

## UL-PML: constraint-enabled distributed product data model

Y. WANG and B. O. NNAJI\*

The global economy has made manufacturing industry more distributed than ever before. Product design requires more involvement from various technical disciplines at different locations. In such a geographically and temporally distributed environment, efficient and effective collaboration on design is vital to maintain product quality and organizational competency. Current standard computer-aided design data formats do not support design collaboration effectively in terms of design information and knowledge capturing, exchange, and integration. Design constraints cannot be represented and transferred among different groups, and design information cannot be integrated efficiently within a distributed environment. A new design data model, the Universal Linkage model, is developed here to represent design-related information for network-based collaborative design. It incorporates geometric and non-geometric constraints with traditional geometric elements, thus allowing more design knowledge sharing in collaborative design. Segments of design information can be linked and integrated into a set of complete product data. Thus, lean information exchange can be realized. This model, which has good properties of openness and extensibility, is represented by Directed Hyper Graph and Product Markup Language.

### 1. Introduction

Global market calls for collaborative design among designers, manufacturers, suppliers and vendors. The business pressures toward outsourcing let much of the design work of complex products be done across firms. Ford estimates there are up to 800 links of supplier relations, and automotive companies are substantively relying on these suppliers to participate in vehicle design (NSF e-Design Workshop 2000). The Defense Advanced Research Projects Agency (DARPA) estimates that the supply chain accounts for more than 50% of weapon system and major subsystem production costs (Parunak 1997). In such a geographically and temporally distributed environment, efficient and effective design collaboration should be assured to maintain product quality and organizational competency. Advanced collaborative design tools and technologies are needed so that stakeholders such as customers, suppliers, government agencies, retailers and others can participate in product development at the early stages so as to reduce the risk of failure and shorten the design cycle.

There are new issues on information modelling and communication in collaborative design. First, collaborative design over networks requires a common and easy-to-use standard for design information representation. To ensure effective information transferring, design data exchange protocols should be established by

---

Revision received November 2003.

NSF I/UCRC Center for e-Design, University of Pittsburgh, 1048 Benedum Hall, Pittsburgh, PA 15261, USA.

\*To whom correspondence should be addressed. e-mail: nnaji@engr.pitt.edu

the computer-aided design (CAD) industry to support different CAD systems. Current neutral formats only capture static geometric information and part of administrative information. Other information that contains modelling history and design intent such as parameters, features, constraints and other dynamic relations is lost due to translation. To exchange all useful information about a product, a more powerful data format should be developed to integrate various design information.

Second, the information infrastructure that supports Internet-based product development should be established to assist the cooperation among various design and analysis systems. Current CAD data formats were designed for stand-alone systems. All the information about components and assemblies has to be available locally in order to be processed. Transferring CAD information among design collaborators requires a large amount of data to be moved around, which is inefficient with current limitations of the communication bandwidth. Furthermore, corporations do not wish to expose complete design data to customers or suppliers because of information confidentiality. Secure communication among collaborators should be established based on users' need to know and their affiliated organization. Current CAD files lack flexibility on selective design information retrieving and reuse. They do not support partial data queries. If only a fraction of design data is needed, information cannot be retrieved without querying the whole file. A collaborative design data model should support lean information processing. It should be compliant with industry standards of programming, communication, networking, system management, and interfaces between applications and system services. It should also have good compatibility and interoperability with current design and engineering systems.

The paper presents a new scheme, UL-PML, for CAD information modelling within the context of Internet-based services and transactions. To maximize the openness, flexibility, scalability and integrity of collaborative design, this data scheme intends to be portable across different Internet protocols, network configurations and operating systems. This data scheme has a distributed style that supports the required scalability and extensibility of collaborative design systems. A Universal Linkage (UL) model is developed to capture geometric and non-geometric entities and relations. The model also allows design information elements to be linked over the Internet so that geographically distributed design partners can build a logically integrated product model. Graphically, Directed Hyper Graph (DHG) symbolizes the UL model. Computationally, Product Markup Language (PML) (Wang and Nnaji 2001, Wang *et al.* 2003) represents this model. PML has the syntax of eXtensible Markup Language (XML) (WWW Consortium), which is an emerging Internet information transferring standard.

Section 2 reviews different design data models and the special requirements for network-oriented CAD data models. Section 3 presents the UL model and DHG representation. Section 4 describes the basic syntax and semantics of PML. The schemas of PML in the context of mechanical design are also defined. Representation of geometric and non-geometric constraints is discussed in section 5.

## 2. Background

There are two types of relations among geometric entities to be modelled. One is a static relation that exhibits the basic structural or topological information of entities, such as the aggregation relation between a line and its two end points. Another is a dynamic relation that is added by the designer as a constraint, such as the distance

between two points or the concentricity of two holes. The dynamic relations can be changed without altering the structural information of geometric elements. The mechanical design needs to capture both static and dynamic aspects.

### 2.1. *Challenges of CAD data modelling in collaborative design*

Current neutral CAD models (e.g. IGES and STEP) are similar to some general information models such as ER (Elmasri and Navathe 1994), IDEF1X (Kusiak *et al.* 1997) and NIAM (Verheijen and Van Bekkum 1982), which emphasize structural and static relationship of entities. Variant relations among geometric entities such as parameters and constraints cannot be represented. There are some research efforts to include parametric information into STEP. Examples are the programme of Enabling Next GENERation mechanical design (ENGEN) (Shih and Anderson 1997) sponsored by DARPA and PDES, Inc., and the work of the National Institute of Standards and Technology (NIST) Parametric Group (Pratt 1996, 2001). Though certain form features and geometric constraints are modelled in the above research, these representation methods are not generic enough to consider both geometric and non-geometric constraints and to support both implicit and explicit modelling.

Design knowledge and constraints consist of both geometric and non-geometric aspects. Generally, there are four kinds of approaches to solve geometric constraints in parametric systems: numerical, artificial intelligence, symbolic and constructive. A unified constraint representation form is needed to support different kinds of internal representations. Since non-geometric constraints cannot be represented in current geometric modelling systems, designers need to interpret most of the non-geometric constraints into geometric ones. However, there are large amounts of non-geometric constraints that cannot be interpreted and integrated into geometry. It is critical to capture non-geometric constraints explicitly in the design data to retain the source of geometric interpretation and prevent misconception and information loss.

To capture specification and design intent, multidisciplinary engineering constraints should be incorporated into CAD data. To allow design collaborators to understand the design and be able to view, edit, analyse and exchange models effectively, parametric construction and the corresponding transition from implicit to explicit models should be included in an integrated product model to enhance interoperability.

Furthermore, a system-independent data format is vital to ensure the openness of information exchange within a distributed design and engineering environment. It will be advantageous that this ideal format is network-oriented at the implementation level, i.e. compatible to the Internet protocols and open standards. At the semantic level, this format should be object-oriented, which extensively supports data abstraction in a well-developed style. With the emergence of XML, the data exchange over the Internet can have a uniform format.

### 2.2. *Application of XML in CAD data modelling*

XML has the characteristics of being simple, extensible, portable, interoperable and object-oriented. Some research applied XML in CAD/computer-aided manufacturing area for meta-design information capturing and exchange. Ratchev *et al.* (2000) developed a decision-making environment for distributed product and facility prototyping in an extended enterprise. XML is used for conveying design and manufacture messages across traditional technology boundaries. Kahn *et al.* (2001) are

working on a framework for transforming EXPRESS into XML and viewing with standard World Wide Web browsers. Burkett (2001) proposes a mapping between EXPRESS and XML Data Type Definition (DTD). NIST's Design Repository project (Szykman *et al.* 1999, 2000) created XML mappings for function and flow in order to support representation of artefact function models in software systems. However, the above research represents geometry based on existing neutral formats (STEP or VRML). Dynamic relations among geometric entities that represent a large amount of design knowledge are not considered.

### 2.3. *Requirements for design information representation in collaborative design*

How to build good design information models to meet the requirements of mechanical design is important. Spooner (1991) has a list of requirements for object-oriented CAD data models. Data must be modelled as objects organized into aggregation and generalization hierarchies. The data model must allow definition of operations (methods) for objects, the intentions and extensions of objects, and dynamic schemas. It must support the inheritance of properties and operations, strong typing, and recursive object structure. It should also have efficient and flexible capabilities for object update and multiple inheritance. It should provide support for procedure, specification and enforcement of data integrity constraints.

Eastman and Fereshetian (1994) propose criteria to evaluate product models in CAD/computer-aided manufacturing development. A good model should provide full abstract data types that include object behaviour, multiple specialization, composite object and relation within composition. It must have the ability to model relations on object structure, relations between variables, and variant relations for schema evolution and the state of integrity. The model should also support integrity management of external applications, management of partial integrity for iterative design, and schema evolution for design evolution and refinement.

From the viewpoint of interoperability, the ideal representation model for collaborative design should have the following properties. It is declarative in nature and self-explanatory. It should be able to capture the inherent properties and relations among objects explicitly. Those relations include functional, structural and performance aspects, as well as parametric, spatial and other constraints. Properties and relations should maintain good persistency during information exchange and design evolution. The model should be semantically comprehensive. The engineering meaning of design can be clearly uttered. The model should be both modularly self-contained and flexible so that various objects and their relations can be captured, stored and queried in an arbitrary manner. Additionally, the representation should be extensible. When new entities and relations are needed, it should be able to be extended. At last, to encourage openness, this model should also be simple enough and comprehensible to both humans and machines.

An open model needs to represent product data and design constraints effectively and thoroughly so that all relevant product information can be carried and exchanged seamlessly. The UL-PML scheme hence is developed for this purpose to overcome the shortcomings of the existing models.

### 3. **Universal Linkage (UL) model**

Three fundamental questions should be answered to build a design information structure. (1) What kinds of information elements are to be captured? (2) How would these elements be represented? (3) How can information be retrieved from these

elements? These questions deal with information abstraction, representation and deduction.

3.1. Information elements of the UL model

Similar to other information models, the UL model has the fundamental elements of *entities* and *relations*. An entity is an object that exists as a distinguishable unit in the universe of discourse of design. It should possess unique attributes and be an abstract image of any real object. It is the associated attributes that identify or modify an entity. A relation captures the logical or natural association between two or more entities.

Relations are categorized into two types: *static* and *dynamic*. A static relation indicates the essential and inherent affiliation of entities in order to form a physical object. Static relations form the basic structure of a part or assembly, which represent inherent geometric and topological affiliations. Static relations include *aggregation*, which transforms a relationship between objects into a higher-level object; *generalization*, which refers to an abstraction in which a set of similar objects is regarded as a generic object; and other *general association*. In CAD information models, geometry-related relations mostly are aggregations while non-geometric (e.g. administration, material) relations include both that of aggregations and generalizations. Generalization is mostly used in the meta level of model definition. Dynamic relation specifies the extrinsic affiliation among entities that indicates additional connection or preference, such as dependency, limitation, or restriction. It is specified operationally by designer. Unlike the ER-type models, which only capture static relations, the UL model differentiates static and dynamic relations because dynamic relations are crucial for constraint representation.

3.2. Directed Hyper Graph (DHG)

Graphically, UL model can be represented by DHG, in which a *node* denotes an entity and an *arc* stands for a relation. A geometric entity is represented by an elliptical node in DHG, while a non-geometric entity is represented by a rectangular node (figure 1).

Arcs with solid line in DHG represent static relations (figure 2). Arcs with dash line denote dynamic relations or constraints. A constraint relation is identified by a *constraint entity*, which can be either geometric or non-geometric.

The definitions of entities and relations should satisfy the following requirements: all types of relations are antireflexive. Aggregation and generalization have transitive properties. The direction of an arc implies a specific asymmetric unitary meaning of the relation. A constraint entity is associated with one, two or more entities.

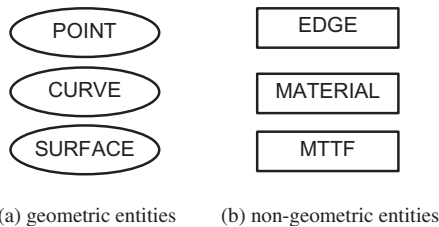


Figure 1. Examples of geometric and non-geometric entities in Directed Hyper Graph.

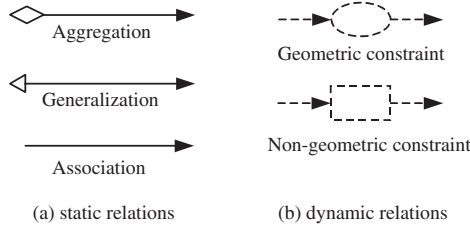


Figure 2. Static relations and dynamic relations in Directed Hyper Graph.

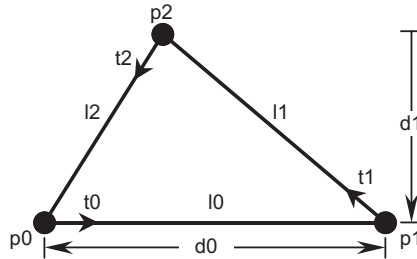


Figure 3. Triangle with dimensional constraints.

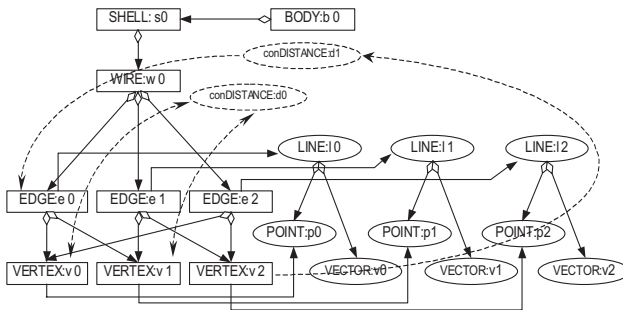


Figure 4. Directed Hyper Graph representation of the triangle in figure 3.

Figure 3 shows a two-dimensional triangle with dimensional constraints. Its geometric and topological information as well as constraints can be modelled in DHG (figure 4).

3.3. *UL among entities*

To enable the seamless composition of a product from different groups, a new modelling technique is needed to support the integration of distributed design information. Besides differentiating the static and dynamic relations among entities, another key feature of the UL model is that relations among entities are not restricted within one data file. The relations of entities located in different files and domains can also be created. Relations are linkages among information elements. A linkage model allows physically distributed entities to be linked, thus a logically integrated set of design information can be built. As shown in figure 5, relations of entities (both static and dynamic) in different domains and physical locations can

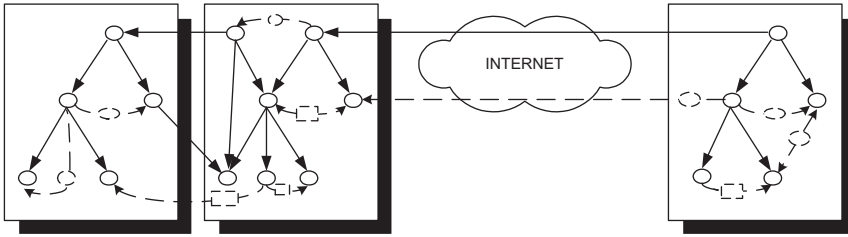


Figure 5. Universal linkage between files.

```
<pml:POINT id="point1" pml:x="1.0", pml:y="1.0", pml:z="0.0">
</pml:POINT>
```

Figure 6. Point in Product Markup Language.

be created. One can easily refer entities in other data files, either at the same machine or at other locations over the Internet.

Graphically, UL model can be illustrated by DHG. Textually, a UL model is represented in PML and processed by computer systems. Section 4 describes the syntax and semantics of PML.

**4. Syntax and semantics of Product Markup Language (PML)**

XML is emerging as the data representation standard for web services. PML is designed to be totally compatible with XML standards.

*4.1. Syntax of PML*

XML provides a common syntax for data modelling. It offers a user-defined and extensible format to represent data and information for different application areas. The syntax of PML strictly follows that of XML to ensure usability and interoperability. The compliance to industrial computation and communication standard is the premise of computational interoperability at the machine level. The syntax of XML is specified at the World Wide Web consortium (W3C XML).

Figure 6 shows a simple example of point modelled in PML following the syntax of XML. Tag set <POINT> and </POINT> specifies the geometric meaning of symbol *point1* and its attributes of *x*, *y* and *z*.

*4.2. Schema of PML*

To enable an XML-style language to be used in a particular area, additional efforts should be made to define the semantics of that language. Specifying the tags used in PML is one of the major tasks in defining PML. This includes what kinds of elements to be used to model geometric and non-geometric entities, what types of attributes to be specified for each entity, and how to capture the relations among entities.

There are two ways to specify the structure of instance documents and the data type of each element and attribute in XML: *Data Type Definition (DTD)* and *Schema*. Some disadvantages of DTD make people turn to Schema. DTD has a different syntax from XML. Two processing systems are needed to process XML and DTD separately. Furthermore, DTD supports a limited capability for specifying

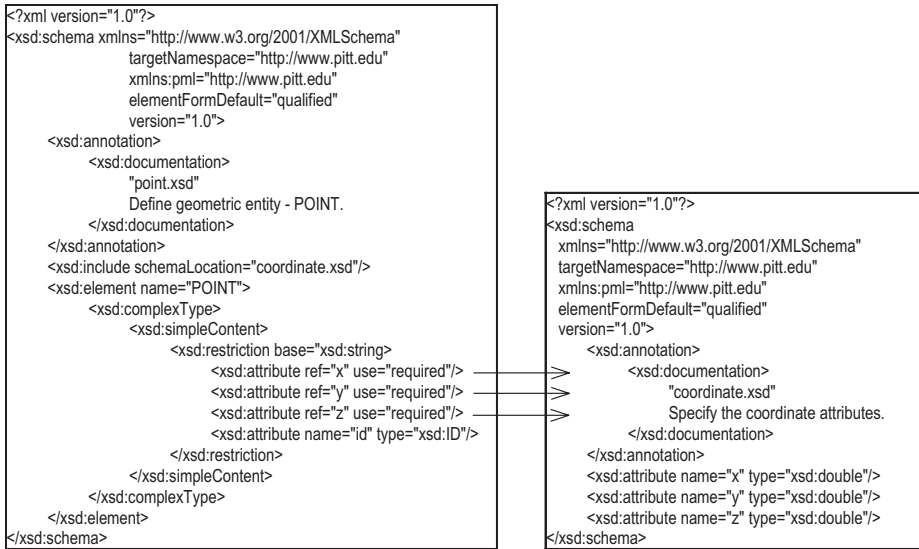


Figure 7. Schema of POINT and the schema of coordinates.

data types. For example, DTD cannot add value range constraints on elements. It does not support all current available data types in database. Comparatively, Schema has advancement over DTD. It uses the same syntax as XML; it is object-oriented and extensible in nature; it has enhanced data-type definition such as element sets, multiple elements with the same name but different contents; and it supports attribute grouping, user defined types and namespace. The PML schemas are defined according to W3C's schema working draft (W3C XML Schema).

Figure 7 shows two simple examples of PML schemas. The left-hand side schema file defines geometric point. A geometric point should contain four attributes, which are coordinate  $x$ ,  $y$ ,  $z$ , and an identification name. The coordinate attributes are defined in the right-hand side schema file, which are referred by the schema of point. Reference between schema files can be built to ensure modularity.

The relation of entities in UL model is modelled by the protocols of XML Xlink (W3C XML Xlink). There are two kinds of links in Xlink: *simple* and *extended*. Simple links offer shorthand syntax for a common and outbound link with exactly two participating resources. Extended links offer full Xlink functionality, such as inbound and third-party arcs, as well as links with arbitrary numbers of participating resources.

In PML, static relations of UL model are modelled by simple links and dynamic relations are modelled by simple or extended links. Links can be *local* within one file, or *remote* between files. A reference is constructed by a *reference id*, which includes a Uniform Resource Identifier (URI) specifying the name and location of the referred data file and an *entity id*. The syntax of a reference id is defined in figure 8.

To model the data structure of DHG by a tree-structured PML, a mapping process is needed. The mapping from DHG to PML tree is under the guidance of graph decomposition rules, which are described in the following section.



```

<reference_id> ::= # <entity_id> | <URI> # <entity_id>
<entity_id> ::= <part_id> | <assembly_id> | <topology_id> |
               <geometry_id> | <constraint_id>
<topology_id> ::= <body_id> | <shell_id> | <wire_id> | <face_id> |
                 <loop_id> | <edge_id> | <coedge_id> | <vertex_id>
<geometry_id> ::= <surface_id> | <curve_id> | <point_id> |
                 <vector_id>

```

Figure 8. Syntax of reference id.

### 4.3. Graph decomposition

All current CAD models are based on graphs, where entities have a hierarchical structure with relations. The purpose of graph decomposition is to disintegrate the graph structure of a data model into tree structure by introducing virtual entities to mirror some geometric or non-geometric entities. A *mirror* of an entity is a virtual entity that reflects the referred entity, thus containing all the attributes of the original one.

The general principles of graph decomposition are as follows:

- Entities are represented by elements/nodes in PML.
- Relations are represented by links in PML.
- Bondage of a mirror with its original entity is represented by a simple link that is from the mirror to the original entity.
- Aggregation is represented by a parent–child relation of elements/nodes in PML in which the parent is the initial entity and the child is the mirror of terminal entity.
- Association is represented by a parent–child relation of elements/nodes in PML in which the parent is the initial entity and the child is the mirror of terminal entity.
- Dynamic relation (constraint) is represented by a simple link, which is from the constraint entity to the target entity if only one entity is involved in the relation.
- Dynamic relation (constraint) is represented by an extended link whose children specify the initial entities and terminal entities if two or more entities are involved in the relation.

The graph decomposition algorithm is shown in figure 9.

Based on the rules of graph decomposition, the DHG of the triangle example in figure 4 can be transformed into a tree-structured PML (figure 10). Elements with a prefix of ‘con’ are constraint entities. Elements with a prefix of ‘ref’ are mirror entities. The tree structure of PML documents allows computer systems to edit and query design data efficiently. Nodes can be selectively viewed and transferred. All relevant product information can be stored in PML trees in an extensible way, which is flexible for interpretation and integration with different systems’ internal representations.

## 5. Design feature and constraint representation

### 5.1. Design feature representation in the UL-PML scheme

In geometric modelling systems, design features or form features can be represented in two levels. One is termed *implicit* or *unevaluated*, where features are defined by construct procedures and parameters. Another is called *explicit* or *evaluated*, by

```

INPUT: Directed Hyper Graph G = (V, E)
OUTPUT: PML Tree T

Add root node TR of T
TR add child TG (Geometry)
TR add child TT (Topology)
TR add child TC (Constraint)

Search the topological node 'BODY' in G
Add a node A corresponding to 'BODY' as a child of TT
Run the following procedure P with input <'BODY', A>
  P: On input node <M, I>
    START P
      Mark M in G
      FOR each unmarked node N with a path from M
        IF N is a topological entity
          Add a node J corresponding to N
            as a child of TT
          Add a mirror node of J as the child of I
            with a simple link referring to J
          Run P on input <N, J>
        ENDIF
        IF N is a geometric entity
          Add a node J corresponding to N
            as a child of TG
          Add a mirror node of J as the child of I
            with a simple link referring to J
          Run P on input <N, J>
        ENDIF
        IF N is a constraint entity
          Add a node J corresponding to N
            as a child of TC
          Add an extended link locator node LOC1
            referring to M as a child of J
          Add an extended link locator node LOC2
            referring to N as a child of J
          Add an extended link arc node starting
            from LOC1 to LOC2 as a child of J
          IF there is a path from N to M
            Add an extended link arc node starting
              from LOC2 to LOC1 as a child of J
          ENDIF
        ENDIF
      ENDFOR
    END P

```

Figure 9. Graph decomposition algorithm.

which features are defined by low-level geometric and topological elements. The design feature representation in the UL-PML scheme is a combination of intentional and geometric features.

If a product model is defined as a set of points in the entity-relation ( $E$ - $R$ ) space, the model domain  $D = (E, R)$  can be subdivided,  $D = (T \times G, S \times C)$ , where  $T$ ,  $G$ ,  $S$  and  $C$  are subspaces of topology, geometry, structural relation and constraint, respectively. *Priori feature*  $F_i \subset D$  and *posteriori feature*  $F_o \subset D$  contain subspaces of  $T$ ,  $G$ ,  $S$  and  $C$ . Design feature  $F$  then is defined as a *relation* between priori and posteriori properties,  $F = (F_i, F_o) = ([T_i, G_i] \times [C_i, S_i], [T_o, G_o] \times [C_o, S_o])$ , where  $i$  and  $o$  denote priori and posteriori properties correspondingly.  $T_i \cap T_o = \emptyset$  and  $G_i \cap G_o = \emptyset$ . Some examples of features are listed in table 1.

In the UL model, priori features are modelled by introducing a new entity type: *feature entity*. The relation between a feature entity and the corresponding topological and geometric entities in priori feature is defined as aggregation. Similar to low-level entities, feature entities can be referred as both abstract class and instance. Design feature entities are categorized as geometric entities. As an example, the priori feature of protrusion in table 1 is shown in figure 11.

```

<?xml version="1.0"?>
<pml:PART id="part0" xmlns:pml="http://www.pitt.edu" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xlink="http://www.w3.org/1999/xlink" xsi:schemaLocation="http://www.pitt.edu/line.xsd">
  <pml:GEOMETRY>
    <pml:POINT id="p0" x="0.0" y="0.0" z="0.0"/>
    <pml:POINT id="p1" x="20.0" y="0.0" z="0.0"/>
    <pml:POINT id="p2" x="12.0" y="10.0" z="0.0"/>
    <pml:VECTOR id="t0" x="20.0" y="0.0" z="0.0"/>
    <pml:VECTOR id="t1" x="-8.0" y="10.0" z="0.0"/>
    <pml:VECTOR id="t2" x="-12.0" y="0.0" z="0.0"/>
    <pml:LINE id="l0">
      <pml:refPOINT xlink:type="simple" xlink:href="#p0" xlink:show="embed" xlink:actuate="onLoad"/>
      <pml:refVECTOR xlink:type="simple" xlink:href="#t0" xlink:show="embed" xlink:actuate="onLoad"/> <pml:LINE>
    <pml:LINE id="l1">
      <pml:refPOINT xlink:type="simple" xlink:href="#p1" xlink:show="embed" xlink:actuate="onLoad"/>
      <pml:refVECTOR xlink:type="simple" xlink:href="#t1" xlink:show="embed" xlink:actuate="onLoad"/> <pml:LINE>
    <pml:LINE id="l2">
      <pml:refPOINT xlink:type="simple" xlink:href="#p2" xlink:show="embed" xlink:actuate="onLoad"/>
      <pml:refVECTOR xlink:type="simple" xlink:href="#t2" xlink:show="embed" xlink:actuate="onLoad"/> <pml:LINE>
  </pml:GEOMETRY>
  <pml:TOPOLOGY>
    <pml:VERTEX id="v0">
      <pml:refPOINT xlink:type="simple" xlink:href="#p0" xlink:show="embed" xlink:actuate="onLoad"/> <pml:VERTEX>
    <pml:VERTEX id="v1">
      <pml:refPOINT xlink:type="simple" xlink:href="#p1" xlink:show="embed" xlink:actuate="onLoad"/> <pml:VERTEX>
    <pml:VERTEX id="v2">
      <pml:refPOINT xlink:type="simple" xlink:href="#p2" xlink:show="embed" xlink:actuate="onLoad"/> <pml:VERTEX>
    <pml:EDGE id="e0" pml:startParam="0" pml:endParam="20">
      <pml:refVERTEX xlink:type="simple" xlink:href="#v0" xlink:show="embed" xlink:actuate="onLoad"/>
      <pml:refVERTEX xlink:type="simple" xlink:href="#v1" xlink:show="embed" xlink:actuate="onLoad"/>
      <pml:refCURVE xlink:type="simple" xlink:href="#l0" xlink:show="embed" xlink:actuate="onLoad"/> <pml:EDGE>
    <pml:EDGE id="e1" pml:startParam="0" pml:endParam="12.8062484748657">
      <pml:refVERTEX xlink:type="simple" xlink:href="#v1" xlink:show="embed" xlink:actuate="onLoad"/>
      <pml:refVERTEX xlink:type="simple" xlink:href="#v2" xlink:show="embed" xlink:actuate="onLoad"/>
      <pml:refCURVE xlink:type="simple" xlink:href="#l1" xlink:show="embed" xlink:actuate="onLoad"/> <pml:EDGE>
    <pml:EDGE id="e2" pml:startParam="0" pml:endParam="15.6204993518133">
      <pml:refVERTEX xlink:type="simple" xlink:href="#v2" xlink:show="embed" xlink:actuate="onLoad"/>
      <pml:refVERTEX xlink:type="simple" xlink:href="#v0" xlink:show="embed" xlink:actuate="onLoad"/>
      <pml:refCURVE xlink:type="simple" xlink:href="#l2" xlink:show="embed" xlink:actuate="onLoad"/> <pml:EDGE>
    <pml:WIRE id="w0">
      <pml:refEDGE xlink:type="simple" xlink:href="#e0" xlink:show="embed" xlink:actuate="onLoad"/>
      <pml:refEDGE xlink:type="simple" xlink:href="#e1" xlink:show="embed" xlink:actuate="onLoad"/>
      <pml:refEDGE xlink:type="simple" xlink:href="#e2" xlink:show="embed" xlink:actuate="onLoad"/> <pml:WIRE>
    <pml:SHELL id="s0">
      <pml:refWIRE xlink:type="simple" xlink:href="#w0" xlink:show="embed" xlink:actuate="onLoad"/> <pml:SHELL>
    <pml:BODY id="b0">
      <pml:refSHELL xlink:type="simple" xlink:href="#s0" xlink:show="embed" xlink:actuate="onLoad"/> <pml:BODY>
  </pml:TOPOLOGY>
  <pml:CONSTRAINT>
    <pml:conDISTANCE xlink:type="extended" pml:lowerBound="19" pml:upperBound="21">
      <pml:LOC1 xlink:type="locator" xlink:label="start" xlink:href="#v1"/>
      <pml:LOC2 xlink:type="locator" xlink:label="end" xlink:href="#v0"/>
      <pml:ARC1 xlink:type="arc" xlink:from="start" xlink:to="end" xlink:actuate="onRequest"/>
      <pml:ARC2 xlink:type="arc" xlink:from="end" xlink:to="start" xlink:actuate="onRequest"/> </pml:conDISTANCE>
    <pml:conDISTANCE xlink:type="extended" pml:lowerBound="9" pml:upperBound="11">
      <pml:LOC1 xlink:type="locator" xlink:label="start" xlink:href="#v2"/>
      <pml:LOC2 xlink:type="locator" xlink:label="end" xlink:href="#e0"/>
      <pml:ARC1 xlink:type="arc" xlink:from="start" xlink:to="end" xlink:actuate="onRequest"/> </pml:conDISTANCE>
  </pml:CONSTRAINT>
</pml:PART>

```

Figure 10. Product Markup Language representation of the triangular part in figures 3 and 4.

A posteriori feature is modelled as a collection of low-level entities and their association with high-level feature entities. The boundary topological entities of a model are the connections between geometry and feature. Any new face generated in the feature evaluation is associated with the feature entity. The posteriori feature of protrusion in table 1 is shown in figure 12. Through feature entities, two levels of feature representation are linked.

Entity specifications for priori features are independent of those for posterior features. Feature definition is separated from feature evaluation, which allows construction procedure and history to be captured together with geometry.

## 5.2. Geometric constraint representation

Geometric constraints include a variety of types. The commonly used ones can be categorized as given in table 2. Geometric constraints are not mutually exclusive.



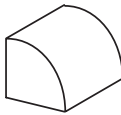





	Priori features		Posteriori features
Protrusion	<i>Profile</i> 	<i>Trajectory</i> 	 $T_o$ : face $\rightarrow$ loop, edge, vertex $G_o$ : surface $\rightarrow$ line, curve, point, vector $C_o$ : dimension/distance, parallelism $S_o$ : association, aggregation
Cut	<i>Profile</i> 	<i>Trajectory</i> 	 $T_o$ : face $\rightarrow$ loop, edge, vertex $G_o$ : surface $\rightarrow$ line, curve, point, vector $C_o$ : dimension/distance, parallelism $S_o$ : association, aggregation
Fillet	<i>Fillet Edge</i> 		 $T_o$ : face $\rightarrow$ loop, edge, vertex $G_o$ : surface $\rightarrow$ line, curve, point, vector $C_o$ : dimension/distance, parallelism $S_o$ : association, aggregation

Table 1. Examples of design feature.

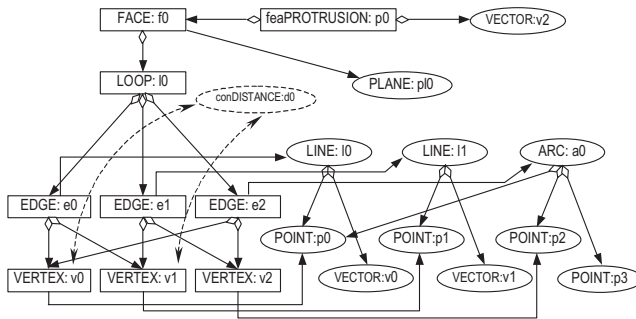


Figure 11. Priori feature of protrusion in Directed Hyper Graph.

One constraint relation may be represented by another constraint. For example, perpendicularity can be represented as angle of  $90^\circ$ . This implies that constraint representation scheme should be flexible enough and extensible.

To prevent information inconsistency, both symbolic and numerical constraints are modelled explicitly in the UL-PML scheme. The symbolic constraints specify

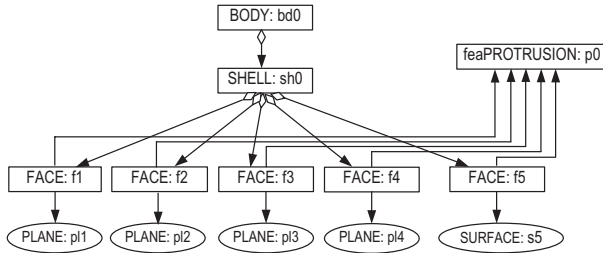


Figure 12. Posteriori feature of protrusion in Directed Hyper Graph.

Dimension	Position	Orientation	Symmetry	Tolerance
Distance	fixed	angle	line symmetry	dimension
Radius	coincidence	horizontal	plane symmetry	straightness
Diameter	concentric	vertical		flatness
	point on curve	curve parallel		circularity
	curve on surface	surface parallel		cylindricity
	curve tangent	collinear		profile of a line
	surface tangent	coplanar		profile of a surface
		perpendicular		angularity
				perpendicularity
				parallelism position
				concentricity
				circular runout
				total runout

Table 2. Categories of common geometric constraints.

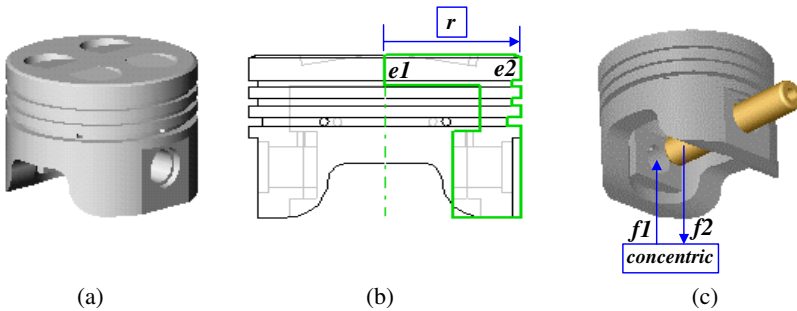


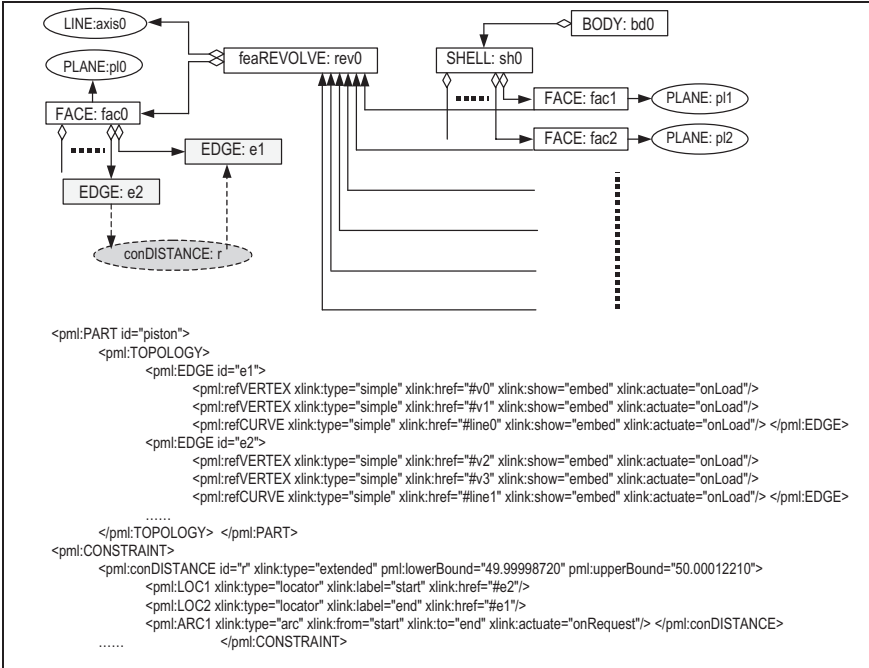
Figure 13. Constraint examples in a piston and its assembly.

geometric relations semantically, which eliminates ambiguity and uncertainty. Geometric constraints are only modelled at the topological and geometric entity level, since form or shape is the major concern of geometric constraints. Each instance of constraints is defined as a constraint entity.

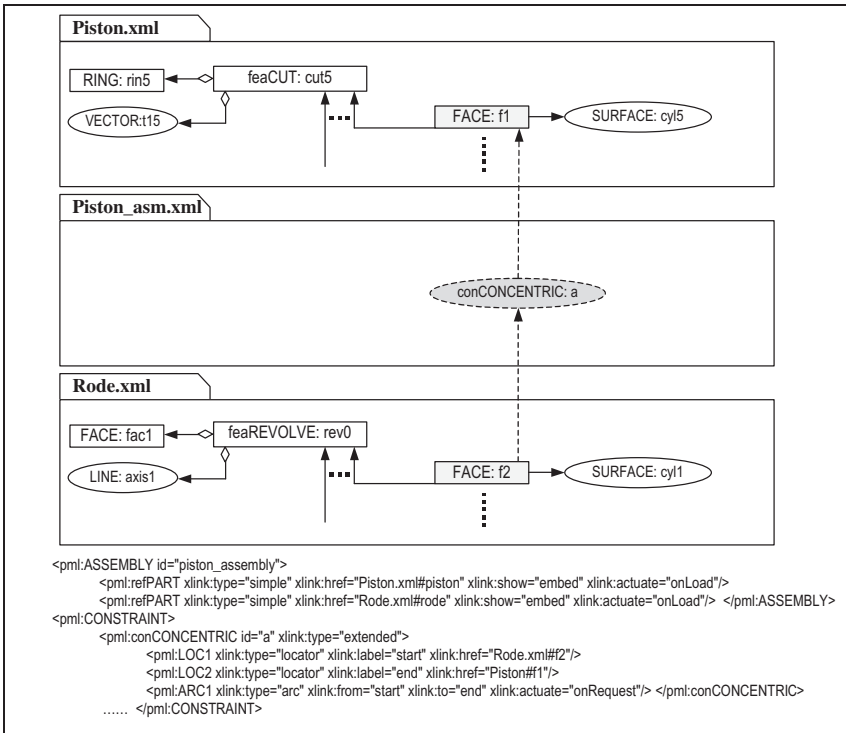
Figure 13 gives some examples of modelling symbolic and numerical geometric constraint for a piston design, The constraints in figures 13(b) and (c) are modelled in PML as in figure 14(a) and (b), respectively.

### 5.3. Non-geometric constraint representation

In UL-PML scheme, non-geometric constraint entities are associated with high-level entities including feature, constraint, part and assembly. To make it general,



(a) distance constraint between edges



(b) concentric constraint between faces in assembly file

Figure 14. Piston geometric constraints in Product Markup Language.

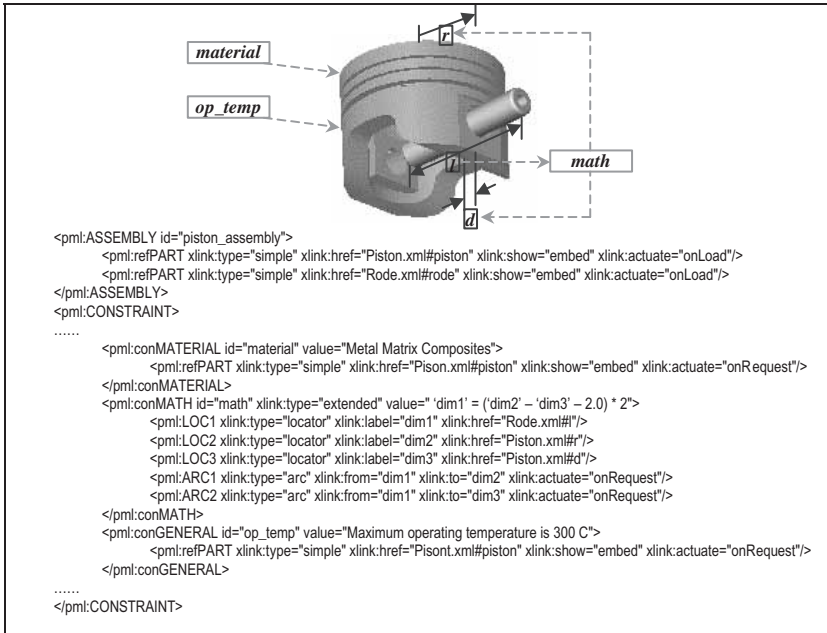


Figure 15. Examples of non-geometric constraints.

non-geometric constraints can be represented symbolically. Character strings are attached to entities of features, parts and assemblies as supplementary information. Domain-specific interpreters are needed to assist design system to understand the constraints. The taxonomy of non-geometric constraints is domain dependent. The UL model can provide a uniform scheme for non-geometric constraint representation. Some examples are shown in figure 15. A constraint can be specific, such as the *material* associated with the part *piston* and the *math* associated with three distance constraints  $d$ ,  $r$  and  $l$  in this example. It can also be general, such as the operation temperature constraint *op\_temp* expressed in character string and associated with the part *piston*.

## 6. Implementation

UL-PML is a distributed CAD data scheme, which allows geometric and non-geometric entities, structures and constraints to be created, stored and queried in a distributed fashion. This allows information to be transferred at the basic entity level rather than at the component level. It provides a flexible way for information exchange intelligently and accumulatively without losing logical integrity. With a top-down approach, a PML tree can provide different levels of detail. With a bottom-up approach, loosely coupled linkage allows lean information transfer.

PML can be applied in two ways. One is to use it as a part of the native data structure in geometric modellers. The other is to transform design data from various formats of existing CAD systems into PML models for information exchange. In this research, both approaches are implemented and tested for polyhedrons. A prototype of geometric modeller using PML as the native data structure is built. Mechanisms of lean information transfers based on protocols of HTTP and CORBA

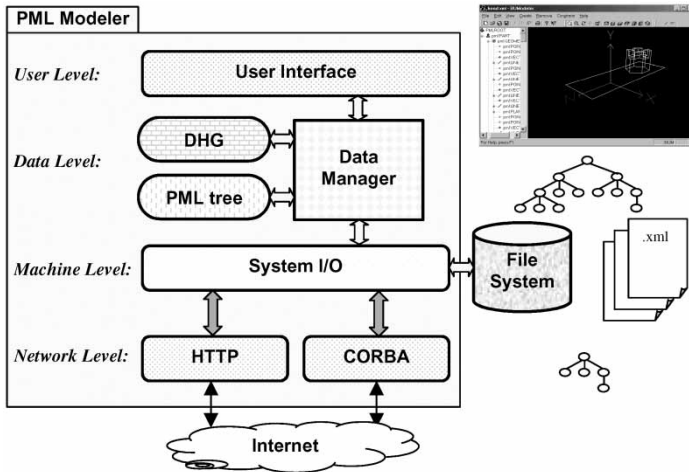


Figure 16. Architecture of a Product Markup Language modeller.

are developed. Distributed geometry and constraint information can be linked based on the UL-PML scheme. The interpretation mechanism between ACIS<sup>®</sup> data structure and a PML model for polyhedrons is developed and tested in an ACIS modeller prototype.

### 6.1. PML modeller

A native PML modeller is developed completely based on PML data format. Figure 16 shows the architecture of the modeller. Within the modeller, geometry can be generated and processed in the form of a PML tree. Data are stored and transferred in PML file format. Users interact with the system in a regular design mode, while the Data Manager is responsible for local PML tree processing and transparent remote data query. The PML modeller uses industry standards for data transfer and remote data access. Design information transfer in PML is independent of network data transmission.

### 6.2. Distributed geometric modelling with different granularity

Unlike current CAD files with the granularity of management at the component level, the UL-PML scheme allows CAD data communication and integration at the basic geometric and non-geometric entity levels. For example, a connector in figure 17 is to be designed by two groups. By the linkage specified as in figure 17(b), the head section at one location (figure 17c) is referring to the top face of the body section that is designed at another location (figure 17d). Thus, one section of a part can be linked to another one during the component design, which enhances design modularization and reuse. Similarly, an assembly file can also refer to distributed files containing components.

### 6.3. Lean information transfer based on HTTP

HTTP is a widely used protocol for Web service. PML remote data access and selective information transfer based on HTTP are developed in the PML modeller. In the example of figure 17, the transfer of face information is based on HTTP (figure 18).



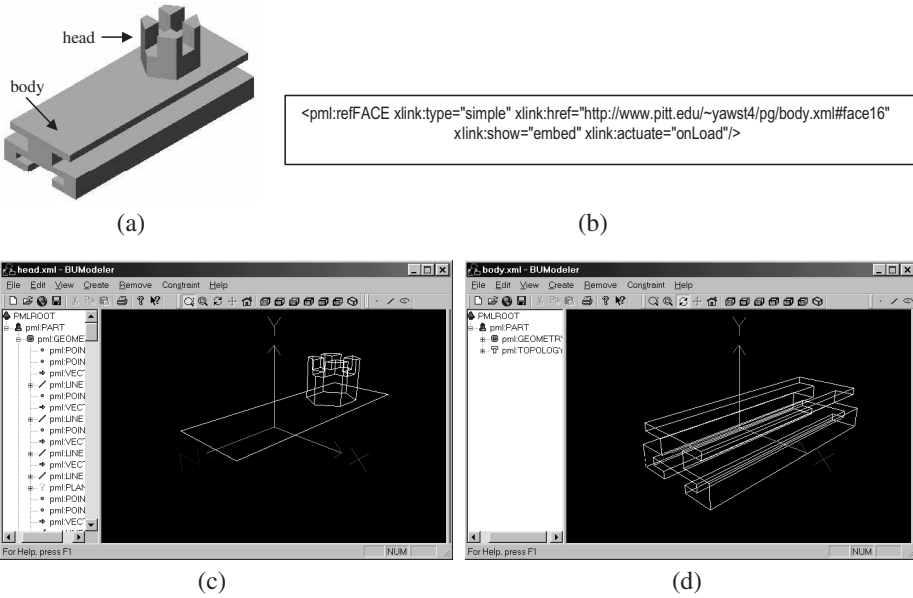


Figure 17. Part to be designed by two groups.

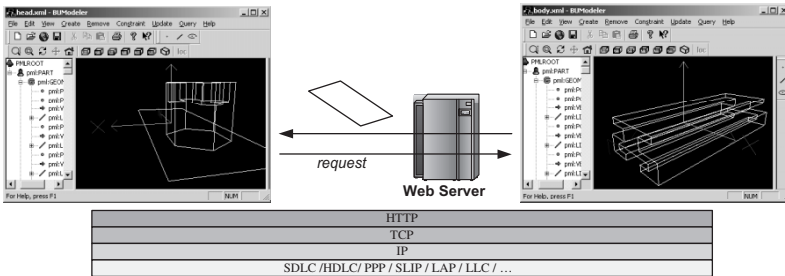


Figure 18. Lean information transfer based on HTTP.

6.4. Lean information transfer based on CORBA

The PML modeller also supports lean information exchange based on protocols of CORBA. Different from HTTP requests, ORB requests are based on a fat-client architecture. The client/server data transfer is transparent through ORB brokers. Clients do not have to specify the IP addresses of the target PML references.

In the example of figure 19, a pair of moulds are designed separately by two groups. Some contacting surfaces of the two parts must match each other geometrically. In the UL-PML scheme, links between the faces in two components can be built. The geometry and topology information of the contacting faces in one can be fetched from the other to maintain consistency. In this linkage relation, *mould1* (figure 19a) is at the server site. Once the data are published in the library (figure 19b) for information sharing, it is available for reference. To meet the surface match requirement, *face504*, *face978* and *face1004* in *mould2* (figure 19c) one is specified to refer to *face3*, *face239* and *face286* in *mould1*, respectively. Instead of transferring the whole data file, only three faces and six bounding edges in table 3 as

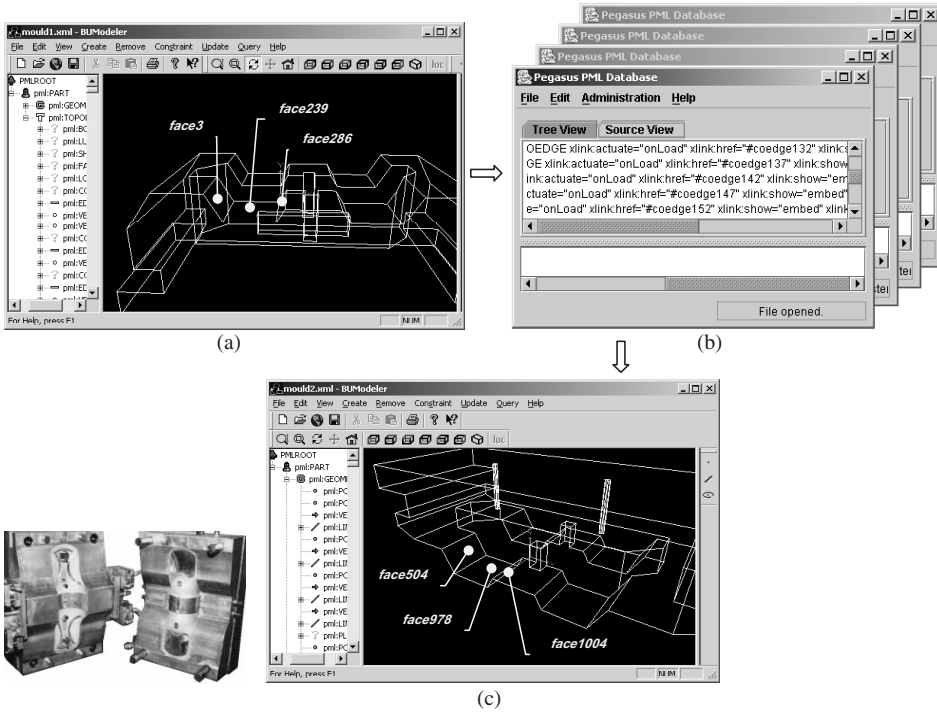


Figure 19. Pair of moulds in collaborative design.

Location	Name	Entity type	Reference	Link type
<i>mould2.xml</i>	<i>face504</i>	face	<i>mould1.xml</i> ## <i>face3</i>	simple
	<i>face978</i>	face	<i>mould1.xml</i> ## <i>face239</i>	simple
	<i>face1004</i>	face	<i>mould1.xml</i> ## <i>face286</i>	simple
	<i>edge508</i>	edge	<i>mould1.xml</i> ## <i>edge13</i>	simple
	<i>edge518</i>	edge	<i>mould1.xml</i> ## <i>edge23</i>	simple
	<i>edge593</i>	edge	<i>mould1.xml</i> ## <i>edge55</i>	simple
	<i>edge635</i>	edge	<i>mould1.xml</i> ## <i>edge168</i>	simple
	<i>edge588</i>	edge	<i>mould1.xml</i> ## <i>edge60</i>	simple
	<i>edge640</i>	edge	<i>mould1.xml</i> ## <i>edge163</i>	simple

Table 3. Selective topology transferred to *mould2*.

well as the corresponding geometry are transferred to the client site through data-sharing agents.

If there is any change about the three faces of *mould1*, *mould2* can be updated by the linkage (figure 20). For each update, only the PML subtrees are transferred across networks. The data transaction based on ORB protocols is shown in figure 21.

6.5. Data transformation between the ACIS and PML models

To demonstrate the possibility of integration between PML and current CAD systems, a geometric modeller prototype based on an ACIS kernel is developed, and geometry information transformation between the PML and the ACIS models for polyhedrons is implemented. Figure 22 shows the architecture of the ACIS modeller

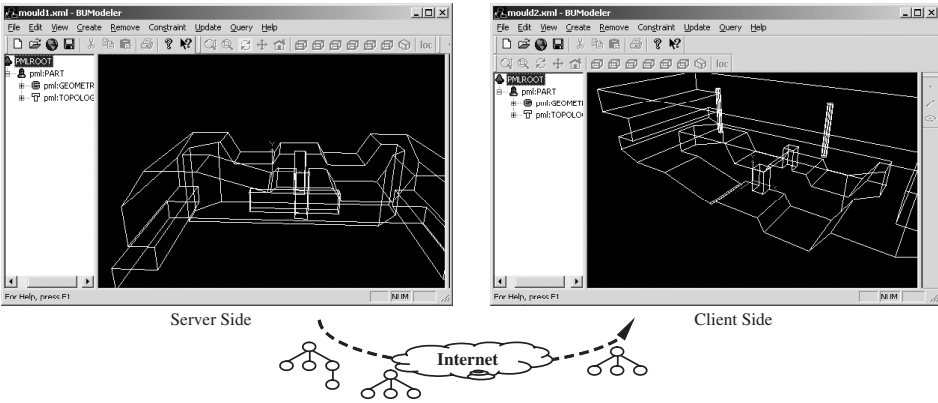


Figure 20. Propagation of an updated design.

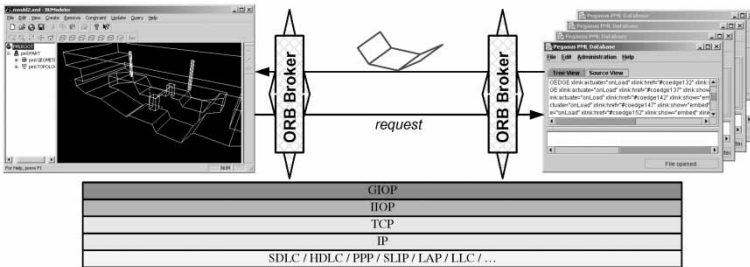


Figure 21. Lean information transfer based on CORBA.

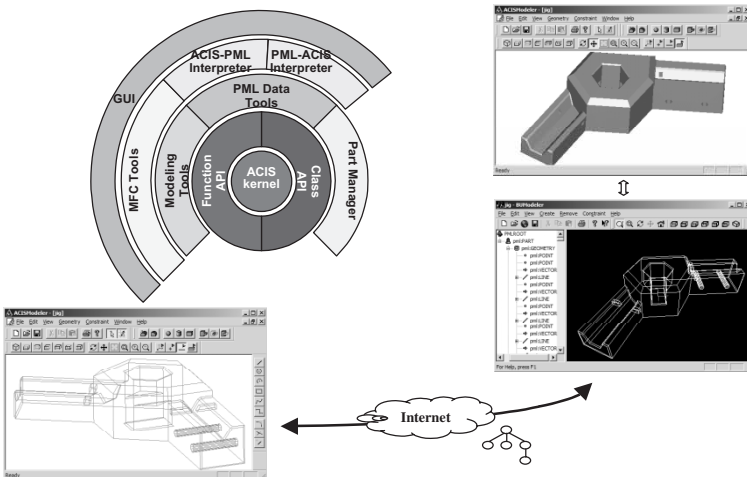


Figure 22. ACIS® model lean information sharing based on Product Markup Language data models.

and lean information sharing based on UL-PML data models. The interpreters between ACIS and PML data structures allow the ACIS modeller to work with the PML modeller in parallel. And lean information sharing channels between ACIS models can be built based on PML data models.

## 7. Conclusions and discussions

In summary, a UL-PML scheme captures geometric and non-geometric relations among entities in a virtual link style in PML so that references between entities can be made across the boundary of files and physical locations in a distributed design environment. This scheme allows design information to be integrated in a collaborative design environment. Besides static relations among design objects, dynamic relations/constraints are also incorporated. PML uses standard XML syntax, and schemas of PML are defined for entities and relations. Tree-structured PML allows design information processing and manipulation to be easily managed. Graph decomposition is needed to map graph-structured entity relations to the PML tree. The properties of UL-PML scheme include the following:

- Network-aware data model intends to improve design *information* interoperability based on general *data* interoperability. At the syntax level, the openness of the UL model is guaranteed. Thus, semantics level interoperability is independent from syntax level interoperability.
- UL model does not require that one data file contain all the information relevant to the designed product. Incorporating physical distribution and logical integration, it makes partial design information storage and retrieval easy to realize. This provides another level of granularity and increases the flexibility during design information query.
- Design information can be stored modularly without compromising the integrity of the whole product. This reduces the requirement for computational time and storage space. Hence, it increases flexibility for scalable designer systems and encourages reuse of designed components/sections.
- Explicit linkage ensures product data's consistency in a distributed environment. Relations of design data elements and constraints are built in the UL model to create a distributed information framework, thus lean information sharing and exchange for collaborative design can be realized over the Internet.
- Integrated geometric and non-geometric constraint representation in the UL model incorporates more design knowledge into design data. The explicit capturing of multidisciplinary constraints, especially non-geometric constraints, enables a more complete information representation than current standard formats, which can provide a more comprehensive support for design intent representation at different design stages.

## References

- BURKETT, W. C., 2001, Product data markup language: a new paradigm for product data exchange and integration. *Computer-Aided Design*, **33**, 489–500.
- EASTMAN, C. M. and FERESHETIAN, N., 1994, Information models for use in product design: a comparison. *Computer-Aided Design*, **26**, 551–572.
- ELMASRI, R. and NAVATHE, S. B., 1994, *Fundamentals of Database Systems*, 2nd edn (Menlo Park: Addison-Wesley).
- KAHN, H., FILER, N., WILLIAMS, A. and WHITAKER, N., 2001, A generic framework for transforming EXPRESS information models. *Computer-Aided Design*, **33**, 501–510.

- KUSIAK, A., LETSCHE, T. and ZAKARIAN, A., 1997, Data modelling with IDEF1X. *International Journal of Computer Integrated Manufacturing*, **10**, 470–486.
- NSF E-DESIGN WORKSHOP, 2000, The white paper of U.S. National Science Foundation workshop on e-product design and realization for mechanically engineered products (available at: <http://www.e-designcenter.info>).
- PARUNAK, H. V. D., 1997, Distributed collaborative design (DisCollab): an ATP opportunity (available at: <http://www.mel.nist.gov/msid/groups/edt/ATP/white-paper>) [White paper of an NIST-ATP Workshop, 'Tools and Technologies for Distributed and Collaborative Design', August 1997].
- PRATT, M. J., 1996, *Requirements Analysis for an Explicit Constraints Schema for STEP*. ISO TC184/SC4/WG3 N502 (T1/Parametrics) 10 May 1996 (available at: <http://www.nist.gov/sc4/paramet/short/iso/n502.pdf>).
- PRATT, M. J., 2001, Extension of the STEP standard for parametric CAD models. *Journal of Computing and Information Science in Engineering*, **1**, 269–275.
- RATCHEV, S. M., SHIAU, J. and VALTCHANOV, G., 2000, Distributed product and facility prototyping in extended manufacturing enterprises. *International Journal of Production Research*, **38**, 4495–4506.
- SHIH, C. H. and ANDERSON, B., 1997, A design/constraint model to capture design intent. *ACM Proceedings of the Fourth Symposium on Solid Modeling and Applications*, pp. 255–264.
- SPOONER, D. L., 1991, Towards an object-oriented data model for a mechanical CAD database system. In K. R. Dittrich, U. Dayal and A. P. Buchmann (eds), *On Object-Oriented Database Systems* (Berlin: Springer), pp. 190–205.
- SZYKMAN, S., SENFAUTE, J. and SRIRAM, R. D., 1999, The use of XML for describing functions and taxonomies in computer-based design. *Proceedings of the 1999 ASME Design Engineering Technical Conferences*, paper no. DETC99/CIE-9025.
- SZYKMAN, S., SRIRAM, R. D., BOCHENEK, C., RACZ, J. W. and SENFAUTE, J., 2000, Design repositories: engineering design's new knowledge base. *IEEE Intelligent Systems*, **15**, 48–55.
- VERHEIJEN, G. M. A. and VAN BEKKUM, J., 1982, NIAM: an information analysis method. In T. W. Olle, H. G. Sol and A. A. Verijin-Stuart (eds), *Information Systems Design Methodologies: A Comparative Review*. Proceedings of the IFIP WG8.1 Working Conference on Cooperative Review of Information Systems Design Methodologies (Amsterdam: North-Holland), pp. 537–589.
- W3C XML SCHEMA, XML schema work draft (available at: <http://www.w3.org/TR/xmlschema-0/>, <http://www.w3.org/TR/xmlschema-1/> and <http://www.w3.org/TR/xmlschema-2/>).
- W3C XML XLINK (available at: <http://www.w3.org/TR/xlink/>).
- W3C XML, XML 1.0 recommendation (available at: <http://www.w3.org/TR/REC-xml/>).
- WANG, Y. and NNAJI, B. O., 2001, Functionality-based modular design for mechanical product customization over the Internet. *Journal of Design and Manufacturing Automation*, **1**, 107–121.
- WANG, Y., KIM, K. Y., MUOGBOH, O. S. and NNAJI, B. O., 2003, Distributed CAD data modeling over the Internet. *Proceedings of the 17th International Conference on Production Research*, paper no. 0186.
- WWW CONSORTIUM (available at: <http://www.w3.org/XML/>).