

DETC2008-49671

FEATURE MAPPING AUTOMATION FOR CAD DATA EXCHANGE

Leen Hanayneh
Yiwen Wang
Yan Wang, Ph.D.
Department of Industrial Engineering
& Management Systems
University of Central Florida
Orlando, FL 32816

Jack C. Wileden, Ph.D.
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Khurshid A. Qureshi, Ph.D.
Knowledge & Process Integration
PD Systems, IT
Ford Motor Company
Dearborn, MI 48124

ABSTRACT

Computer-aided design (CAD) data interoperability is one of the most important issues to enable information integration and sharing in a collaborative engineering environment. A significant amount of work has been done on the extension and standardization of neutral data formats in both academy and industry. In this paper, we present a feature mapping mechanism to allow for automatic feature information exchange. A hybrid semantic feature model is used to represent implicit and explicit features. A graph-based feature isomorphism algorithm is developed to support feature mapping between different CAD data formats.

1. INTRODUCTION

One of the major cost elements associated with product design is the precious time engineers spend on data translation between different CAD formats, data integration between different systems, routine data reprocessing in different application scenarios and redesign due to loss of information, as well as error correction due to human errors in the above processes. Complementary to data exchange standards such as the Standard for the Exchange of Product Model Data (STEP), design process automation can reduce time and cost in a virtual collaborative engineering environment. Interoperable CAD model generation, intent and knowledge capturing, and semantic-level information exchange are critical to enable the automation.

Software companies usually take a static identify-and-map approach to improve feature-level interoperability between CAD systems by understanding and using the available APIs from the systems, which are generally ad-hoc, low-level, narrowly targeted, difficult to use, and prone to produce erroneous interactions. The majority of the work by academic researchers focuses on interoperable feature modeling. Our objective is to enhance the current interoperable data modeling

approaches with research on the possible way of data exchange process automation.

In this paper, a hybrid semantic feature modeling approach is taken to simultaneously represent implicit and explicit features. This approach provides a neutral feature semantics format which represents design history independent feature information. Feature definitions and interrelationships can be modeled by graph structures. We further develop a method for finding graphs similarity which allows us to dynamically map feature information from source systems to the neutral semantic feature model and then transform to the target systems at run-time.

The advantages of the dynamic mapping approach based on hybrid semantic feature models are listed as follows.

- a. Compared to the static identify-and-map approach, the dynamic mapping reduces the complexity of searching and mapping processes for multiple systems. The time used in this process has a linear relationship with the number of systems whereas the traditional one-by-one translation process has a factorial complexity and is difficult to manage as the number of systems grows.
- b. A multi-resolution approach is taken in hybrid semantic feature modeling to record design intent independent of design history. Model construction information can be abstracted at multiple levels. This approach is to support extensible feature modeling. When system-specific feature definitions are changed at the low level, the high-level model is not affected.
- c. The similarity-based dynamic mapping enables real-time feature model matching. Without prior definitions, new feature data can be mapped in an automatic way. A document-driven design (DDD) mechanism [1, 2] is taken to automate the process of feature generation and mapping. The DDD mechanism is developed to support history-neutral modeling in CAD systems, by which the

construction of features is independent of design history. Every feature can exist in a separate document. Creating a new feature or modifying an existing feature can be fulfilled by only adding a new document or modifying the relevant document.

- d. The DDD approach also increases the flexibility of model exchange. Separate documents can be used to capture individual features. As a result, a fat-server-thin-client approach can be adopted in a distributed design environment. The server may be used to keep the document repository, preview 3D models in neutral feature format, and translate between the neutral file format and the target format. APIs can be installed on the client side to capture specific features.

In the remainder of the paper, Section 2 gives a brief review of recent research on feature modeling for interoperable data exchange. An introduction of graph matching algorithms is also given. Section 3 describes our new hybrid semantic feature model representations. Section 4 presents the feature matching algorithm to enable the dynamic mapping process.

2. BACKGROUND

2.1 Feature Modeling for Interoperable Data Exchange

There are plenty of research efforts on form feature representation [3-6]. In ASU Features Testbed Modeler [7-9], features are defined in terms of parameters and rules about geometric shape. Interaction between features includes spatial relationship, volume-based CSG tree and Boolean operations. E-Rep [10-13] distinguishes generated features, datum features, and modifying features and regards a CAD model as being built entirely by a sequence of feature insertion, modification, and deletion description.

Several user-defined feature representation methods were proposed. Shah et al. [14] presented a declarative approach using geometric entities and algebraic constraints. Middleditch and Reade [15] proposed a hierarchical structure for feature composition and emphasized the construct relationship of features. Hoffmann and Joan-Arinyo [16] define user-defined features by standard features, constraints, and attributes procedurally. Bidarra et al. [17] include validity constraints in the specification of user-defined features. Wang and Nnaji [18] model intentional feature and geometric feature independently and embedded with parametric constraints.

Based on current framework of STEP standards, the ENGEN Data Model (EDM) [19, 20] extended STEP's current explicit entity representation by adding some predefined local features such as round and chamfer in a bottom-up approach. PDES's Form Feature Information Model (FFIM) [21, 22] adopted a dual representation of explicit and implicit features. Explicit features are represented generally by face lists, while implicit features are categorized into depression, protrusion, passage, deformation, transition, and area features.

Some researchers used a hybrid CSG/B-Rep structure. Roy and Liu [23] constructed CSG using form primitives and form features. A face-edge type data structure is used at the low-level B-Rep. These two data structures are linked by reference faces. Wang and Ozsoy [24] used primitive features and form features to build the CSG structure. Dimension and orientation information are represented as constraint nodes in the CSG tree. A face-edge type data structure is used for lower level entities. The connection between two structures is built by pointers from set operator nodes in CSG to B-Rep data structures and from faces to feature faces. Gomes and Teixeira [25] also developed a CSG/B-Rep scheme, in which CSG represents the high-level relationships between features, and the B-Rep model describes the details. An additional Feature Topological Structure in parallel with the B-Rep model defines volume form features.

Besides the above models that concentrate on hybrid representation and integration, another approach is to capture how an engineer uses 3D CAD software to create models. Choi et al. [26] proposed a macro parametric approach to provide capabilities to exchange parametric information. This approach is to translate the macro recorded in the source system to a neutral macro format, and then translate it to the macro of the target system. Li et al. [27] established a real-time collaborative design environment based on some neutral modeling commands. APIs of the source and target systems were used to record design history by system modeling operations.

Different from the above methods, we would like to improve interoperability based on feature semantics that support history independent constructions. A multi-resolution feature representation approach is taken. Dynamic feature mapping between domains is also proposed. This automated mapping process is based on graph isomorphism and similarity.

2.2 Graph Matching

A graph $G = (V, E)$ in its basic form is composed of vertices V and edges E . V is the set of vertices and $E = V \times V$ is the set of edges in graph G . The order of a graph G is defined as the number of vertices of G as $|V|$ and the number of edges as $|E|$. If two vertices in G , $u, v \in V$, are connected by an edge $e \in E$, then the two vertices are said to be adjacent or neighbors. Edges are said to be undirected when they have no direction, and a graph G containing only such types of edges is called undirected. When all edges have direction and therefore (u, v) and (v, u) can be distinguished, the graph is said to be directed.

Vertices and edges can also be labeled providing more information about a graph. A graph with such information is called a labeled graph. Moreover, the vertices and edges can have attributes giving an attributed graph.

Graph matching involves matching a graph with another graph or subgraph by mapping vertices and edges. When two graphs are structurally identical, i.e. every node and edge in one graph is only mapped once, they are said to be isomorphic.

When two graphs are not isomorphs, the interest will be in finding the most similar subgraphs.

The graph isomorphism is an interesting problem from a computational complexity viewpoint. It is clearly in NP but is not known to be in P or to be NP-complete. Many algorithms have been developed to implement graph isomorphism. The most popular ones are summarized as follows.

The Ullmann algorithm [28] is a backtrack algorithm. It is devised for both graph and subgraph isomorphism. It is still today one of the most commonly used algorithms for exact one-to-one graph matching because it significantly reduces the size of the search space. It reduces the number of candidate vertices that should be searched. The algorithm searches both graphs at the same time.

The NAUTY algorithm [29] is also a backtrack algorithm. It canonically labels a graph before checking for isomorphism. NAUTY algorithm is among the fastest ones. It takes a polynomial time for most of the cases except for some rare ones where it employs exponential time. It is only used for graph isomorphism but not subgraph isomorphism.

The VF algorithm [30] takes a bottom up approach. The algorithm is based on a depth-first search strategy and is suitable for large graphs with reduced memory requirements. The VF2 algorithm [31] is an improved version of the VF algorithm. This improved version is more efficient in terms of time and spatial complexity. It is fast and requires $O(n)$ space where n is the number of vertices in a graph. It can be used for both graph and subgraph isomorphism.

The Schmidt and Druffel algorithm [32] uses the information contained in the distance matrix representation of a graph to establish an initial partition of the graph vertices. This distance matrix information is then used in a backtracking procedure to reduce the search tree of possible mappings.

All the above algorithms are deterministic. In contrast, the algorithm developed by Corneil and Gottlieb [33] is non-deterministic, and it is applied to non-directed graphs.

Since finding the isomorphism between two graphs is an NP-hard problem, heuristic similarity measures have been implemented to compute the similarity between two graphs. Similarity can be computed by either cost based measures or feature based measures. In cost based (tree edit) approaches, some functions are used to determine the minimum cost to transform one graph to the other. In the feature based approaches, the similarity is calculated by extracting a set of structural features and comparing them.

Many algorithms have been implemented on tree and graph similarity. Zhang et al. [34] designed an algorithm that uses the number of simple edit (e.g. insert and delete) operations needed to transform one tree into the other to calculate the similarity. Bhavsar et al. [35] developed a weighted tree similarity algorithm, which computes the similarity of a subtree in a recursive way, and it has the option of tailoring the similarity measures based on the subjective preferences. The similarity flooding algorithm [36] matches graphs of two data schemas or data instances. It calculates similarity by first

finding an initial mapping and then iterating the result until a stable mapping is reached.

In general, we can represent features in CAD models as directed and labeled graphs. Our effort here is focused on how to map the features between two different systems to allow interoperable data exchange. Since the definitions of features vary in different CAD systems, it is highly likely that the same feature is represented by different graph structures. To map a feature from one graph structure to another, we need to take a subgraph isomorphism approach to determine the degree of similarity between the two graphs depending on both the structure and the label.

3. HYBRID SEMANTIC FEATURE MODELING

We propose a hybrid semantic feature model to represent implicit and explicit features in order to enable history-neutral feature construction. It is to capture feature semantics in a multi-resolution fashion and support dynamic feature mapping.

3.1 Hybrid Semantic Feature Model

Our hybrid semantic feature model focuses not only on how to describe features, but also on how to represent the relationship between features. We use multi-resolution graph structure to represent the model.

The properties of our hybrid model include:

1. It is flexible by using multi-resolution graph structure.
2. It is a comprehensive description of the relationship between features and their attributes.
3. It is designed to be used in document-driven design processes.
4. It can be used for history independent design processes.

A feature graph is a directed, labeled and attributed graph used to record design intent. Our model is associated with eight attributes, and we classify them as *individual*, *interfacial* or *alias*. The *individual attributes* are used to define implicit features only, such as Parameter, Sketch and Boolean Sign in the design process of a specific feature. On the other hand, the *interfacial attributes* are supplied to define the boundaries of features. They are used to capture the relationships between features. Four interfacial attributes are Point (P), Line (L), Surface(S), and Solid Body (B).

The same feature may have a different name in two different CAD systems. For example, to create a long object of a fixed cross-sectional profile is referred to as “*Extruded Boss/Base*” in Solidworks® and “*Protrusion*” in SolidEdge®. In order to compensate for this fact, an eighth attribute called *alias* is added to every feature. This attribute adds more information and facilitates the mapping process, which will be discussed in Section 3.2.

Figure 1 illustrates how a feature *Extrude1* is represented as a feature graph. All of these attributes are represented by eight different shapes which are only used to make the graph easier to read. The center node of the graph is the feature node. The direction of an edge denotes *reference* or *inclusion*

dependency between two nodes, which also differentiates explicit attributes from implicit attributes. *Explicit* attributes are used to create a feature whereas *implicit* attributes are to serve as references for other dependent features. For example, *Extrude1* in Figure 1 was created by the extrusion of a sketch in a plane, so its explicit attributes are Surface (Front) as the datum plane, Sketch (Sketch Graph), Parameter (Extrude Depth) and Plus Boolean Sign. At the same time, the implicit attributes of *Extrude1* including Point (P11), Body (B1), Surface (S11), and Line (L11) are used to form other features and they form a direct connection between *Extrude1* and other features.

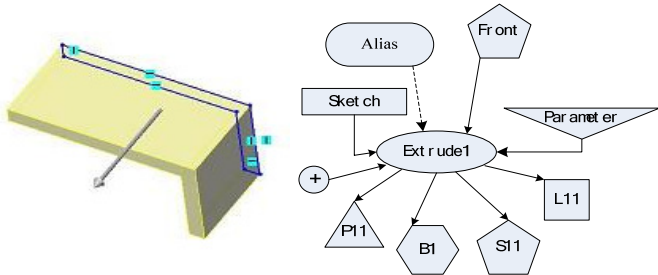


Figure 1: A feature graph example of extrusion

Based on the interrelationship between features in a 3D CAD model, the hybrid semantic model is defined as the assembly of all its features. It provides a comprehensive description of the model with a three-level multi-resolution graph. Figure 2 shows an example of *anchor joint* model, which is created by eight features and two reference planes in Solidworks®. This example will be used to explain the principle of our multi-resolution feature graph.

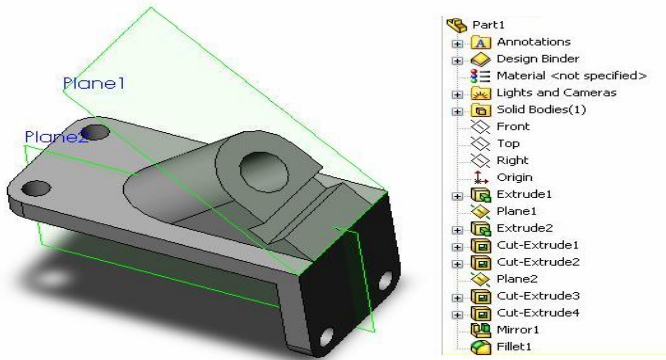


Figure 2: An example of anchor joint model

In a feature graph, we define three different levels. It is the backbone of a hybrid semantic model. It includes the feature dependency information and reflects the connection between features without specifying the attributes of each feature.

At Level 1, a feature graph captures the basic feature relationships. As shown in Figure 3, *Extrude1* is the base part of the model in Figure 2. *Extrude2*, *Cut-Extrude3*, *Cut-Extrude4* and *Fillet1* are created based on the interfacial attributes of *Extrude1*. *Cut-Extrude1* and *Cut-Extrude2* are

created based on *Extrude2*. *Mirror* is created from *Cut-Extrude3* and *Cut-Extrude4*.

Closely related features can also be grouped and form composite features at Level 1. For example, *Cut-Extrude1* and *Cut-Extrude2* are both created on *Extrude2* forming *Group1* as illustrated in Figure 3. The grouping further reduces the complexity of graphs.

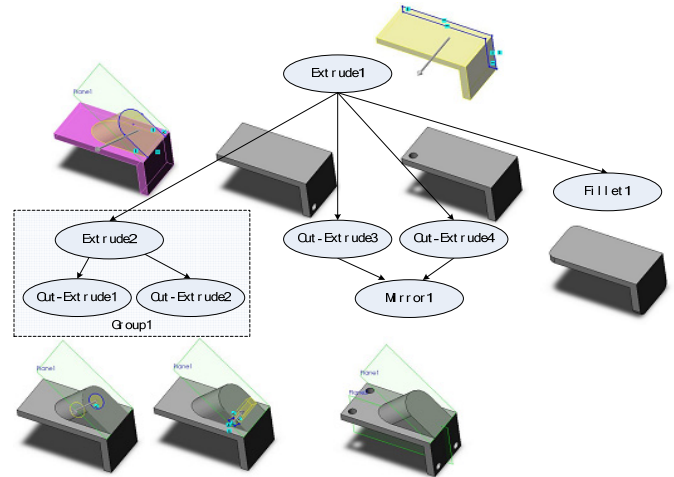


Figure 3: Feature graph expanded to level 1

A Level 2 graph expands the corresponding Level 1 graph by including more information of how features are geometrically related by interfacial attributes. For example, Figure 4 shows the Level 2 expansion of several features from the Level 1 graph in Figure 3. The corresponding feature definitions of this example along with interfacial features in the 3D model are shown in Figure 5. The interfacial attributes that connect two features occur as implicit attributes in one feature and explicit in the other feature. Here, *S12* is an implicit attribute of *Extrude1* and is an explicit attribute of *Cut-Extrude3*. In this model, *Extrude1* uses a datum plane as its sketch plane. Each of the two features *Cut-Extrude3* and *Cut-Extrude4* is created based on two planes in *Extrude1*. The *mirror* feature is based on *plane2*, which in turn is defined by three points (*P11*, *P12*, and *P13*) in *Extrude1*.

In Level 3 graphs, the individual attributes of features are added to include the complete feature information. As shown in Figure 4, the Sketch, Parameter of cut depth and Boolean Minus sign were reflected in *Cut-Extrude1*, as well as the Boolean sign and sketch were reflected in *Cut-Extrude3*. The Level 3 graph includes full design information by representing the dependency between the features as well as all of their attributes.

The three levels of feature graphs can provide sufficient details to help in the dynamic feature mapping process. Theoretically the mapping can be performed at any level between feature graphs. For the feature creation purpose, we usually need Level 3 graphs with complete information of definitions. The next section explains the algorithm used for the mapping process.

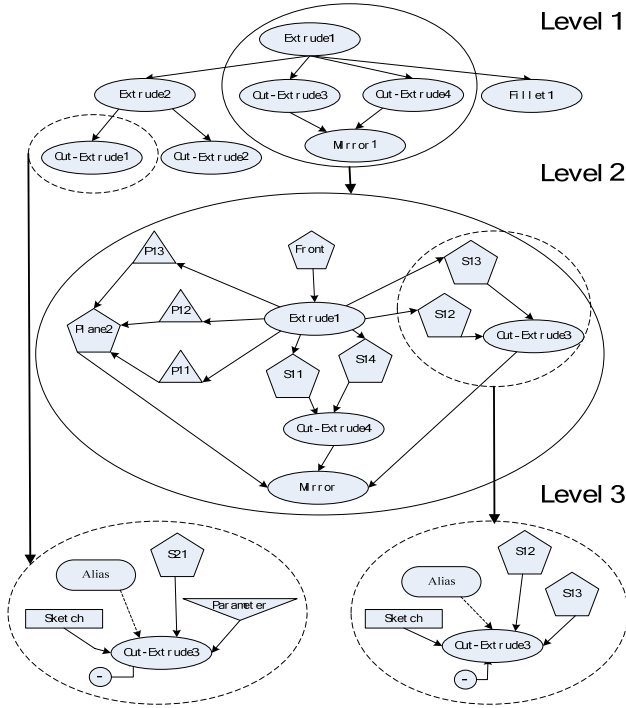


Figure 4: Feature graph expanded to level 2 and level 3

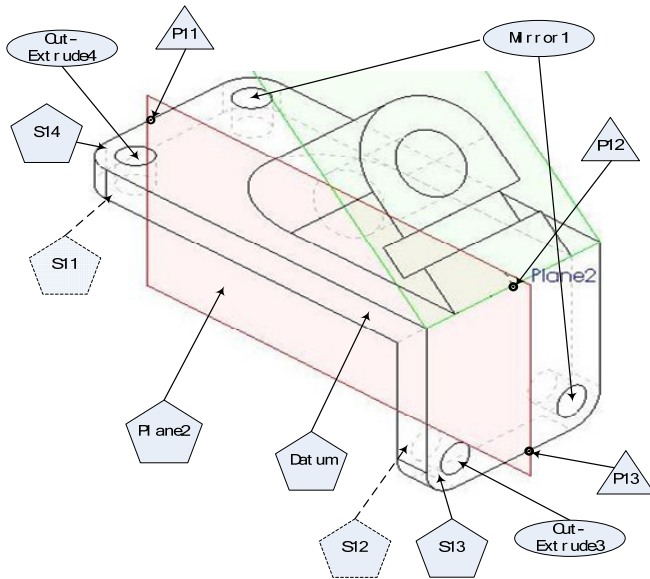


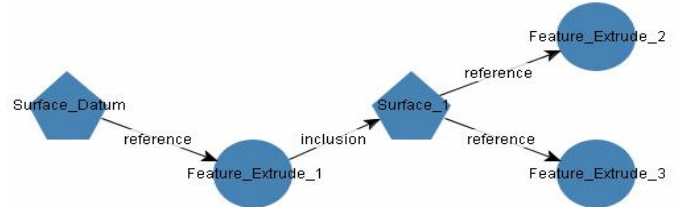
Figure 5: Geometric information of Level 2

Compared to other interoperable feature modeling approaches in Section 2.1, our method not only describes a model by the features it contains, but also defines the connection and relationships between features. The model is represented as a multi-resolution graph structure, which

provides different levels of design intent for the mapping procedure.

3.2 Dynamic Mapping

A semantic model usually represents relationships as *subject-predicate-object* triples. The semantic similarity measures play an important role in information retrieval. To model and represent the hybrid semantic features in a way that facilitates the processing and mapping, we use the Resource Description Framework (RDF) [37] to document features. RDF is built on top of the XML syntax which enables syntax-level interoperability. RDF describes graphs of statements about resources. The resources in our feature model are labeled nodes, each corresponding to a feature which is further decomposed into a subtree. The matching process is then performed on the feature subtrees. Figure 6 shows a simple example of three features connected at Level 2 of the hybrid semantic model and its corresponding RDF document.



```

<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [
  <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <!ENTITY kb 'http://protege.stanford.edu/kb#'>
  <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
]
<rdf:RDF xmlns:rdf="&rdf;" xmlns:kb="&kb;" xmlns:rdfs="&rdfs;">
<kb:Surface rdf:about="&kb;Surface_DatumPlane"
  rdfs:label="Surface_DatumPlane"/>
<kb:Surface rdf:about="&kb;Surface_1"
  rdfs:label="Surface_1"/>
<kb:Sketch rdf:about="&kb;Sketch_1"
  rdfs:label="Sketch_1"/>
<kb:Sketch rdf:about="&kb;Sketch_2"
  rdfs:label="Sketch_2"/>
<kb:Sketch rdf:about="&kb;Sketch_3"
  rdfs:label="Sketch_3"/>
<kb:Parameter rdf:about="&kb;Parameter_1"
  rdfs:label="Parameter_1"/>
<kb:Parameter rdf:about="&kb;Parameter_2"
  rdfs:label="Parameter_2"/>
<kb:Parameter rdf:about="&kb;Parameter_3"
  rdfs:label="Parameter_3"/>
<kb:Feature_Extrude rdf:about="&kb;Feature_Extrude_1"
  rdfs:label="Feature_Extrude_1">
  <kb:reference rdf:resource="&kb;Parameter_1"/>
  <kb:reference rdf:resource="&kb;Sketch_1"/>
  <kb:reference rdf:resource="&kb;Surface_DatumPlane"/>
</kb:Feature_Extrude>
<kb:Feature_Extrude rdf:about="&kb;Feature_Extrude_2"
  rdfs:label="Feature_Extrude_2">
  <kb:reference rdf:resource="&kb;Surface_1"/>
  <kb:reference rdf:resource="&kb;Parameter_2"/>
  <kb:reference rdf:resource="&kb;Sketch_2"/>
</kb:Feature_Extrude>
<kb:Feature_Extrude rdf:about="&kb;Feature_Extrude_3"
  rdfs:label="Feature_Extrude_3">
  <kb:reference rdf:resource="&kb;Parameter_3"/>
  <kb:reference rdf:resource="&kb;Sketch_3"/>
  <kb:reference rdf:resource="&kb;Surface_1"/>
</kb:Feature_Extrude>
</rdf:RDF>

```

Figure 6: RDF example for a feature graph

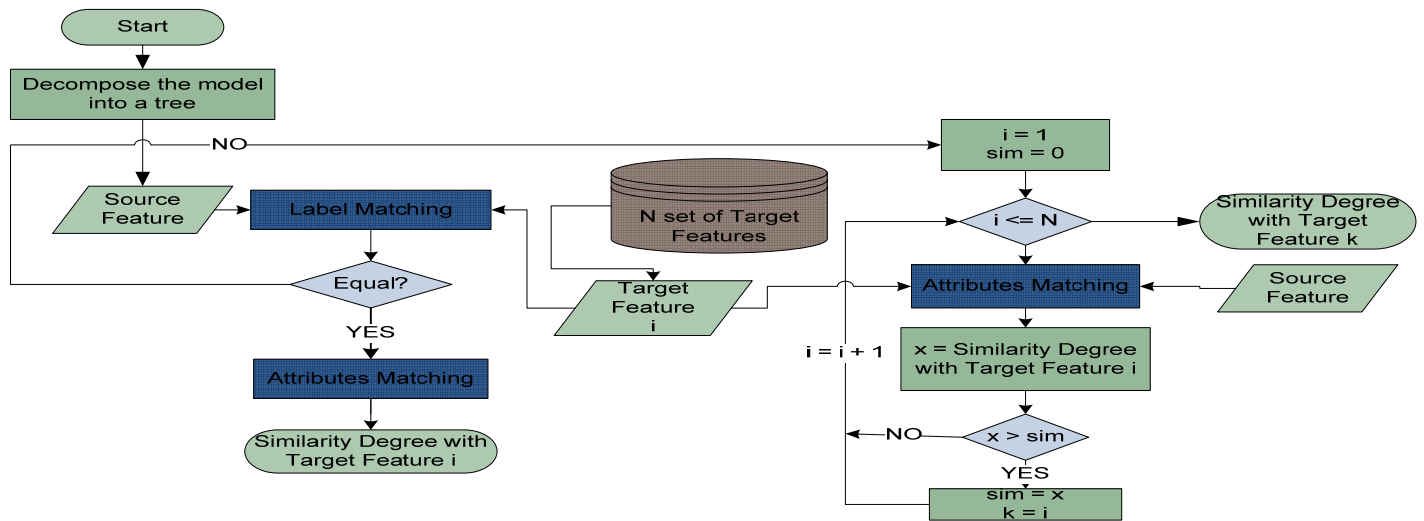


Figure 7: The mapping algorithm flowchart

A mapping algorithm for feature matching is developed. The algorithm follows a feature based matching approach to calculate the similarity. It compares two feature graphs by their labels and structures. Figure 7 is a flowchart for the algorithm used to determine the degree of semantic similarity. Two similarity measures are considered through two stages: Label Matching and Attributes Matching. Label Matching simply compares the names of the two features as well as their *alias* names and returns if they are equal or not. In the Attributes Matching, the number of implicit attributes of the feature is counted and the type of the implicit attributes is then listed. Attributes similarity is then measured according to the number of the same type of attributes that are exactly matched.

The dynamic mapping procedure can be summarized as follows:

- 1- Decompose the model into a tree T_f , where each node corresponds to a feature
- 2- Starting with the root of T_f as the Source Feature
- 3- Search for a feature in the set of target features to match the label of the Source Feature
 - If Found then
 - Compare the attributes of the two features and map the source feature to the target feature
 - Else
 - a- Search for a feature in the set of target features with the same number and type of attributes
 - b- Determine similarity degree depending on the label matching and the attributes matching
 - c- Map the source feature to the target feature with the highest similarity degree
- 4- Traverse through the nodes of T_f to get the next Source Feature and repeat step 3

Our semantic mapping algorithm starts with Label Matching which checks if the labels are equal. If the two labels are equal then the source feature is mapped to that target

feature. Attributes Matching is used to calculate the degree of similarity between those two features. The set of target features is sorted alphabetically to reduce the complexity of the searching from $O(N)$ to $O(\log N)$. In the case when the Label Matching returns a true, Attributes Matching is applied to determine the degree of similarity. If no feature with the same label is found, then Attributes Matching is repeated for all sets of target features and the source feature is mapped to the target feature that holds the highest degree of similarity.

After a source feature gets mapped to a target feature, the target feature name will be added as an *alias* to the source feature to eliminate redundancy. If another instance of the same feature is found in the next mapping, Label Matching will find a target feature with the same name as the *alias* and thus reduce the matching complexity.

Our matching algorithm is tailored for matching graphs represented in the feature graph format. The types of attributes can only be one of the eight attributes described in section 3.1. Attributes Matching can be accomplished by comparing the attributes labels and then counting the matched attributes. This reduces the computational complexity and gives our algorithm an advantage over the matching algorithms presented in Section 2.2.

4. IMPLEMENTATION AND RESULTS

This section describes the implementation of the mapping algorithm. The mapping algorithm is developed based on Jena [38]. Jena is an open source RDF Java toolkit for building semantic Web applications, and it provides programming APIs for processing RDF and RDF Schema.

Feature definitions and data for source and target features are assumed to be in RDF/XML format in our demonstration. It is not difficult to convert native CAD data formats to RDF/XML, which has been implemented in some commercial CAD systems such as SolidWorks[®]. In our implementation, the source feature RDF and a set of target features RDF files are passed to our program to test the algorithm. The source feature

is extracted from the RDF file by searching the *subject-predicate-object* triple relationships represented in the RDF.

```
<kb:Feature_Extrude rdf:about="&kb;Feature_Extrude_1"
  rdfs:label="Feature_Extrude_1">
  <kb:reference rdf:resource="&kb;Parameter_1"/>
  <kb:reference rdf:resource="&kb;Sketch_1"/>
  <kb:reference rdf:resource="&kb;Surface_DatumPlane"/>
  <kb:inclusion rdf:resource="&kb;Surface_1"/>
</kb:Feature_Extrude>
```

Figure 8: Extract feature and its attributes from RDF

As shown in Figure 8, the feature name can be extracted from the *subject* with the API *subject.getLocalName()*, and the attributes can be found from the *object* string where the *predicate* of this relationship is the *reference*. The implicit attributes are used as references to create the feature.

After extracting the feature name with its corresponding implicit attributes, similarity can be measured from attributes. There are three following cases in the mapping procedure.

Case 1: *LabelMatching* returns *true* and *AttributesMatching* returns *SimilarityDegree = 1*.

A feature in the target set of features is found with the same name as the source feature or one of the alias names. The two features have isomorphic graphs and thus the *Similarity Degree = 1*.

Case 2: *LabelMatching* returns *true* and *AttributesMatching* returns $0 < \textit{SimilarityDegree} < 1$.

A feature in the target set of features is found with the same name of the source feature or one of the alias names. The two features are not isomorphs but have a certain degree of similarity between 0 and 1.

Case3: *LabelMatching* returns *false* and *AttributesMatching* returns $0 < \textit{SimilarityDegree} < 1$.

No feature is found in the target set of features with the same name as the source feature. A certain degree of similarity according to the attributes is computed.

The *Similarity Degree* is measured by first counting the implicit attributes of the source feature (*S_count*) and of the target feature (*T_count*). A comparison between the types of attributes is performed, and a counter (*sim_count*) is incremented with each matching attribute. Then similarity is measured as

$$\textit{SimilarityDegree} = \begin{cases} \frac{\textit{sim_count}}{\textit{S_count}} & (\textit{if } \textit{S_count} \geq \textit{T_count}) \\ \frac{\textit{sim_count}}{\textit{T_count}} & (\textit{otherwise}) \end{cases}$$

The source feature is then mapped to the target feature with the highest *Similarity Degree*.

To evaluate the algorithm, two simple models are studied: *Anchor Joint1* and *Anchor Joint2*, as shown in Figure 9. They were built in Solidworks® which is used as the source system. Each model is decomposed into a graph of features. We would like to map each feature from the source system to a corresponding one among the target features.

Table 1 shows a library of target features from *SolidEdge*® with their corresponding attributes. This set of features is used as an example to illustrate how the definition of features in the target system can be different from the definition of features in the source system.

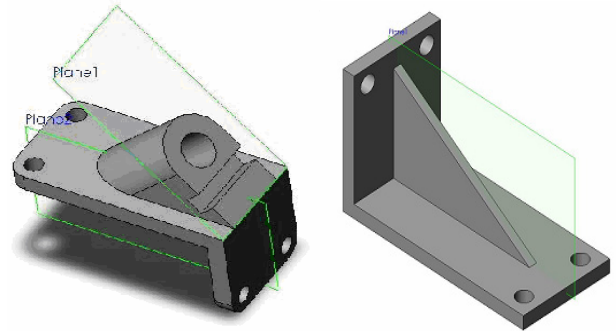


Figure 9: *Anchor Joint1* and *Anchor Joint2* in Solidworks®

Table 1: A library of target features in SolidEdge®

Feature	Attribute
Protrusion	Parameter
	Sketch
	Surface
	BooleanSign
Cutout	Parameter
	Sketch
	Surface
	BooleanSign
Revolved Protrusion	Parameter
	Sketch
	Surface
	BooleanSign
Rib	Sketch
	2*Surface
Revolved Cutout	Parameter
	Sketch
	Surface
	BooleanSign
Round	n*Line
	Parameter

Table 2: Results of the mapping

Model	Source Feature	Alias	Implicit Attributes	Label Matching	Target Feature	Similarity Degree	
Anchor Joint1	3xExtrude		Surface	False	Protrusion	1	
			Sketch				
			Parameter				
			BooleanSign				
Anchor Joint2	Cut		2xSurface	False	Cutout	0.75	
			Sketch				
			BooleanSign				
	Extrude	Protrusion		Surface	True	Protrusion	1
				Sketch			
				Parameter			
				BooleanSign			
	Rib			Surface	True	Rib	0.50
				Sketch			
				Parameter			
BooleanSign							

Each source feature is mapped separately using the algorithm described in Section 3.2. Table 2 shows the results of the mapping of the two models to the target features. For example, *Anchor Joint1* is composed of three *Extrude* features. Each *Extrude* has three implicit attributes, *Surface*, *Sketch*, and *Parameter*. The Label Matching checks if there is a feature in the set of target features with a name matched to *Extrude*. It returns *False*. Notice that in the target system the *Extrude* feature is called *Protrusion*. Then the Attributes Matching compares *Extrude* to every feature in the set of target features. *Protrusion* is chosen as the target feature since it has highest similarity degree with *Extrude*. Using the similarity degree formula, we receive $S_count = 4$, $T_count = 4$ and $sim_count = 4$, thus *Similarity Degree* = 1.

Anchor Joint2 is composed of three features, *Cut*, *Extrude*, and *Rib*. The Label Matching does not find a feature in the set of target features with the same name as *Cut* and returns *False*. The Attributes Matching then computes the similarity degree and matches it to *Cutout*. In this case, using the similarity degree formula, we receive $S_count = 4$, $T_count = 4$ and $sim_count = 3$, thus *Similarity Degree* = 0.75. *Extrude* has already been mapped to *Protrusion* while mapping the features in *Anchor Joint1*. The Label Matching returns a *True* for the *Rib* feature. However, the feature definitions based on implicit attributes are different in two systems. Therefore we receive *Similarity Degree* = 0.50 where $S_count = 3$, $T_count = 4$ and $sim_count = 2$.

The above two examples demonstrate the functionality of our matching algorithm. On average, the time complexity of the Label Matching is $O(\log N)$ where N is the number of target features. Attributes Matching has a time complexity of $O(nm)$, where n and m are the number of attributes in the target and source features respectively.

5. CONCLUDING REMARKS

This paper aims to enable lossless data exchange in distributed CAD environments with ease of design alternative evaluation and reuse, and reduced human errors. We presented a hybrid semantic feature modeling method to provide a neutral feature semantics format, which represents design history neutral feature information. The feature semantics are captured in RDF documents with the standard XML syntax, which facilitates knowledge management. CAD data exchange automation can be achieved by a document-driven design approach, where individual feature information is stored and transferred in documents. The hybrid semantic feature model captures both implicit and explicit features as well as necessary feature interrelationships.

With the hybrid semantic feature model, CAD models can be broken into history neural individual features once the feature boundaries are defined. This fine-grained documentation enables more efficient data exchange especially in real-time collaborative design. The feature-level dynamic matching supports the automation of data translation when multiple systems are used. We also developed a semantic mapping algorithm, which allows the semantic mapping with certain degrees of similarity instead of 100% accurate matching.

One drawback of the proposed approach is the dependence on a neutral format. Theoretically, features can be translated easily into the neutral format. However, in practice this can become a bottleneck. Moreover, this dynamic matching approach is designed for a large number of systems. Its efficiency will not be obvious if mapping between only a few systems. It could be more feasible to create one-to-one mappings instead of using a neutral model for a small number of systems.

ACKNOWLEDGEMENT

This work is supported in part by the National Science Foundation and I/UCRC Center for e-Design.

REFERENCES

- [1] Wang, Y., Nnaji, B.O., and Chiang, W.-S., 2005, "Document-Driven Design for Distributed CAD Services in Service-Oriented Architecture," *2005 ASME International Design Engineering Technical Conferences & The Computer and Information in Engineering Conference*, Long Beach, California, Paper No.DETC2005-84504.
- [2] Wang, Y. and Nnaji, B.O., 2006, "Document-Driven Design for Distributed CAD Services in Service-Oriented Architecture," *ASME Journal of Computing and Information Science in Engineering*, **6**(2), pp. 127-138.
- [3] Shah, J., 1991, "Assessment of Features Technology," *Comp.-Aided Des.*, **23**(5), pp.331-343.
- [4] Salomons, O.W., van Houten, F.J.A.M., Kals, H.J.J., 1993, "Review of Research in Feature Based Design," *J. Manuf. Sys.*, **12**(2), pp. 113-132.
- [5] Shah, J.J. and Mäntylä, M., 1995, *Parametric and Feature-based CAD/CAM: Concepts, Techniques, Applications*, John Wiley & Sons.
- [6] Pratt, M.J. and Anderson, B.D., 2001, "A Shape Modeling Applications Programming Interface for the STEP Standard," *Comp.-Aided Des.*, **33**(7), pp. 531-543.
- [7] Shah, J.J. and Rogers, M.T., 1988, "Functional Requirements and Conceptual Design of the Feature-based Modeling System," *Comp. Aided Eng. J.*, **5**(1), pp. 9-15.
- [8] Shah, J.J. and Rogers, M.T., 1988, "Expert Form Feature Modeling Shell," *Comp.-Aided Des.*, **20**(9), pp. 515-524.
- [9] Shah, J., Rogers, M., Sreevalsan, P., Hsiao, D., Matthew, A., Bhatanagar, A., Liou, B., Miller, D., 1990, "An Overview of the ASU Features Testbed," *Proc. 1990 ASME Computers in Engineering Conference, Boston, Massachusetts*, pp. 233-242.
- [10] Hoffmann, C.M. and Juan, R., 1993, "Erep – An Editable, High-Level Representation for Geometric Design and Analysis," P. Wilson, M. Wozny, and M. Pratt, eds., *Geometric Modeling for Product Realization*, North -Holland, pp. 129-164.
- [11] Chen, X. and Hoffmann, C.M., 1995, "Towards Feature Attachment," *Comp.-Aided Des.*, **27**(9), pp. 695-702.
- [12] Chen, X and Hoffmann, C.M., 1995, "On Editability of Feature-based Design", *Comp.-Aided Des.*, **27**(12), pp. 905-914.
- [13] Hoffmann, C.M., 1997, "EREP Project Overview," D. Roller and P. Brunet, eds., *CAD Systems Development*, Springer, Berlin, pp. 32-40.
- [14] Shah, J. Ali, A., and Rogers, M., 1994, "Investigation of Declarative Feature Modeling," *Proc. 1994 ASME Computers in Engineering Conference*, Minneapolis, MN, pp. 1-11.
- [15] Middleditch, A. and Reade, C., 1997, "A Kernel for Geometric Features," *Proc. 4th ACM Symp. on Solid Modeling & Applications*, Atlanta, GA, pp. 131-140.
- [16] Hoffmann, C.M. and Joan-Arinyo, R., 1998, "On User-defined Features," *Comp.-Aided Des.*, **30**(5), pp. 321-332
- [17] Bidarra, R., Idri, A., Noort, A., and Bronsvort, W.F., 1998, "Declarative User-Defined Feature Classes," No.DETC98/CIE-5705, *Proc. 1998 ASME Computers in Engineering Conference*, Atlanta, GA.
- [18] Wang, Y. and Nnaji, B.O., 2004, "UL-PML: Constraint-Enabled Distributed Design Data Model," *Int. J. Production Research*, **42**(17), pp. 3743-3763.
- [19] Shih, C.H. and Anderson, B., 1997, "A Design/Constraint Model to Capture Design Intent," *Proc. 4th ACM Symp. on Solid Modeling & Applications*, Atlanta, GA, pp. 255-264.
- [20] National Institute of Standards and Technology, <http://www.nist.gov/sc4/paramet/short/engen/edm46.pdf>
- [21] National Institute of Standards and Technology, 1988, *Product Data Exchange Specification: The First Working Draft*, NISTIR88-4004.
- [22] Shah, J.J. and Mathew, A., 1991, "Experimental Investigation of The STEP Form-Feature Information Model," *Comp.-Aided Des.*, **23**(4), pp.282-296.
- [23] Roy, U. and Liu, C.R., 1988, "Feature Based Representational Scheme of a Solid Modeler for Providing Dimensioning and Tolerancing Information," *Robotics & Comp.-Integrated Manuf.*, **4**(3/4), pp. 335-345.
- [24] Wang, N. and Ozsoy, M., 1991, "A Scheme to Represent Features, Dimensions, and Tolerances in Geometric Modeling," *J. Manuf. Sys.*, **10**(3), pp.233-240.
- [25] Gomes, A.J.P. and Teixeira, J.C.G., 1991, "Form Feature Modelling in a Hybrid CSG/Brep Scheme," *Computers & Graphics*, **15**(2), pp.217-229.
- [26] Choi, G.H, Mun, M., and Han, S., 2002, "Exchange of CAD Part Models Based on the Macro-Parametric Approach," *Int. J. of CAD/CAM*, **2**(1) pp. 13-21.
- [27] Li, M., Gao, S., Wang, C.C.L., 2007, "Real-Time collaborative Design With Heterogeneous CAD Systems Based on Neutral Modeling Commands," *ASME J. Comp. Inf. Sci. Eng.*, **7**, pp. 113-125.
- [28] Ullman, J.R., 1976, "An Algorithm for Subgraph Isomorphism," *Journal of the ACM*, **23**, pp. 31-42.
- [29] B.D. McKay, Department of Computer Science, Australian National University Canberra ACT 0200, Australia, *Nauty User's Guide (Version 2.4)*
- [30] Cordella, L. P., Foggia, P., Sansone, C., Vento, M., 1996, "An Efficient Algorithm for the Inexact Matching of ARG Using a Contextual Transformational Model," *in Proceedings of the 13th ICPR*, IEEE Computer Society Press, **3**, pp. 180-184.
- [31] Cordella L.P., Foggia, P., Sansone, C., and Vento, M., 2004, "A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(10).
- [32] Schmidt, D.C., Druffel, L.E., 1976, "A Fast Backtracking Algorithm to Test Directed Graphs for Isomorphism Using Distance Matrices," *Journal of the ACM*, **23**, pp. 433 – 445.

- [33] Corneil, D.G., Gotlieb, C.C., 1970, "An efficient algorithm for graph isomorphism," *Journal of the ACM*, 17, pp. 51-64.
- [34] Zhang, K. and Shasha, D., 1989, "Simple fast algorithms for the editing distance between trees and related problems," *SIAM J. Compute*, **18**(6), pp. 1245–1262.
- [35] Bhavsar, V.C., Boley, H. and Yang, L., 2004, "A Weighted-Tree Similarity Algorithm for Multi-Agent Systems in e-Business Environments," *Computational Intelligence*, **20**(4), pp.584-602, 2004.
- [36] Melnik, S., Molina, H.G. and Rahm, E., 2002, "Similarity flooding: A versatile graph matching algorithm," *Proceedings of Eighteenth International Conference on Data Engineering*, San Jose, California, pp. 117-128.
- [37] RDF, <http://www.w3.org/RDF/>
- [38] HP Labs, Jena-A Semantic Web Framework for Java, <http://jena.sourceforge.net/>