilot Demonstration in mid 1997 at Ford's Alpha Manufacturing facility will illustrate program vendors' support for EDM in capturing and exchanging key aspects of design intent.

Acknowledgements : We gratefully acknowledge the support of DARPA and PDES, Inc. for this research . We also wish to thank members of the ENGEN team, and in particular, Dr. Mike Pratt, Dr. Jami Shah, Dr. Chris Hoffman, and Mr. Noel Christensen for their interest, review and valuable comments.

References

1. Pratt, M. J., `The STEP Standard, and its Extension to cover Parametric, Constraint-based and Feature-based Modelling', in Advances in Computer Aided Design (H. P. Santos, ed.), Proc. CADEX '96 Conference, Hagenberg, Austria, Sept. 1996; IEEE Computer Society Press (1996).

2. Christensen, Noel, 1996, Parametrics Framework Proposal (Working Draft), ISO Parametrics Committee, pages 1-20 (on the WWW at http:/www.nist.gov/sc4/paramet/short/framewk/dec96).

3. Shih, Chia-Hui, 1996, ENGEN Data Model Version 4.2 (on the WWW at http:/www.nist.gov/sc4/paramet/short/engen/edm42.ps or .doc)

4. International Organisation for Standardisation (ISO), International Standard ISO 10303-11, Product Data Representation and Exchange, Part 11 - EXPRESS Language Reference Manual, 1994.

5. International Organisation for Standardisation (ISO), International Standard ISO 10303-203, Product Data Representation and Exchange, Part 203- Application Protocol: Configuration Controlled Design, 1994.

6. Pierra, G., Ait-Ameur, Y., Besnard, F., Girard, P. and Potier, J.-C., A General Framework for Parametric Product Models within STEP and Parts Library, in Proc. PDTAG Product Data Technology Days '96, `BusinessBenefits from PDT', Shell Centre, London, England, 18 -- 19 April 1996.

7. Hoffmann, Christoph, and Juan, Robert, 1993, Erep- An editable, high-level representation for geometric design and analysis, In *Geometric Modeling for Product Realization* (eds. P.R. Wilson, M. J. Wozny & M. J. Pratt) North-Holland (1993).

8. International Organisation for Standardisation (ISO), International Standard 10303-1, Product Data Representation and Exchange, Part 1: Overview and Fundamental Principles, 1994.

9. Shah, Jami, and Mantyla, Martti, 1995, Parametric and Feature-Based CAD/CAM, Wiley Interscience, New York.