

A Feature-Based Framework for Semantic Interoperability of Product Models

Ravi Kumar Gupta* - B. Gurumoorthy

Indian Institute of Science, Centre for Product Design and Manufacturing, Bangalore, India

The paper addresses the problem of exchanging product semantics along with other product information such as shape. Exchanging the semantics/meaning associated with shape data enables manipulation of and reasoning with the shape model at higher levels of abstraction. The semantics associated with shape data can convey design intent, inter-relationships between entities in the shape and other data important for downstream applications such as manufacturing. As the product model does not support semantics, its use in other systems/domains leads to construction of new models. Using a single product model across the product lifecycle is beneficial from the point of view of maintaining integrity of the data and avoiding the effort in creating multiple models. The paper first identifies different types of semantic interoperability problems arising during exchange of product models in product development. These are: different terms referring to same shape, different representations for a shape, meaning of terms are context dependent and mismatch in entities supported in two. We present a one-to-many framework for exchange of product information model, product semantics in particular. This framework is built using the Domain Independent Form Feature (DIFF) model as the representation of features in the shape model along with an ontology that captures the vocabulary in use in feature models. A reasoning module that can extract multiple construction /views of a feature has also been developed. This reasoning module is used to associate multiple construction paths for the features and associate all applicable meanings from the ontology with the DIFF model. Each CAD system can now use the semantics and construction history supported by it to further manipulate the product model. Results of implementation of the use of the above solution in exchanging product semantics with a commercial CAD system will be presented and discussed.

© 2008 Journal of Mechanical Engineering. All rights reserved.

Keywords: information exchange, product lifecycle management, product data exchange, semantic interoperability

1 INTRODUCTION

Given the sheer complexity and variety required in products today to meet the requirements of an increasingly savvy and aware customer, it is impossible for any organisation to manage the product development process without collaboration [1].

Collaboration across multiple locations, multiple domains/disciplines is required to be able to deliver the right product at the right time and right cost. For such a collaboration to be successful, not only data but information and knowledge must be exchanged.

Product Lifecycle Management (PLM) is emerging as a "computational framework which effectively enables capture, representation, retrieval and reuse of product knowledge" across the product lifecycle to support such a knowledge-intensive product development environment [1].

If PLM as a solution has to include all phases in the product lifecycle and all the stakeholders, then exchange of data and information between the different phases and stakeholders becomes a critical element of PLM. Exchange of product information (including data) as opposed to data alone is a key differentiator of PLM over the earlier approaches. The motivation being that if product information is exchanged then, it is possible to have knowledge based solutions in each phase to reason about the information to arrive at decisions.

Currently, exchange of data requires the use of dedicated translators or recreation of models. Product information (data at a higher level of abstraction) however is exchanged only through human intervention. With product development happening in multiple locations with multiple tools/systems, semantic interoperability between these systems/domains becomes important.

Semantic is the meaning associated with a terminology in a particular context and interoperability means the ability to work together to accomplish a common task. So, semantic interoperability of product model refers to automatic exchange of meaning associated with the product data, among application domains throughout the product development cycle. Application domain refers to any of the following - product design, manufacturing, ERP, CRM, and SCM. Semantic interoperability implies the existence of a common and shared understanding of the meaning underlying the information that is being exchanged [4]. In contrast to the common usage of the term "product semantics" in the design community, our interest is in the semantics of the product information that is being exchanged and not the semantics communicated by the product itself.

Exchanging the semantics/meaning associated with shape data enables manipulation of and reasoning with the shape model at higher levels of abstraction. The semantics associated with shape data can convey design intent, inter-relationships between entities in the shape and other data important for downstream applications such as manufacturing. As the product model does not support semantics, its use in other systems/domains leads to construction of new models. Using a single product model across the product lifecycle is beneficial from the point of view of maintaining integrity of the data and avoiding the effort in creating multiple models. Lack of semantic agreements is due to several reasons. Semantics associated with data and procedures is not explicitly represented and is often context-dependent. Mismatch in terms and meanings also arise due to independent development efforts often aimed at establishing proprietary naming and other conventions. Resolving the semantic mismatch in most domains requires the involvement of people. In the product development cycle several different domains (engineering design, industrial design, manufacturing, supply chain, marketing) come into play making the ability to exchange product data with semantics very critical.

1.1 Product Data Exchange

Exchange of product data has undergone considerable evolution since the days of annotated

engineering drawings. At that point the focus was to exchange primarily shape/geometric data between design and manufacturing. With the advent of computer aided design and drafting systems, exchange of shape models between different CAD/CADD systems was required. Different approaches being used to handle the interoperability problem between product models are - a single CAD environment for all tasks, direct data transfer between different systems which requires $(n(n-1))$ translators for "n" tools. Another approach uses neutral file formats. This approach requires $(2n)$ translators for "n" tools as depicted in Figure 1.

Use of neutral format therefore became the preferred framework to solve the data exchange problem. The Drawing Exchange Format (DXF) is the defacto neutral format used to exchange 2D drawing data across different drawing tools. Then, Initial Graphics Exchange Specification (IGES), another neutral format, was introduced for exchange of geometry information between dissimilar systems. IGES however, is capable of transferring only the geometry of the product; the non-geometry and design intent are lost. Standard Exchange of Product data model (STEP, formally ISO 10303) evolved to interrelate all geometric and non-geometric data in a useful and meaningful way to represent product content model so that the complete description can be exchanged between CAD systems. Standard for Transfer and Exchange of Product model data (STEP) is at present most comprehensive standard to address the needs for exchange of geometric data. A major advantage of STEP (that is yet to be fully exploited) is that it is possible to develop standards for exchange of data between different domains in the product lifecycle. Analysis and manufacturing are two of the domains that have been handled so far. With the emergence of features technology in CAD systems the problem of exchanging feature models (that are geometric data at higher levels of abstraction) became important. Current art in exchange of product data

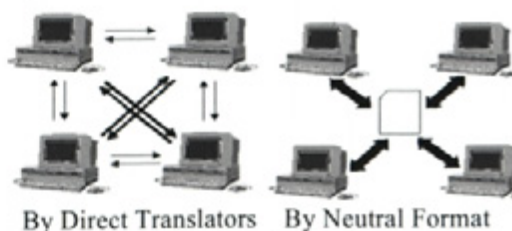


Fig. 1. Product data exchange

is at the level of exchanging feature models. Feature based product data exchange is emerging to accommodate design intent for data exchange. Features are capable of carrying constraints, parameters and application attributes. A working draft on construction history features has been developed by the STEP group [6]. Current art in exchanging part geometric data (shape models) does not solve the semantic interoperability problem as the shape model does not convey the product semantics.

1.2 Issues in Semantic Interoperability in Product Development

In the present paper, the focus is restricted to exchange of shape models. Semantic interoperability arises due to the use of shape model in different systems and different domains. Different types of semantic interoperability problems arising during exchange of shape models in product development are first identified.




1.2.1 Different Labels Referring to Same Shape

Two or more terms may refer to same shape in product development environment. A cylinder removed from one another cylinder which can be defined in terms of bush and circular hole referring to the same shape as shown in Table 1. Similarly one can define other models in the Table 1.

1.2.2 Different Representation for Same Shape

Associated representation for a shape may be different. For example cylinder removed from one another cylinder can be obtained by revolution, sweep or extrusion as described in Table 2.

Table 1. Shape has different labels

Three dimensional model			
Term	Bush Circular hole	Cube Box	Slot Region between two ribs Two ribs on a plate
Shape (Meaning)	Cylinder removed from one another cylinder	Block of equal length, width and height	Block removed from one another block

Similarly other model in Table 2 can be obtained by extrusion or sweep.

1.2.3 Meaning of a Term is Context Dependent

The term condenser has a different meaning in heat transfer domain and the electrical engineering domain. Examples in product design and manufacturing are shown in Table 3.

2 LITERATURE REVIEW

As mentioned earlier the need for sharing and exchanging product data between various domains has been around for a while now [17]. Owen [10] and Pratt [12] provide a review of the work on exchange of data and features between CAD systems. Most efforts in exchanging semantics involve features. This is only natural as features evolved to carry semantic information about form, function and behaviour [2]. In this section we focus only on those efforts that address the exchange of product semantics using features. There have been several attempts in defining ontologies for features [5] and [14]. The focus here is to extend feature specification by using ontology of design concepts (as high level modelling entities) to link product function to shape. Most efforts in building feature ontologies have focused on

Table 2. Shape has different representations



Three dimensional model		
Shape (Meaning)	Cylinder removed from one another cylinder	Block removed from one another block
Representation	Extrusion of 2D face (2 concentric circles (rin, rout)) to height "h"	2D face swept between two faces 2D face (rectangle) revolved around an axis through 360° Extrusion of 2D face to depth "h"

Table 3. Semantics in product design and manufacturing [11]

Term	Meaning in particular context	
	Solid modeling	2.5D machining
Extrusion	extruded geometry	2.5D machined object
hasShape	has 3D geometric shape	Has machining contour
Cube	cubical shape	rectangular machining contour
Block	extruded object with cubical shape	2.5D object machined with rectangular contour

capturing taxonomy and not on any reasoning based on the ontology [2].

Brunetti, et al. [2] propose the use of features to achieve a semantic interface to different CAX applications. They describe a conceptual framework of how an ontology of features and shape can be used to provide a semantic retrieval system or semantic interface to 3D modelling systems. The framework prescribes ontologies at different levels of abstraction namely, the model, features, constraints, topology and geometry that are available in a CAD system. The paper only describes the conceptual model and no implementation is described.

Patil, et al. [11] also present an ontology based approach to enable semantic interoperability. They propose the use of an ontology defined in Product Semantic Representation Language (PSRL) as an intermediate layer between the two systems that need to exchange the product data and semantics. Semantic translation then becomes a problem of mapping from one system to the ontology in PSRL and then from this to the target system. The axioms and definitions that form the ontology in PSRL have to be a union/superset of the terms in the systems exchanging data. Therefore for every new system to be included, the ontology in the PSRL has to be extended with the new terms or labels not present in the ontology.

Mostefai, et al. [8] propose an ontology based approach to enable collaboration. The proposed ontology supports queries on the product model across three views (design, assembly and manufacturing). They also mention the concept of equivalence between entries in the ontology that is similar to the first type of interoperability problem identified in this paper. In their approach the linkages between the entries in the ontology have to be specified and the ontology editor then uses these linkages to answer queries and establish equivalence. The ontology proposed would have to be significantly expanded for them to address the semantic mismatches identified in the present work.

Subramani [16] describes another approach to exchange the product data via feature models. In this work, feature-volume based product data exchange is proposed. Feature-based modeling captures semantics and the designer's intent through parameters and constraints. This method transfers product data as feature volumes; feature

volume contains feature faces and their attributes. STEP definition of faces and geometry is used to represent the feature volume. Construction history of the feature model is recreated using the face attributes. Unlike current methods for data exchange, the proposed scheme allows exact representation of 2D and 3D constraints through face classification and multiple construction procedures for each feature instance. The latter allows handling of situations where the receiving system does not support some of the procedures in the source system. Since individual feature volumes are transferred, constraint and parameter representation is preserved and validation of features with respect to the part model is avoided. The proposed method has been implemented using the eXtensible Markup Language (XML), which carries semantic representation.

Presently, the use of XML schema has been proposed to enable the exchange of data between different systems/applications. Several XML schemas have already been proposed by researchers [7] and [16] and vendors (3DXML, PLM-XML, X3D). However, these focus only on enabling exchange of data and visualization of shapes.

3 OVERVIEW

Our research is focused on enabling seamless exchange of product information (as opposed to only shape data) across the entire product lifecycle. As a first step to this goal, the present study aims at exchanging product semantics along with product shape.

In an earlier work, a framework based on domain independent form features (DIFF) (Fig. 2)

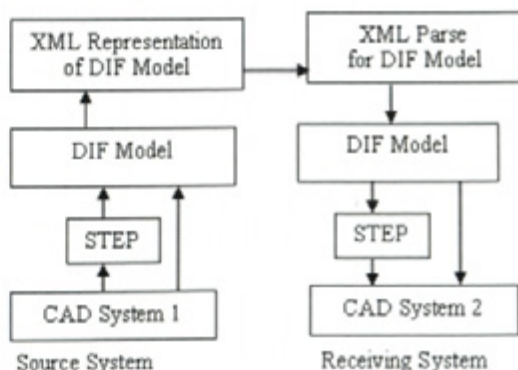


Fig. 2. Feature-base data exchange architecture [16]

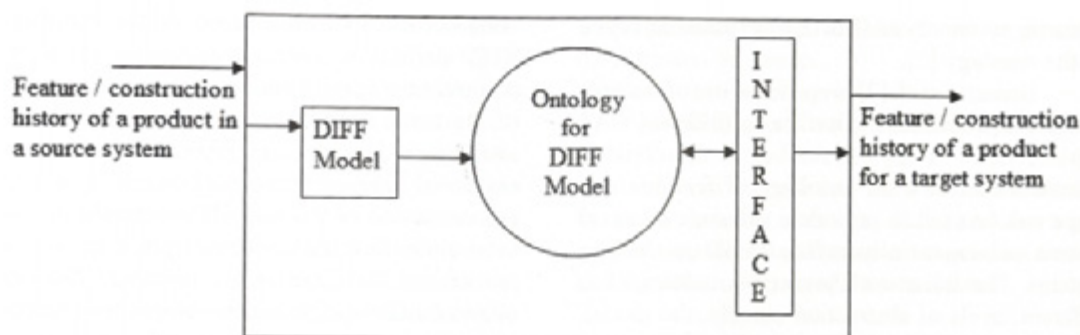


Fig. 3. Schematic diagram for semantic interoperability of product model

was proposed to enable exchange of feature models between CAD systems [16].

In the present work, we present a one-to-many framework for exchange of product information model, product semantics in particular. This framework is built using the DIFF as the representation of features in the shape model along with an ontology that captures the vocabulary in use in feature models. Though Figure 3 shows the schematic for only one source and target system, there can be any number of target systems and source systems. Interface is used to select / read features and construction history of a product for a target system.

Given the feature or construction history in the source system, feature volumes in the DIFF format can be constructed [15]. From the DIFF model, alternate labels for use in the target system can be identified. If there is no label matching (target system is new) then matching involves construction of DIFF models for all the labels available in the target system and then finding a match by comparing the DIFF models obtained with the DIFF model corresponding to the feature to be exchanged. In order to exchange construction history associated with the feature label a similar procedure is followed. A reasoning module that can extract multiple construction / views of a feature has been developed. This reasoning module is used to associate multiple construction paths for the features and associate all applicable meanings from the ontology with the DIFF model. Each target system can now use the semantics and construction history supported by it to further manipulate the product model. In case of mismatch in the labels of terms in the construction history, the correct construction path is identified by matching the corresponding DIFF features. However the present implementation has been done using an ontology editor in the interests of quick prototyping. A

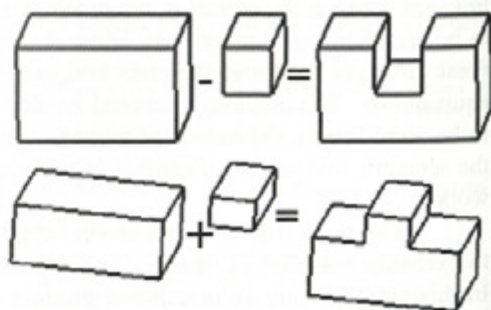
prototype of the ontology and the reasoning on the ontology has been built using the Protégé ontology editor [13]. In the following section we first briefly describe the DIFF feature structure followed by a description of the tool developed.

4 DOMAIN INDEPENDENT FORM FEATURE (DIFF) MODEL

Feature is defined in terms of faces and faces adjacency relationships. Features are viewed as formed by subtracting/adding a single solid-piece from/to a base-solid as depicted in Figure 4. The solid existing before subtraction or addition is referred to as the base-solid and the solid subtracted or added is referred to as solid-piece [9]. The created feature inherits the structure of the solid-piece. The classification of faces and faces adjacency relationships in the DIFF model is described in the following section.

4.1 Classification of Feature Faces

The faces that form the closed shell are classified as shell-faces and the two faces which



Base-solid \pm Solid-piece = Created feature (final solid)

Fig. 4. Feature as formed by subtracting/adding a single solid-piece from/to a base-solid

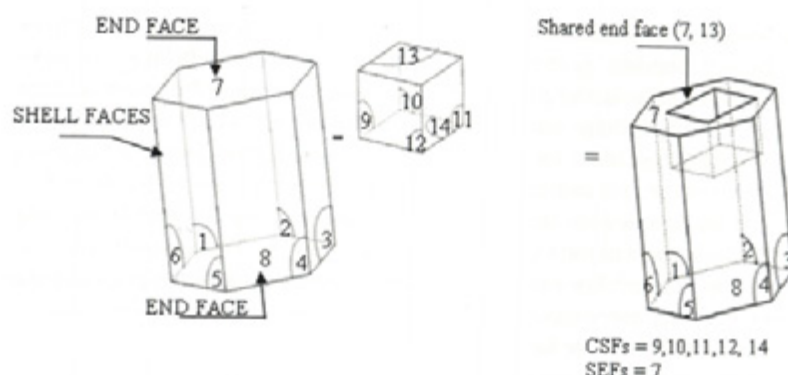


Fig. 5. Classification of feature faces

close the ends of the shell are classified as end-faces as shown in Figure 5. Addition or subtraction of the solid-piece leaves an impression (feature) on the base-solid. The faces in the impression which did not exist in the base-solid before the addition or subtraction operation, are classified as created faces (newly created faces). The neighbouring faces of the impression exist in the base-solid before the operation, are shared by the solid-piece and the base-solid which are classified as shared faces (modified faces) as shown in Figure 5.

The faces in the final solid associated with an individual feature are classified as follows:

- Created shell faces (CSFs); newly created faces in the base-solid corresponding to the shell-face of the solid-piece.
- Shared shell faces (SSFs); already existing faces in the base-solid corresponding to the shell-face of the solid-piece.
- Created end faces (CEFs); newly created faces in the base-solid corresponding to the end-face of the solid-piece.
- Shared end faces (SEFs); already existing faces in the base-solid corresponding to the end-face of the solid-piece.

Since features are defined in terms of these four types of faces, feature definitions are consistent and machine-understandable. These four types of faces of each feature are stored in the feature model with face adjacency relationships.

4.2 Semantics of Product Model

Feature definitions are structured to separate the generic content from the non-generic content. The overall form and shape of a feature are separated into type and shape. The type of the

feature is specified by the generic type and the shape of the feature is specified by the cross-section of the feature. Class of similar features based on faces and face adjacency relationships are identified. Features, having similar types of faces and face adjacency relationships of a class, lie in that class. An instance of a class has same meaning as that of the class. Instances of such a class are created by specifying values for its parameters. A class of object is often called a family of objects, and an instance is a member of the family. A member of a class is referred to as feature/generic feature. We propose to define an ontology of form features in terms of the DIFF representation of a feature. Given any feature or construction history, the volume associated with the feature can be obtained and the DIFF representation of the feature volume captured. Once the DIFF representation of a feature is available, matching entities with the

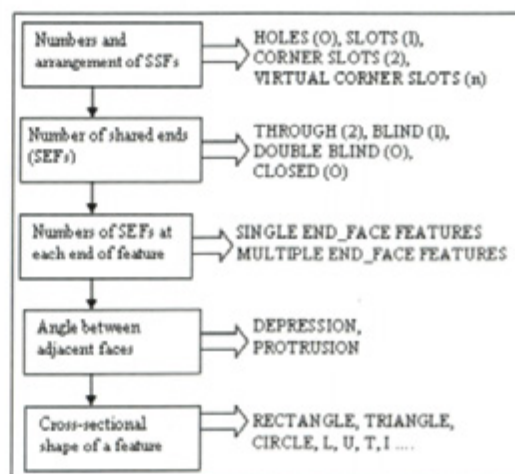


Fig. 6. Hierarchy of classification criteria in feature definition [9]

same DIFF representation can be searched to find the label or construction of interest.

The ontology of features in the present approach is implemented using Protégé ontology editor [13]. Screen shot of protégé editor for DIFF model is shown in Figure 8 in the next section. Our objective is to enumerate the generic form features, the generic and non-generic content of form feature are separated as "type" and "shape". For example, in circular through-hole feature, generic aspects are two shared end-faces, and concave angle between adjacent CSFs. The non-generic content is the circular cross-sectional shape of the cylindrical created-face. Hierarchy of classification criteria in feature definition is depicted in Figure 6.

4.3 Features Classification and Feature Taxonomy

A feature can be defined as a set of faces with adjacency relationship which enables association of knowledge. The four types of faces, described in DIFF model capture the form of the feature-solid and the feature creation process. Features are classified in terms of number of faces and faces adjacency relationships of the four types of faces which are characterized by the following four factors.

4.3.1 Numbers and Arrangement of SSFs

Based on this factor, features are divided into four classes. These classes are defined as follows:

Hole, zero shared-shell-faces: This case arises when the shell of the feature-solid is completely inside the base-solid. This class corresponds to features commonly referred to as holes. In the proposed taxonomy also, it is referred to as *hole*.

Slot, one shared-shell-face: This class of features results from the coincidence of a single shell-face of the feature-solid with the base-solid. This class corresponds to features commonly referred to as slots, and in our taxonomy also, it is referred to as *slot*.

Corner slot, two adjacent shared-shell-faces: This class of features results when any two adjacent shell faces of the feature-solid coincide with two adjacent faces of the base-solid. Since two faces meet at a corner we have named this class as *corner slot* in our taxonomy. The feature referred to as *step* in the literature, belongs to this class.

Virtual corner slot, Three or more adjacent shared-shell-faces: These features result from coincidence of 3 or more adjacent shell faces of feature-solid with 3 or more adjacent faces of the base-solid. Though these features are not cited in the literature as individual features, their combinations are referred to as virtual slots and virtual pockets. This class of features is named as *virtual corner slot* in the proposed taxonomy.

4.3.2 Type of End Faces

Each class (holes, slots and corner slots) is further divided into sub-classes through, blind, and double blind based on the type of the two end faces.

Double blind, zero shared end-faces: This class corresponds to the set of features that are generated such that the two ends of the feature-solid are totally inside the base-solid and hence, there are two CEFs and no SEFs. This class is referred to as *double-blind* in our taxonomy.

Blind, one shared end-face: This class of features is generated when one end of the feature-solid coincides with face(s) of base-solid and hence, there are one SEFs and one CEFs. This class is referred to as *blind* in our taxonomy.

Through, two shared end-faces: This class of features arises when both ends of the feature-solid coincide with the face(s) of the base-solid and hence, there are two SEFs and no CEFs. This class is referred to as *through* in our taxonomy.

Closed, no ends: When the feature-solid is a result of sweep about a closed path, such as toroid, there are no ends. There is no SEFs and no CEFs. Features of this class are referred to as *closed* features in the proposed taxonomy.

The combination of the above two steps of classification results in generic types of features such as through hole, blind slot, double blind corner slot, etc... The variation in the number of SEFs at one coinciding end is broadly classified into single-shared-end-face (SSEF) corresponding to single SEF and multiple-shared-end-face (MSEF) corresponding to more than one SEF.

4.3.3 Cross-sectional Shape of a Feature Based on Numbers of CSFs and SSFs

Feature definitions are structured to separate the generic content from the non-generic content. The overall form and shape of a feature are

separated into type and shape. The type of the feature is specified by the generic type and the shape of a feature is the cross-sectional shape of the CSFs and SSFs. Some of the common shapes are rectangle, triangle, circle, L, U, T and I.

4.3.4 Type of Angle Between Adjacent Faces

Each class is further divided into sub-classes, depression and protrusion based on the angle between adjacent faces of a feature.

Depression; This class of feature has angle between two adjacent CSFs or adjacent CEFs and CSFs as concave.

Protrusion; This class of feature has angle between two adjacent CSFs or adjacent CEFs and CSFs as convex.

If CSFs are more than one then angle between adjacent CSFs is sufficient to answer whether a feature is protrusion or depression. If CSFs is equal to one then angle between adjacent CEFs and CSFs is required to answer protrusion or depression feature.

5 ONTOLOGY FOR DIFF MODEL

The structure defined above is used to develop ontology of features. Protégé editor [13] is used to develop ontology for DIFF (domain independent form

feature) model with semantics. A high level view of the ontology is shown in Figure 7.

All features are classified in terms of the criteria described in section 4 and Figure 6. Figure 8 shows the class structure for the generic feature type (marked in the left panel). Some instances of the generic feature type are shown in the middle panel. The attributes that are associated with each feature instance and used in the reasoning are shown in the right panel. DIFF feature "Through Slot" (marked in the middle panel) with attributes' value (in the right panel) are also described in Figure 8.

The construction history associated with the feature refers to the possible ways the feature can be modeled or constructed. The user defined feature subclass is a place holder for features with different labels and also for further extensions of the feature ontology to handle features that are not either described or shape based. Figure 9 shows the instances of user defined features (marked in the figure) such as those used in a product model as shown in Figure 10 (marked in the figure). A user defined feature is stored as new feature if the feature is different from the DIFF model. The DIFF structure can be obtained for such features as shown in Figure 11.

A user defined feature "Boss-revolve4" is not there in the DIFF model. The feature "Boss-revolve4" is stored as new feature as well (marked

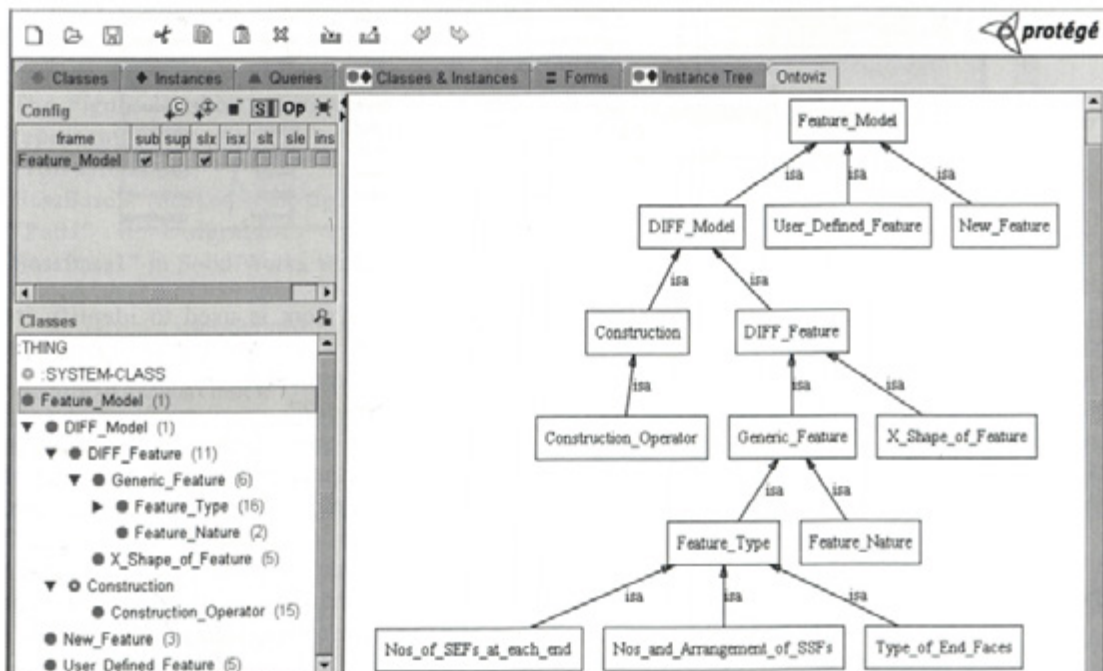


Fig. 7. Structure of DIFF model and developed ontology

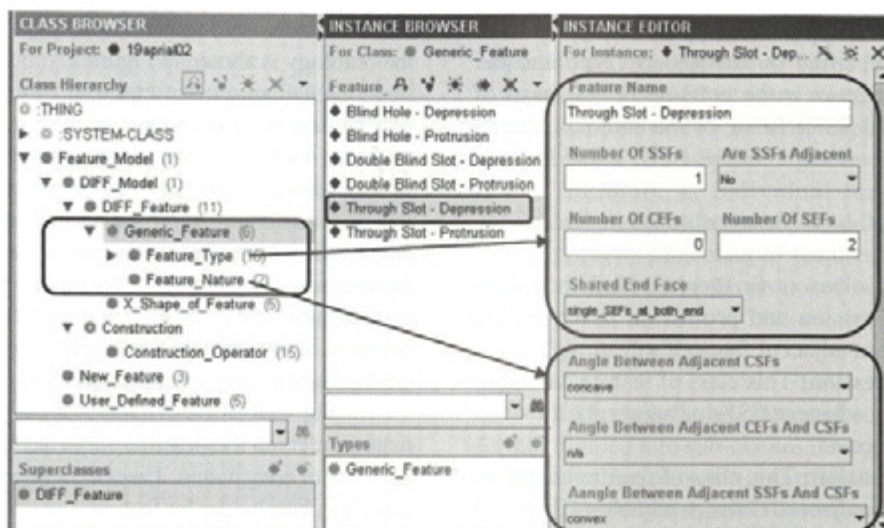


Fig.8. Structure of DIFF model with classes, slots and feature instances

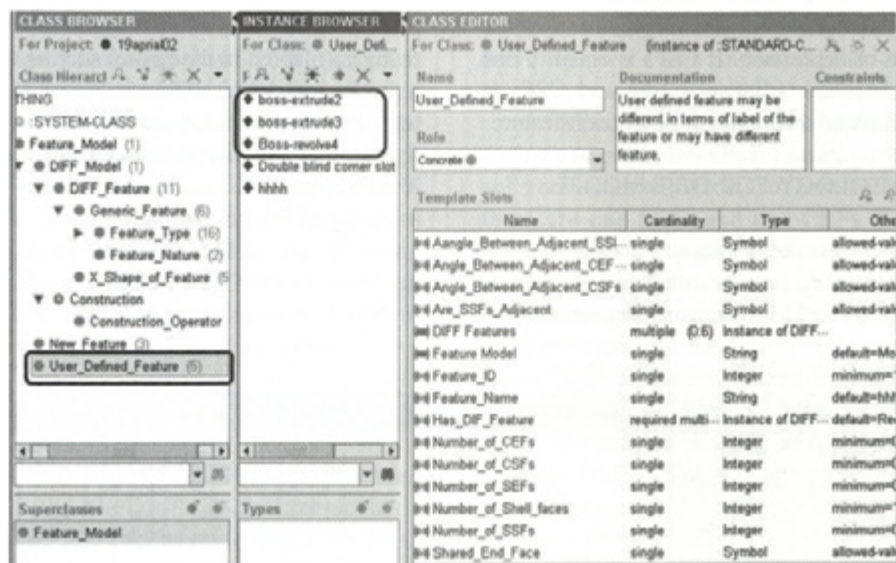


Fig. 9. Instances of user defined feature

in the figure). This feature has the same DIFF representation as the features "Through Corner Slot" and "Protrusion" (see screen shot in Fig. 11). Mismatches in feature labels between different applications are described in the next section. Mismatches in presentation /construction history are also described in the following sections.

5.1 Handling Different Labels Referring to Same Shape

Given a feature from a host system (Fig. 10), the feature volume corresponding to each feature

in the source system is used to identify its corresponding DIFF structure. Figure 12 shows the DIFF structure identified for one such feature say label "boss-extrude2".

Using the query feature in the ontology editor, the label in the source system is first matched with the label (feature name) in the DIFF structure. The query for user defined feature "boss-extrude2" is depicted in Figure 13 which is used to find the same feature in the ontology (marked in the left panel). The feature corresponding to boss-extrude2 is present in the DIFF model with different label namely, "Rectangular double blind slot -

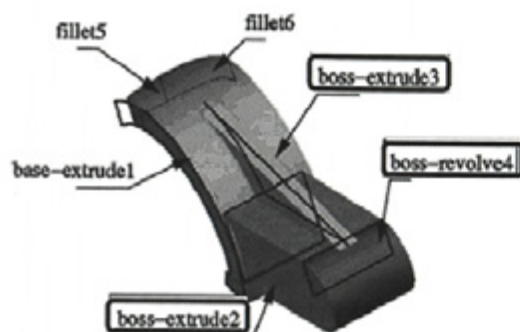


Fig. 10. Example of user defined features [11]

protrusion (boss)” (marked in the right panel). The other labels associated with this DIFF feature are now searched to check if there is a match with the target system.

5.2 Handling Features with Different Construction History/Representation

As mentioned earlier, the construction history or representation of each DIFF feature is stored in the DIFF structure. Figure 14 shows the different construction possibilities for a particular feature. Let us take a construction method “Sweep_Blind_Hole_Protrusion” in DIFF model. The Figure 14 shows different constructions/representations for “Sweep_Blind_Hole_Protrusion” as “Pad1” and “Extruded_BossBase1” (marked in the right panel). We know “Pad1” in Unigraphics and “Extruded_BossBase1” in Solid Works which are equivalent to each other.

Given a user-defined feature for which the matching features in another system have been identified, the next task is to resolve any mis-match in the construction process/representation of the feature. First it is checked if the target system supports any of the construction history associated with the DIFF feature corresponding to the feature being exchanged. Otherwise, for the different construction methods available in the target system, the DIFF representation is obtained and used to match with the feature being exchanged. Figure 14 shows the output for a query for other construction methods for a given feature.

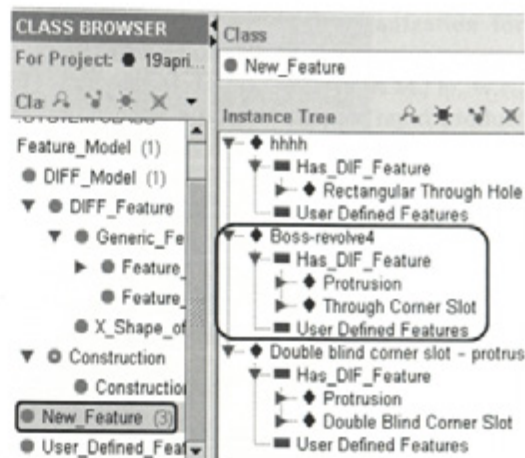


Fig. 11. Instances of new feature in the ontology

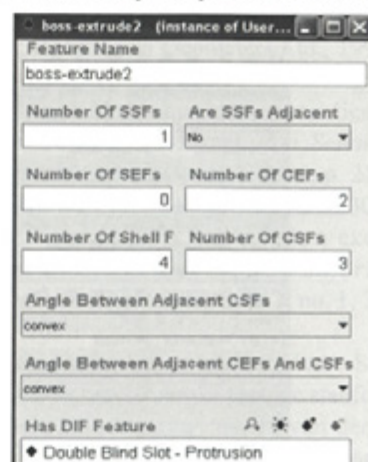


Fig. 12. DIF Feature information extracted for user defined feature label “boss-extrude2”

6 DISCUSSIONS

Using a single product model across the product lifecycle has been suggested to maintain integrity of the data which avoids the effort in creating multiple models. Product model is created only once in any modeling software. The same product model can be used for further manipulations and editions throughout the product lifecycle and can also be used among different vendors to share knowledge.

We have identified different types of semantic interoperability problems arising during exchange of product models in product development. These are: different terms referring to same shape, associated representation for a shape

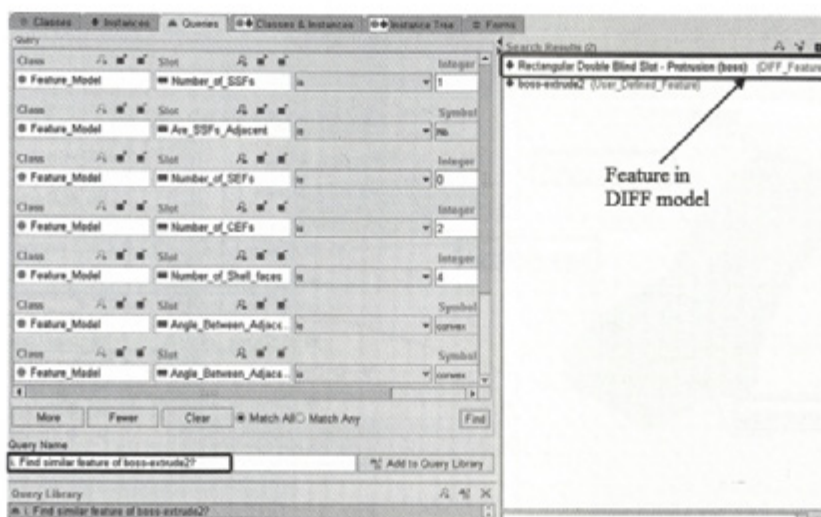


Fig. 13. Query to find similar feature of boss-extrude2

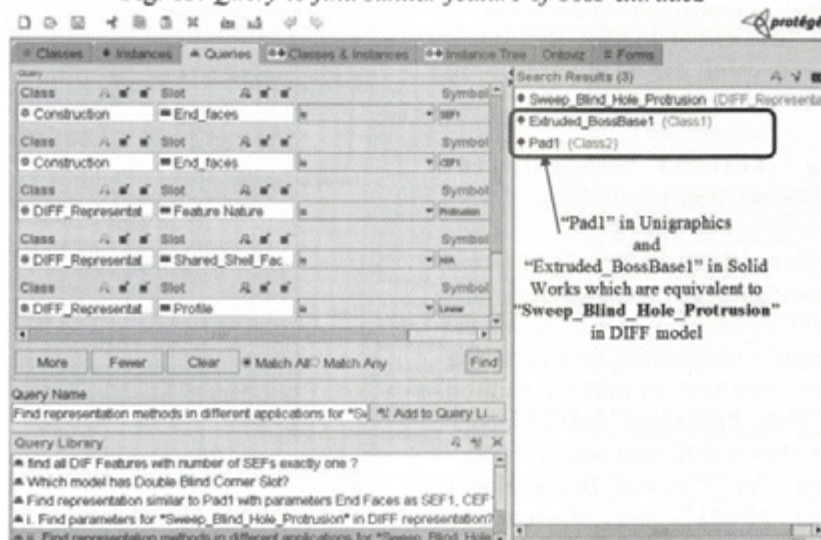


Fig.14. Query "Find representation methods in different applications for "Sweep_Blind_Hole_Protrusion" in DIFF representation?"

may be different, meaning of terms are context dependent and term with meaning is there in one domain may not be there in other domain.

Once an ontology for a DIFF model for a product model (for any source system) is developed then the features and construction history for any target system can be obtained. There is no need to enumerate separate feature and construction history for a new system. The features and construction history for a product model can be obtained through DIFF model.

We have presented a one-to-many framework for exchange of product information

model, product semantics in particular as semantics associated with shape data can convey design intent, inter-relationships between entities in the shape and other data important for downstream applications such as manufacturing. Feature based product data exchange has been used as features (means geometric data at higher levels of abstraction) are capable of carrying constraints, parameters and application attributes.

DIFF structure is described and ontology is developed that captures the vocabulary used in feature models. A prototype of the ontology and the reasoning on the ontology has been built using

the Protégé ontology editor. The method is demonstrated to handle mismatches in labels and construction history using Protégé ontology editor.

7 CONCLUSIONS

A new feature based ontology has been proposed to address the problem of semantic interoperability between shape models. In contrast to present art, the proposed ontology enables reasoning to handle situations where equivalence between terms is not already captured in the existing ontology. A prototype implementation that is able to handle mismatches in labels and construction history has been described. Handling other mismatches and incorporation of the feature model and ontology in the core product model [3] has been identified as future work.

8 REFERENCES

- [1] Ameri, F., Dutta, D. Product lifecycle management: closing the knowledge loops. *Computer-Aided Design & Applications*, vol. 2, no. 5, 2005, p. 577-590.
- [2] Brunetti, G., Grimm, S. Feature ontologies for the explicit representation of shape semantics. *International Journal of Computer Applications in Technology*, vol. 23, no.2, 2004, p. 192-202.
- [3] Fenves, S., Fofou, S., Bock, C., Bouillon, N., Sriram, R. D. *CPM2: A revised core product model for representing design information*. National Institute of Standards and Technology, NISTIR 7185, Gaithersburg, MD 20899, USA, 2004.
- [4] Heiler, S. Semantic interoperability. *ACM Computing Surveys*, vol. 27, no. 2, 1995, p. 271-273.
- [5] Horváth, I., Pulles, J., Bremer, A. P., Vergeest, J. S. M. Towards an ontology-based definition of design features. *Proceedings of the SIAM Workshop on Mathematical Foundations for Features in Computer Aided Design, Engineering and Manufacturing*, October 22-23, 1998, Troy, Michigan USA.
- [6] ISO 10303. *ISO/CD 10303-Part III: Product data representation and exchange: Integrated application resource: Construction history*

- features*. International Organization for Standardization, 2004.
- [7] Lee, C.K.M., Lau H.C.W., Yu, K.M., Ip, W.H. A generic model to support rapid product development: an XML schema approach. *International Journal of Product Development*, vol. 1, no. 3/4, 2005, p. 323-340.
- [8] Mostefai, S., Bouras, A., Batouche, M. Effective collaboration in product development via a common sharable ontology. *International Journal of Computer Intelligence*, vol. 2, no. 4, 2005, p. 206-212.
- [9] Nalluri, S.R.P.R. *Form feature generating model for feature technology*. PhD thesis, Indian Institute of Science, Department of Mechanical Engineering, Bangalore, India, 1994.
- [10] Owen, J. *STEP: An introduction*. Winchester, UK: Information Geometers Ltd., 1993.
- [11] Patil, L., Dutta, D., Sriram, R. Ontology-based exchange of product data semantics. *IEEE Transactions on Automation Science and Engineering*, vol. 2, no. 3, 2005, p. 213-225.
- [12] Pratt, M. J. Introduction to ISO 10303 - the STEP standard for product data exchange. *Journal of Computing and Information Science in Engineering*, vol. 1, no. 1, 2001, p. 102-103.
- [13] Protégé. *Protégé ontology editor*. Stanford University, 2007, <http://protege.stanford.edu/>
- [14] Pulles, J. P. W., Horváth, I., van der Vegte. Beyond features: an ontology oriented Interpretation. *Proceedings of the International conference on engineering design (ICED 99)*, August 24-26, 1999, Munich, p. 1761 - 1764.
- [15] Subramani S, Nalluri, S.R.P.R., Gurumoorthy B. 3D clipping algorithm for feature mapping across domains. *Computer Aided Design*, vol. 36, no.8, 2004, p. 701-721.
- [16] Subramani, S. *Feature mapping, associativity and exchange for feature-based product modeling*. PhD thesis, Indian Institute of Science, Department of Mechanical Engineering, Bangalore, India, 2005.
- [17] Szykman, S., Fenves, S.J., Keirouz, W., Shooter, S.B. A foundation for interoperability in next-generation product development systems. *Computer Aided Design*, vol. 33, no.7, 2001, p. 545-559.