Ontology-Based Exchange of Product Data Semantics

Lalit Patil, *Student Member, IEEE*, Debasish Dutta, and Ram Sriram, *Senior Member, IEEE*

***Abstract*—An increasing trend toward product development in a collaborative environment has resulted in the use of various software tools to enhance the product design. This requires a meaningful representation and exchange of product data semantics across different application domains. This paper proposes an ontology-based framework to enable such semantic interoperability. A standards-based approach is used to develop a Product Semantic Representation Language (PSRL). Formal description logic (DAML + OIL) is used to encode the PSRL. Mathematical logic and corresponding reasoning is used to determine semantic equivalences between an application ontology and the PSRL. The semantic equivalence matrix enables resolution of ambiguities created due to differences in syntaxes and meanings associated with terminologies in different application domains. Successful semantic interoperability will form the basis of seamless communication and thereby enable better integration of product development systems.**

***Note to Practitioners*—Semantic interoperability of product information refers to automating the exchange of meaning associated with the data, among information resources throughout the product development. This research is motivated by the problems in enabling such semantic interoperability. First, product information is formalized into an explicit, extensible, and comprehensive product semantics representation language (PSRL). The PSRL is open and based on standard W3C constructs. Next, in order to enable semantic translation, the paper describes a procedure to semi-automatically determine mappings between exactly equivalent concepts across representations of the interacting applications. The paper demonstrates that this approach to translation is feasible, but it has not yet been implemented commercially. Current limitations and the directions for further research are discussed. Future research addresses the determination of semantic similarities (not exact equivalences) between the interacting information resources.**

***Index Terms*—CAD/CAM integration, ontologies, product data exchange, semantic interoperability.**

I. Introduction

MODERN product development is performed by cross-functional teams distributed geographically and temporally. Designers do not exchange mere geometric information about the design. All activities such as design, manufacturing, and marketing affect product development right from the stage of conceptual design. Thus, a product development activity requires the expertise and interaction of a broad range of disciplines. It is not a stand-alone activity like traditional design which involved a single user or co-located design team.

A broad spectrum of knowledge is used and shared in these modern collaborative but distributed teams. However, geographic, functional, and cultural boundaries between the teams have prevented the development of a single tool to support the entire product development activity. Therefore, next-generation product development systems will lead to the development and usage of a wide variety of software tools. Designers will use them to study and enhance the product from their own perspective in the best possible way. Further, growth in the use of the Internet has facilitated communication between various information systems that support product development. However, an essential feature of product information is the well-defined meaning (semantics) in a particular context. Consequently, this heterogeneous software environment will be successful only if it supports a meaningful representation and exchange of product data across different application domains.

There are several shortcomings of current product development systems regarding their suitability to collaborative design [1]. This paper focuses on the fact that there is no standard method to represent and interoperate semantics at various stages during the product development activity. Increasing trends for collaborative design tools necessitate the need to account for semantics to ensure seamless interoperability, which current systems lack.

This paper proposes a framework for the exchange of product data semantics between different domains within a product development activity. The next section discusses the meaning of semantics in the context of product development. Later, the paper presents an ontology-based approach for interoperability of product data. It studies current commercial and research efforts of relevance to semantic interoperability. Further sections present the development of a standard intermediate product ontology [Product Semantic Representation Language (PSRL)] and describe its semantics. It then presents a methodology for the exchange of semantics of product data between the PSRL and other application ontologies. This paper concludes with observations and scope for future work.

II. Semantics in Product Development

Semantics can be broadly defined as the meaning associated with a terminology in a particular context. Consider the representation of product information in two different domains of product development. Let the first domain represent a solid

Manuscript received May 14, 2004; revised January 12, 2005. This paper was recommended for publication by Associate Editor S. Gupta and Editor P. Ferreira upon evaluation of the reviewers' comments. This work was supported by the National Institute of Standards and Technology under Grant 60NANB2D01017 and by the PLM Alliance at the University of Michigan, Ann Arbor.

L. Patil is with the Mechanical Engineering Department, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: lpatil@umich.edu).

D. Dutta is with the Division of Graduate Education, National Science Foundation, Arlington, VA 22230 USA, on leave from the Mechanical Engineering Department, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: dutta@umich.edu).

R. Sriram is with the Design and Process Group, Manufacturing Systems Integration Division, National Institute of Standards and Technology (NIST), Gaithersburg, MD 20899-0001 USA (e-mail: sriram@nist.gov).

Digital Object Identifier 10.1109/TASE.2005.849087

TABLE I
SEMANTICS IN PRODUCT DESIGN AND MANUFACTURING

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	cuboidal shape	rectangular machining contour
Block	extruded object with cuboidal shape	2.5D object machined with rectangular contour

modeling (design) system and the other one represent a manufacturing system for the 2.5-dimensional (2.5-D) machining of a product. Let *Extrusion, hasShape, Cube*, and *Block* be four of the terms common in the two domains. However, each term may have a different meaning based on the application domain in which it is defined. For example, in the domain of solid modeling, *Extrusion* refers to an extruded object created by the geometric extrusion of a planar sketch. In the domain of 2.5-D machining, it refers to any 2.5-D machined object. Table I represents the meanings (of these terms) in the two contexts of solid modeling and process planning.

Such clashes of meanings require the development of a methodology to explicitly state and exchange meaning of terminologies of applications. This involves more than a mere exchange of geometric and nongeometric data. The next section presents a framework to enable semantic interoperability using an intermediate PSRL and briefly discusses the requirements associated with semantic data interchange through such a language.

III. SEMANTIC INTEROPERABILITY OF PRODUCT INFORMATION

The concept of self-integrating systems [2] is an evolution over the concept of self-describing systems in the Semantic Web [3]. The aim of this vision is the development of systems that can describe their own semantics and then establish successful semantic interoperability by interacting with each other. This paper does not focus on such a self-adjusting system. Instead, it assumes that semantics for the product data are already defined for every interacting application. The focus is on the ability to translate these semantics through a common format that represents product-specific information. This intermediate format is called PSRL. The development of such a system is a necessary initial requirement toward the development of self-integrating systems.

A. Framework for Semantic Interoperability

A typical ontology-based framework to enable semantic interoperability is depicted in Fig. 1. Consider two computer-aided-design (CAD) systems[1] A and B which are required to exchange semantics of product data between them in a collaborative product development environment. Each of them has its own predefined terminologies and semantics incorporated into its ontology. It is assumed that these ontologies are explicitly and formally defined. Each of these ontologies can be mapped into the proposed intermediate PSRL. Such semantic maps are represented by the thick arrows in Fig. 1. Syntax from system A

[1]This paper uses the term "CAD systems" to indicate any software that is a part of the process of product development.

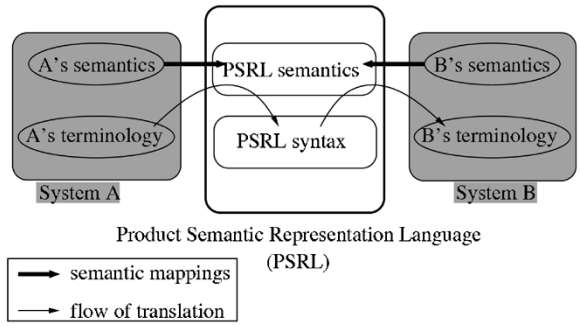

Fig. 1. Framework for semantic interoperability.

is then translated to syntax in the PSRL. This is then translated to syntax in system B. This is done with the help of the pre-determined semantic mappings and the reasoning abilities associated with the PSRL. The flow of such translation is depicted by thin arrows in Fig. 1. Since the ontologies are properly mapped across the systems, this translation of syntaxes enables a translation of the semantics as well [4].

The next section briefly states the requirements for a language and the corresponding exchange methodology to qualify as an enabler of semantic interoperability across CAD systems.

B. Requirements for Semantic Interoperability of CAD Systems

The following requirements distinguish semantic interoperability from current CAD interoperability which focuses on the translation of terminologies from one system to another. These requirements also formulate the research issues that need to be addressed during the development of the PSRL.

- **Application independence and dependence**
  The intermediate PSRL should be able to represent information that is common to all interacting applications. In other words, the PSRL should be application-independent. However, semantics of product information are relevant in a particular context (the application software). This implies that the PSRL should also be capable of representing application-specific product information. The number of software systems interacting in a product development activity is not fixed. Further, there is an increasing number of new software that are being developed to support the product development. Therefore, the intermediate PSRL should be extensible to account for the addition of new semantics throughout the product development activity.
- **Expressiveness**
  The PSRL must adequately express the meaning associated with a syntax. It must be a computer-readable format to enable automatic interoperability. The primary requirement for successful collaboration is that this language should be able to represent semantics associated with product data relevant throughout the various stages of the product development activity. It should not restrict itself to representation of information specific to any application domain.
- **Unambiguity**
  There can be different kinds of ambiguities that may arise during the translation of information from one CAD system to another.

Even if two systems use the same phrases, they may have different meanings. A system that converts semantics successfully should be able to identify meanings which are different within one system (or between different application contexts), but which are similar in physical representation based on syntax.

Ambiguity may be created by the existence of two different terms or sentences that have the same meaning. Suppose that a cylindrical surface is generated in the CAD software, I-DEAS. I-DEAS stores information of this surface as a NURBS entity. Let the same cylindrical surface be now generated in another CAD software, Unigraphics. Unigraphics stores information of this surface in its analytic form, i.e., information about axis, radius, etc. is stored. However, both specifications are just two different ways to convey the same meaning (that it is a cylindrical surface). A system that converts semantics successfully should be able to identify meanings which are equivalent within an application context, but which vary in physical representation.

Section IV discusses the state of the art of semantic interoperability and relevant research efforts.

IV. Semantic Interoperability—State of the Art

The primary goal of CAD interoperability is to facilitate sharing and exchange of product information amongst various modules within a product development system. The need for CAD interoperability is well documented in literature [5].

Various approaches to enable CAD interoperability have been studied [6]. The large number of interacting CAD systems necessitates the development of a single neutral intermediate format. This approach eliminates the need for multiple point-to-point translators between various interacting systems. It exponentially reduces the required number of translators.

The universally accepted solution for the exchange of solid model data is ISO 10 303 [6]. The next section studies ISO 10 303 as a commercial standard with particular emphasis on semantic interoperability. Later, other research efforts of relevance to semantic interoperability in collaborative product development are explored.

A. Current Standards—ISO 10 303

ISO 10303, informally known as STEP (STandard for the Exchange of Product model data) is an international effort toward the standardization of computer-interpretable representation and exchange of product data for engineering purposes.

The nature of its description makes ISO 10 303 suitable for neutral file exchange and as a basis for implementing and sharing product databases [6]. ISO 10 303 has been successful in the transfer of product shape in terms of its geometry and topology.

However, ISO 10 303 has been developed to exchange primarily product shape and shape-related information. Following are some of its shortcomings that are relevant to semantic interoperability.

- **Existence of a restricted common data model**. A well-documented common data model forms the basis of representation for all current standards. Only certain geometric information that is common to all commercial CAD systems can be converted from one CAD system to another. This is because the current standards are a subset of the intersection of information relevant to different CAD systems. They represent information in terms of a very small domain of phrases and subphrases common to all solid modelers. This leads to loss of the designer's intent during conversion of information to and from the intermediate standard. Successful collaboration in product development will require the translation of all information (and not just a subset defined common to all CAD systems) from one CAD system into information relevant to another CAD system.
- **Inability to translate semantics**. Current standards of interoperability fail to map even simple geometry related semantics between various CAD software. ISO 10 303 focuses only on the translation of terminologies from one CAD system to another. It does not attempt to translate the meaning associated with the design from one context to another. This associated meaning (which is usually the designer's intent) is lost during the conversion. Moreover, information that is lost in one context may be needed in another. ISO 10 303 does not satisfy all the requirements for semantic interoperability (Section III-B) because they are different than those for pure translation of terminologies from one system to another.

B. Other Research Efforts

Certain research efforts have focused on issues that are of relevance to the problem of semantic interoperability of product information. This section discusses these efforts.

Efforts to enhance ISO 10 303 to enable the exchange of parametrization and constraint information associated with solid models are discussed in [7]. This will enable a designer to edit the translated solid model in the receiving CAD system.

In [8], a unified modeling language (UML) [9] based mapping from CAD systems to product data management (PDM) software is presented. The significance of this effort is that it tries to integrate heterogeneous systems. However, the focus of the work is STEP-based translation of elementary product management data by direct mapping of terminologies from one system to another. There is no attempt to enable semantic interoperability.

New tools that support product life cycle are being developed. These tools represent a progress toward collaborative design. Nongeometric information such as function, behavior, various interrelationships, and design rationale needs to be represented and shared as a part of the entire design effort. However, these tools focus primarily on database-related issues and do not lay emphasis on information models for artifact representation [1]. It is the lack of a formal representation for product data that creates a barrier to its effective capture and exchange. Several research efforts are focused on the creation of a product representation in the domain of collaborative product design. They

are dedicated to answer questions at a higher level of collaborative design. They try to consider not only the "what" but also the "how" and the "why" of a design. A detailed survey of such research efforts is provided in [10]. The *core product model* [5] is proposed as a foundation for interoperability in next-generation product development systems. This paper analyzes these efforts to derive the neutral PSRL (Section III-A).

An effort of significant relevance is the development of the process specification language (PSL) at the National Institute of Standards and Technology (NIST) [11]. PSL defines a neutral representation for interoperability of information relevant to manufacturing processes. It considers the representation of process data used throughout the life cycle of a product, from early indications of the manufacturing process flagged during design, through process planning, production scheduling, and control. An ontology is being developed to facilitate exchange of information among various manufacturing process related software.

The focus of PSL is to represent and capture manufacturing process-specific data only. Efforts such as feature-based design have focused on equating some concepts across design and manufacturing, but they are not sufficient. Further, these efforts have been limited to the mapping of geometry and related information from design to manufacturing. Concepts that need to be represented in product data (such as design rationale, function, behavior, interpart relationships) are significantly different from those in process-specific data.

In [12], an approach toward the development of a product ontology and semantic mapping using first-order logic is presented. This effort proposes the development of a shared ontology. However, no details are provided on the development of the ontologies and how the system enables semantic interoperability.

In [13], an ontological approach is proposed to enable the exchange of features between application software. It uses the knowledge interchange format (KIF) [14] to model participating ontologies and to create a common intermediate ontology. Rules are manually specified to enable mapping of concepts from one domain to another.

It is observed that most research efforts relevant to product development are targeted toward the development of methods that enhance ISO 10 303 (STEP). Section IV-A discusses limitations of ISO 10 303. The need is to develop methods for semantic data exchange in the domain of product development. Other research efforts in this direction are incomplete and do not attempt to satisfy the requirements for the exchange of semantics as mentioned in Section III-B.

The AMIS project [15] at NIST identifies several technologies available to develop automated methods for integration of software systems. It identifies semantic conflicts as an important issue in solving an integration problem. Technologies and techniques that affect such an integration include the following:

- technologies to formally capture and represent semantics of software system;
- automated reasoning of tasks;
- mechanisms for integrating semantic information from multiple sources and recognizing commonalities.

Thus, in order to develop a system for semantic interoperability of product information, it is necessary to address the following broad topics:

- **Capture of intent:** It is necessary to capture the semantics of product information relevant to a particular application.
- **Representation of semantics:** It is necessary to encapsulate the semantics in a formal language to enable computer-readable and unambiguous representation.
- **Translation of semantics:** It is necessary to develop tools to enable the translation of semantics through the automatic or semi-automatic determination and resolution of semantic conflicts.

This paper proposes an ontological approach to capture the meaning of product information. Description logics are used to formally encode the semantics. The use of mathematical logic provides reasoning abilities that assist in the determination and resolution of semantic conflicts across ontologies.

Section V presents the development of the product ontology that forms the intermediate PSRL. Later sections will present a methodology for semantic mapping to and from this PSRL.

V. Product Semantic Representation Language

Ontologies have been found to facilitate semantic interoperability for the purpose of integrating systems [4]. There are several definitions of an ontology. However, with respect to interoperability, an ontology can be defined as an explicit specification of a conceptualization [16]. An ontology language usually introduces concepts (entities), properties of concepts (attributes), relationships between concepts (associations), and additional constraints.

Literature documents several methods that are proposed for building an ontology. A skeletal methodology for the building of ontologies is presented in [17]. It forms the basis of the ontology design for this work. This section describes the procedure of expressing engineering product knowledge into the intermediate ontology and thereby describes the PSRL.

A. Identifying Purpose and Scope

The first phase in the development of an ontology is to identify its purpose and scope [17]. The purpose of the PSRL is to serve as an interlingua to enable semantic interoperability.

Scope of the Ontology: Two application domains are considered for the development of tools and techniques to enable semantic interoperability of product information. They are: 1) CAD and 2) computer-aided process planning.

The design phase consists of application of design rules to generate a three-dimensional (3-D) CAD model of the product. Computer-aided process planning aids in creation of process plans in manufacturing. The input of this process is the CAD model of the workpiece to be created. The result of the process is a detailed process plan (or NC code) from which the workpiece can be fabricated.

This particular paper limits itself to the usage of SolidWorks (SW) and Unigraphics (UG) to generate and modify the CAD model. UG is also used (as the software in process planning domain) for the automatic NC code generation for the final fabrication of the product. The part domain is limited to prismatic

parts bounded by CSG primitives which can be machined using numerical control (NC) milling process.

After identifying the purpose and defining the scope of the ontology, the next phases in building the ontology involve capturing the ontology and coding it.

B. Capturing the Ontology

This phase involves identifying key concepts and relationships in the domain of interest. It also identifies terms to refer to such concepts and relationships and provides textual definitions for them.

The intent of this work is to create a strong foundation of requirements necessary to represent product data and not to create an exhaustive list of requirements necessary for every application. This is particularly important because the number of applications is constantly changing. Further, the application software enhance their capabilities over a period of time. The aim of this research is to develop a system that will be extensible. Thus, it should account for future enhancements and development of new application software.

Further, this research does not focus on the development of a new product representation model. Therefore, it does not develop new terminologies and assign new semantics to them. The focus is on the development of a neutral representation language in the context proposed in Section V-A. Corresponding mechanisms to enable semantic interoperability are developed for the effective exchange of product information.

In order to develop the PSRL ontology, the two software, viz. SolidWorks and Unigraphics, and the *Core Product Model* [5] were studied in detail. The *Core Product Model* presents a generic product representation scheme for the entire product development activity. This helps to generalize the domain of interest for other software similar to SolidWorks and Unigraphics. It also provides concepts that account for future developments in the product development software.

This detailed study led to the identification and classification of a certain set of core concepts and relations (properties) required to describe the domain of product development. This paper uses the *Core Product Model* as a basis for the development of a formal representation of product information. It should be noted that the definitions mentioned in this paper assume certain knowledge of the domain of product design. Providing detailed and complete definitions is beyond the scope of this paper.

Core Concepts in PSRL: The key concepts for the ontology can be briefly described as follows.

- Every concept is an *Object*. Thus, every *Assembly*, *Artifact*, *View* (*Front*, *Top*, etc.) is an *Object*.
- A *Specification* is information relevant to an *Artifact* based on customer needs and engineering requirements.
- An *Assembly* represents a collection of *Artifacts*.
- An *Artifact* represents a distinct entity in the design such that it has a *Form*, a *Function* and a *Behavior*.
- The *Form* of the artifact is the physical design solution for the problem specified by a corresponding *Function*.
- The *Function* specifies what the *Artifact* is supposed to do.
- The *Behavior* represents how the *Artifact* implements the *Function*.
- Every *Form* is represented by its *Geometry* and *Material*.
- A *Feature* is a *Geometry* with other associated *Objects* that may lead to some knowledge about its *Function*.
- A *Constraint* is an *Object* that defines a shared property that must hold in all cases.

More detail on the textual descriptions of the terms defining these concepts and relationships can be found in [5].

Core Relationships in PSRL: The key relationships for the PSRL can be briefly described as follows.

- An *Object* may have other *Objects* associated with it.
- An *Object* can be made up of subobjects. Every subobject represents a *child* of the object.
- Correspondingly an *Object* may have a super-object that represents its *parent*.

Using these core concepts and relationships, the ontology is explicitly coded to form a representation language (PSRL).

C. Coding the Ontology

Informal ontologies hinder the effectives of interoperability because they lead to ambiguities. Further, they cannot be used for automation because they are not entirely computer-interpretable. Therefore, formally defined ontologies are necessary for successful semantic interoperability.

This section describes the encoding of the ontology into a formal logical language. The lexicon and the axioms for the intermediate language (PSRL) are described in this section. Axioms also provide basic reasoning ability to the language. In order to keep the scope of the work feasible, not all the concepts and relations gathered in the previous phases are modeled completely. However, enough concepts and relations are included to provide a logical overview of domain of product development.

The semantics for the PSRL are encoded using representation and reasoning mechanisms based on description logic [18]. Description logics (DLs) are knowledge representation languages tailored for expressing knowledge about concepts and concept hierarchies. Most DLs are decidable subsets of first-order logic [19]. They are not as expressive as first-order logics. However, the decidability and tractability of reasoning services have made them a widely used tool for representing ontologies.

The syntax for encoding the PSRL is based on standards from the Semantic Web, in particular, DARPA Agent Markup Language (DAML) [20]. The latest release of the language $\mathrm{DAML+OIL}$ [21] provides a set of logical constructs to define ontologies and is based on XML. The underlying DL is obtained recursively by starting from a schema $S = (\mathrm{CN}, \mathrm{RN}, \mathrm{IN})$ of concepts names (CN), role names (RN), and individual names (IN). Concepts describe common properties of individuals. Roles are interpreted as binary relations between concepts.

1) Lexicon for the PSRL: Along with logical and nonlogical symbols provided by $\mathrm{DAML} + \mathrm{OIL}$ [21], the core of the PSRL consists of a nonlogical part of the lexicon (concepts and relations) that represents basic concepts in the PSRL ontology. These are equivalent to concept names and role names in the DL. In particular, these include the following:

- **Concepts:** *Object*, *Assembly*, *Artifact*, *Behavior*, *Constraint*, *Form*, *Function*, *Feature*, *Geometry*, *Material*, and *Specification*
- **Roles:** *hasChild*, *hasParent*, and *hasAttribute*

The intuitive semantics of these concepts and relations have been briefly described in Section V-B. All other concepts are derived from the basic concept, *Object*.

2) Axioms for the Core PSRL: The basic notions of this core PSRL are axiomatized formally using description logics. These axioms capture the basic properties of the ontology. They also provide semantics to the lexicon used in the PSRL. The precise definitions of the nonlogical symbols and corresponding axioms are described in this section.

- **Object** It is the most basic concept name in the PSRL. All concepts (and concept names) are subclasses of *Object*. In DAML + OIL, it is formally defined as follows.

```
⟨daml:Class rdf:ID="Object"⟩
 ⟨daml:sameClassAs⟩
  ⟨daml:Restriction⟩
   ⟨daml:onProperty
      resource="#name"/⟩
   ⟨daml:Cardinality⟩1
      ⟨/daml:Cardinality⟩
  ⟨/daml:Restriction⟩
 ⟨/daml:sameClassAs⟩
⟨/daml:Class⟩
```

However, for the purpose of this paper, we represent all other axioms using description logic syntax [22].

- **hasChild** This relation represents the existence of a child object. It has the following axioms:
  Axiom 1: The *hasChild* relation is transitive:

$$\mathrm{hasChild}^{+} \sqsubseteq \mathrm{hasChild}. \tag{1}$$

  Axiom 2: The *hasChild* relation is an inverse of *hasParent*:

$$\mathrm{hasChild} \equiv \mathrm{hasParent}^{-}. \tag{2}$$

- **hasParent** This represents the existence of a parent object. It has the following axioms:
  Axiom 3: The *hasParent* relation is transitive:

$$\mathrm{hasParent}^{+} \sqsubseteq \mathrm{hasParent}. \tag{3}$$

  Axiom 4: The *hasParent* relation is an inverse of *hasChild*:

$$\mathrm{hasParent} \equiv \mathrm{hasChild}^{-}. \tag{4}$$

- **hasAttribute** The *hasAttribute* relation is used to interpret the role played by single *Objects* as attributes within a description of another *Object*. Specific subproperties such as *hasFunction*, *hasForm*, and *hasConstraint* are created to define explicit relationships between various *Objects*. Each of these sub-properties may have more sub-properties, e.g., *hasDimensionalConstraint* is a subproperty of *hasConstraint*. Thus, there are no generic axioms associated with the *hasAttribute* relation.
  The *hasComponentArtifact* relation is a subproperty of *hasAttribute* relation. It is also a subproperty of the *hasChild* relation. It is used to represent the composition relationship between an *Assembly* and its *Artifacts*. Thus, an *Assembly* is described by its *Artifacts*, and the *Artifacts* are children of the *Assembly*.

Fig. 2. Example product (bracket).

It should be noted that some of the above-mentioned axioms (for example, transitivity of relations) are represented in DAML + OIL as properties of relations (and concepts). However, they are only different representations of axioms in the language. Furthermore, the axioms mentioned above are not the only ones for the corresponding relations and concepts. There are more axioms and more representation is required for a complete definition of any concept or relation. For example, the range of *hasParent* is an *Object*. Similarly, its domain is an *Object*. Such details have not been provided in this paper.

The encoding of the core constructs in the PSRL enable a generic formal representation of product information. More concepts need to be encoded within the PSRL. These provide more specific and instantiable constructs in the PSRL. The development of such new concepts utilizes the core constructs that have been mentioned in this section.

VI. Representation of New Concepts in the PSRL

This section demonstrates the ability to represent new concepts to represent product information in the PSRL. These concepts are developed onto the core PSRL mentioned in Section V.C. Only a few representative *Objects* are modeled here. Further, only geometric features are considered emphasizing the need for semantic data interchange even for geometric entities.

A. Example Product

The bracket shown in Fig. 2 is an object from the National Design Repository [23]. It has several features characteristic of a prismatic part generated by milling process.

The taxonomy for the features in this example component is shown in Fig. 3. It should be noted that the *Hole* feature has not been considered here because it does not present any semantic conflicts during the translation of product information from the application domains considered for the case study later.

The PSRL representation and involved axioms are stated as follows:

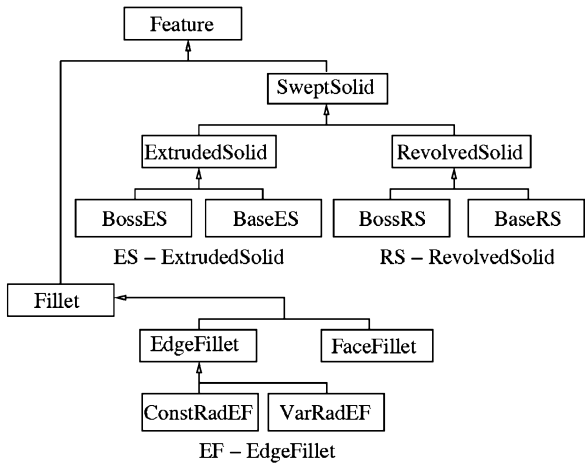

Fig. 3. Taxonomy of features of example product (Fig. 2).

Axiom 5 Every *SweptSolid* is a *subclass* of a *Feature* that has a *SweptSketch* as at least one of its parents:

$$\text{SweptSolid} \sqsubseteq \text{Feature} \sqcap \exists \text{hasParent} \cdot \text{SweptSketch}. \quad (5)$$

Axiom 6 An *ExtrudedSolid* is defined as *SweptSolid* such that it has exactly one *Direction* and exactly one *depth* of extrusion. Further, the *SweptDirection* has the *Constraint* that the *value* of the *AngleBetweenDirectionAndSketchPlane* (measured in *Degrees*) is 0:

$$\begin{aligned}\text{ExtrudedSolid} \equiv\ & \text{SweptSolid} \\ & \sqcap =1\text{hasSweptDirection} \sqcap =1\text{hasDepth} \\ & \sqcap \forall \text{hasSweptDirection} \cdot (\text{Direction} \\ & \sqcap \text{hasConstraint} \cdot (\\ & \text{AngleOfDirectionWithSketch} \\ & \sqcap\ (\text{Degrees} \sqcap \text{hasValue} \cdot \text{“0"}))). \end{aligned} \quad (6)$$

Axiom 7 A *BaseExtrudedSolid* is an *ExtrudedSolid* such that all its parents are a *SweptSketch*. Clearly, this means that the *BaseExtrudedSolid* does not have any *Feature* as its parent:

$$\begin{aligned}\text{BaseExtrudedSolid} \equiv\ & \text{ExtrudedSolid} \\ & \sqcap \forall \text{hasParent} \cdot \text{SweptSketch}. \end{aligned} \quad (7)$$

Axiom 8 Every *BossExtrudeSolid* is an *ExtrudedSolid* such that at least one of its parents is a *Feature*:

$$\begin{aligned}\text{BossExtrudedSolid} \equiv\ & \text{ExtrudedSolid} \\ & \sqcap \exists \text{hasParent} \cdot \text{Feature}. \end{aligned} \quad (8)$$

Axiom 9 Every *Feature* is disjoint with a *Sketch*:

$$\text{Feature} \sqcap \text{Sketch} \equiv \bot. \quad (9)$$

It should be noted that these definitions and axioms are derived from a study of the representations in SolidWorks and Unigraphics. Therefore, they represent a superset of the representations for the two application software. Similarly axioms for representation of a *RevolvedSolid* and *Fillet* are stated in the PSRL.

The PSRL models core concepts and relations that are common to the typical feature-based modeling systems. These core concepts can be combined using the language constructors to form new concepts as described in this section. Using this property, we can model the full set of features that are encountered in a typical CAD system. New atomic concepts and relations can also be specified in the PSRL if the existing set of atomic entities is insufficient to model any particular feature. Thus, for all practical purposes, all different types of features can be easily modeled within the PSRL.

The development of such a PSRL forms an interlingua and the basis of the development of methodologies to enable semantic data interchange.

VII. Semantic Data Interchange

Semantic translation involves determination of mappings between semantically equivalent terms between the application and PSRL ontologies. A mapping transforms instances in one ontology into instances in the other. This is done by using logical reasoning to compare definitions of terms in the two ontologies.

A key aspect of DAML + OIL is its well-defined formal semantics [21]. It provides methods to interpret composite descriptions in the language. Therefore, it facilitates machine interpretation of the ontologies. Such semantics allow the application of reasoners to infer relationships between concepts. In particular, *subsumption* and *satisfiability* can be inferred. This aids in the building of class hierarchies based on the DAML + OIL description.

In order to support reasoning, information from the application ontology is converted to PSRL's format. Therefore, both the application ontology and the PSRL ontology are encoded in a formal description language (DAML + OIL). The steps to translate semantics of product information from an application A to another application B through the PSRL are as follows.

1) **Syntactic translation:** Product information from the ontology of application A is converted into the format of the PSRL (a formal description logic). For this purpose, only the syntax is changed keeping the terminologies intact. Thus, syntax and terminologies of application A are converted to the syntax of PSRL and the terminology of application A, respectively. The syntax of PSRL is RDF/XML using DAML + OIL vocabulary.
2) **Semantic translation:** This phase involves the exchange of semantics. It is based on logical reasoning procedures that can determine equivalences between two symbols (both represented by similar syntaxes). In this stage, information represented by syntax of the PSRL and terminologies of application A is converted into a representation based on syntax and terminologies of the PSRL.

A reverse procedure is then applied to convert product information from the PSRL to the application B. The flow of information during this interoperability of semantics is depicted in Fig. 4.

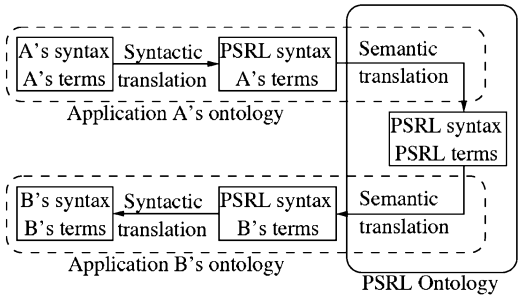

Fig. 4. Flow of information during semantic interoperability.

A. Syntactic Translation

One of the major challenges while building a syntactic (and semantic) translator is that no application has an explicit ontology. The framework of semantic interoperability requires translation of the application ontology into an axiomatized ontology (A's terminologies, PSRL syntax). In order to overcome this issue, this work models the software application directly in the PSRL syntax using the application's terminologies. This results in an axiomatized ontology like the PSRL. Such an axiomatized ontology provides the capabilities of inferencing and reasoning as mentioned earlier.

Thus, the problem of syntactic translation is not considered in this work. The primary focus is on the problem of semantic translation and, therefore, an axiomatized application ontology is considered as an input that is available or is manually generated for this purpose.

B. Semantic Translation

The translation of semantics is primarily based on the determination of semantic equivalences between two interacting ontologies. This section presents a methodology for the determination of such a semantic equivalence matrix. This methodology is derived from the concept of interpretation in mathematical logic.

1) Semantic Equivalence: In product development, it is desired that two concepts are equivalent if every instance of one concept is a valid instance of the other one. Such a definition of semantic equivalences will ensure that the designer's intent (in instancing a concept) during the creation/modification of a product is preserved in the other software system.

Two concepts, α from an ontology A and β from another ontology B, can be said to be semantically equivalent if any instance i of α in A is a valid instance of β in B, i.e., $\alpha(i) \iff \beta(i)$. Therefore, semantic equivalences can be expressed as bi-conditionals, i.e., α is semantically equivalent to β if $\alpha^B \iff \beta$ where α^B is the interpretation of α in B.

It should be noted that, in order to achieve this condition for semantic equivalence, it is necessary that exactly equivalent definitions of concepts should exist in the two ontologies. The method described in this paper assumes that the PSRL is designed to have an exactly equivalent definition (as mentioned above) for every concept in every application ontology. More detail on this assumption is given in Section XI. A determination of such equivalence is based on the concept of interpretation in mathematical logic.

2) Interpretation in Mathematical Logic: This section presents the mathematical background for the determination of semantic equivalences between two ontologies. More detail on the definitions that follow and corresponding proofs can be obtained in [24].

Let L_A be the language of the axiomatized application ontology A. Let L_P represent the language of the PSRL ontology P.

For any language, L_Δ of an axiomatized ontology is denoted by Δ

Definition: The theory generated by the ontology Δ is the set

$$T_\Delta = \{\alpha : \alpha \text{ is a sentence, and } \Delta \models \alpha\}.$$

For every sentence $\theta \in \Delta, \Delta \models \theta$. This implies that $\Delta \subseteq T_\Delta$.

Let T_A represent the theory of A and T_P represent the theory of the PSRL P.

Definition: T_A is interpretable in T_P if and only if there is a finite definitional extension T_P^+ of T_P such that $T_A \subseteq T_P^+$.

Intuitively, this means that T_A is interpretable in T_P if and only if there are definitions (in T_P) of those concepts of T_A that may not be in T_P such that all their properties in T_A are provable in T_P.

Therefore, every sentence $\alpha \in T_A \Longrightarrow \alpha \in T_P^+$. Let $A\alpha$ be the name of the sentence in T_P^+ created from α by replacing the symbols of L_A not in L_P by their definitions in L_P. Now, for every $A\alpha$ in $T_P^+, T_P^+ \models A\alpha \iff T_P \models \alpha^P$, where α^P is a sentence in L_P that represents the translation of $A\alpha$.

Thus, once the definitional extension T_P^+ is created, the determination of semantic equivalence of the sentence $\alpha \in T_A$ is the determination of the sentence $\alpha^P \in T_P$.

3) Determination of Semantic Equivalences: Let a concept in A be completely represented (including its properties) by a sentence α. Symbols (and sentence names) in A that exist in P with the same name and semantics are called *invariants*. The process of determining an equivalent sentence name (α^P) in T_P of α has the following two cases.

Case 1) α **is an invariant**. α is a concept in A that has exactly the same terminology, semantics and representation in both A and P. Therefore, $\alpha^P \equiv \alpha$.

Case 2) α **is not an invariant**. This case presents the following two subcases:

a) α **is an atomic concept**. In this case, an equivalence α^P in T_P is manually specified. The existence of such a concept in T_P is guaranteed because T_P (the intermediate PSRL) is designed to have all concepts that exist in T_A.

b) α **is not an atomic concept**. In this case, α is expressed completely (including its properties) only in terms of symbols that are invariants or have predefined mappings from T_A to T_P.

Using the interpretations (in T_P) of the symbols that are used to define α in T_A, a new sentence name $A\alpha$ is constructed in T_P^+. Although it is a new sentence name, $A\alpha$ represents exactly the same sentence represented by α. Thus, $A\alpha$ arises from α by

replacing the symbols of L_A not in L_P by their semantic equivalences in L_P.

T_P has been designed to have semantic equivalences for all symbols and sentence names in T_A. Therefore, the process of determination of the translation of $A\alpha$ (and therefore of $\alpha \in T_A$) is the determination of a sentence name $\alpha^P \in T_P$ that represents the same sentence defined by the new name $A\alpha \in T_P^+$.

Further, T_P is decidable by design (because of the use of $\mathrm{DAML} + \mathrm{OIL}$ which is designed as a decidable description logic language). Therefore, T_P^+ is also decidable. Logical reasoning can be used to parse through the finite set of names in T_P^+ to find the semantic equivalent name α^P such that $\alpha^P \in T_P$.

This is depicted by the pseudocode in Procedure 1. Thus, equivalences for primitive concepts are specified manually. These are then used to determine complex semantic equivalences.

Procedure 1: Procedure to determine semantic equivalences
Require: T_A, T_P, T_P^+, Sentence with name α defined by symbols ψ
Ensure: α^P in T_P is semantically equivalent to α
  for every ψ in α **do**
    replace ψ by ψ^P
  end for
  $A\alpha \leftarrow \alpha$
  $T_P^+ \leftarrow T_P^+ \cup A\alpha$
  Find sentence name $\alpha^P \in T_P$ such that $\alpha^P \equiv A\alpha$

4) Semantic Equivalence Matrix and Translation of Semantics: The determination of such semantic equivalences from A's ontology (A) to the PSRL (P) leads to the generation of a semantic equivalence matrix SEM_{AP}. This can be represented as a list of ordered pairs of concepts in the two ontologies, i.e., $\mathrm{SEM}_{AP} = \{(\alpha, \theta) : \alpha \in A, \theta \in P, \alpha^P \equiv \theta\}$.

This is followed by a similar determination of semantic equivalences from B's ontology to the PSRL. This is represented as another semantic equivalence matrix SEM_{BP}.

Consider a concept α in A's ontology. Suppose its semantically equivalent concept in the PSRL is θ, i.e., $(\alpha, \theta) \in \mathrm{SEM}_{AP}$. Let θ in PSRL be the semantically equivalent concept for β in B's ontology, i.e., $(\beta, \theta) \in \mathrm{SEM}_{BP}$. These semantic equivalences are determined by using the procedure mentioned above.

Therefore, $A\alpha \iff \theta \iff B\beta$, where $A\alpha$ is as defined above. $B\beta$ is obtained from B's ontology in a similar manner. Therefore, α and β are semantically equivalent concepts of each other. In other words, any instance of α in A is a valid instance of θ in the PSRL. Further, any instance of β in B is a valid instance of θ in the PSRL. Therefore, any instance of α in A is a valid instance of β in B.

This enables the translation of semantics between application ontology [A's (and B's) terminologies, PSRL syntax] and the PSRL (PSRL terminologies, PSRL syntax). A physical product is represented in the form of instances (individuals) of the various concepts in an ontology. Once the semantic equivalence matrices are determined, the translation of product information is achieved by translating values of the properties of individuals to create equivalent individuals (as instances of equivalent concepts) in the other ontology.

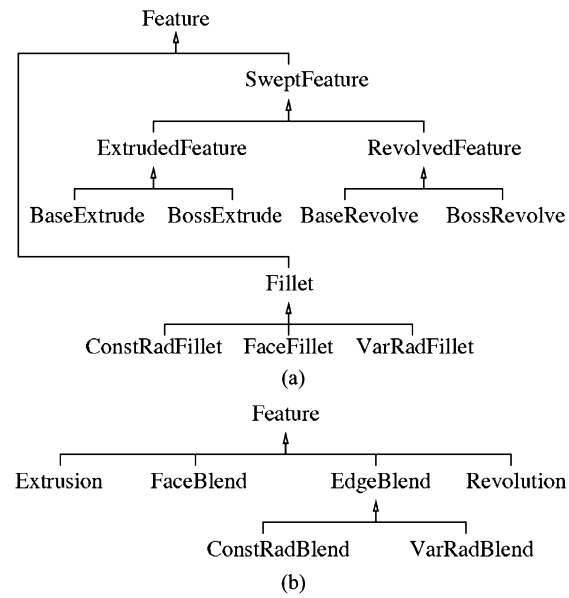

Fig. 5. Feature taxonomies for example product (Fig. 2) in SW and UG. (a) Feature taxonomy in SW. (b) Feature taxonomy in UG.

VIII. Implementation

This section presents details on the implementation of the semantic mapping methodology explained in Section VII.

Axiomatized ontologies for the application software and the PSRL are developed using the OilEd [25] ontology editor that allows a user to build ontologies using $\mathrm{DAML} + \mathrm{OIL}$.

The mapping matrix explicitly defines the semantic mapping between a pair of terminologies belonging to different domains. This semantic equivalence matrix is stored as a list of ordered pairs of strings (a, p) in a text file. The first string a represents the term in the application ontology, and the second one p represents interpretation in the PSRL ontology. In our work, one mapping matrix is generated for each application ontology that interacts with the PSRL.

The semantic equivalence determination methodology (Procedure 1) is implemented using the Jena toolkit [26] that provides a Java framework for writing Semantic Web applications. It provides support for the creation, modification, and reasoning of ontologies. Determination of semantic equivalences is performed using the Jena inferencing capabilities.

IX. Case Study

Consider the example part in Fig. 2. The aim is to translate product semantics between the interacting CAD systems SW and UG. Fig. 5(a) and (b) shows the taxonomies of feature information in the two domains. These taxonomies are only a subset of the real ones. The elements considered are representative of the need for the exchange of semantics.

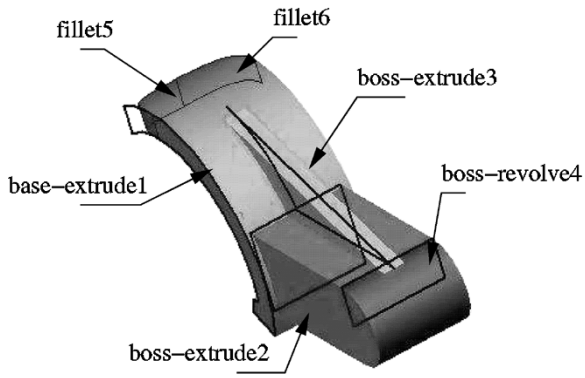

Fig. 6. Instances of feature concepts in SW for the example product.

Each concept in a taxonomy is associated with a set of instances, e.g., for the product in SW, *BaseExtrude* has an instance *base-extrude1*. Further, instances of a concept are also instances of the ancestor. For example, an instance of *BaseExtrude* is an instance of the *ExtrudedFeature* concept. Fig. 6 shows the product during an intermediate stage of development in SW. It shows the names of the individuals generated as a result of the instantiation of the concepts.

A. Axiomatized Ontologies

This section presents a partial ontology for the product in DL syntax (the physical representation language is DAML + OIL) for SolidWorks application software. The representation derives directly from the feature taxonomy depicted in Fig. 5(a). A similar ontology is developed for the Unigraphics domain. The axiomatized ontology derived from Fig. 5(a) consists of statements such as the following:

$$\begin{aligned}\text{ExtrudedFeature} &\sqsubseteq \text{Feature}\\ &\sqcap \exists \text{hasParent}\cdot\text{SweptSketch}\\ \text{BaseExtrude} &\equiv \text{ExtrudedFeature}\\ &\sqcap \forall \text{hasParent}\cdot\text{SweptSketch}\\ \text{BossExtrude} &\equiv \text{ExtrudedFeature}\\ &\sqcap \exists \text{hasParent}\cdot\text{Feature}.\end{aligned}$$

These concepts (and their definitions) in SolidWorks are similar to the definitions mentioned in Section VI for the example product.

A study of the two application ontologies (SW and UG) has provided us with the part of the PSRL ontology (see Section VI) that corresponds to this particular product.

B. Semantic Conflicts

Table II lists the semantic conflicts that occur during translation of the product information between SolidWorks and Unigraphics. The table lists elements that are equivalent in the two domains but are different in terminologies.

C. Semantic Equivalences and Translation of Semantics

The semantic mapping methodology described in Section VII leads to the determination of a pair of semantic equivalence matrices. The first matrix represents semantic mappings between SW and PSRL, and the second matrix represents semantic mappings between UG and PSRL. Table III shows a combined view of the two matrices to depict the semantic mappings across the three ontologies.

TABLE II
SEMANTIC CONFLICTS BETWEEN SW AND UG FOR THE EXAMPLE PRODUCT

Symbol in SW ontology	Symbol in UG ontology
BaseExtrude	Extrusion
BossExtrude	Extrusion
ConstRadFillet	ConstRadBlend
VarRadFillet	VarRadBlend
BaseRevolve	Revolution
BossRevolve	Revolution

TABLE III
SW–PSRL–UG SEMANTIC MAPPINGS FOR THE EXAMPLE PRODUCT

Symbol in SW ontology	Equivalence in PSRL	Symbol in UG ontology
BaseExtrude	BaseExtrudedSolid	Extrusion
BossExtrude	BossExtrudedSolid	Extrusion
ConstRadFillet	ConstRadiusEdgeBlend	ConstRadBlend
VarRadFillet	VarRadiusEdgeBlend	VarRadBlend
BaseRevolve	BaseRevolvedSolid	Revolution
BossRevolve	BossRevolvedSolid	Revolution

The table shows that the term *Extrusion* in UG is semantically equivalent to both *BaseExtrudedFeature* and *BossExtrudedFeature*. However, based on definition in the actual instance (which determines if there exists a parent as a *Feature*) of this concept, it maps to one of the two concepts as mentioned above.

In particular, consider the instance *base-extrude1* of *BaseExtrude* concept in SW (Fig. 6) and its translation into UG.

This instance is represented in the SW ontology as

$$\text{BaseExtrude}(\text{base}-\text{extrude1}) \sqcap \text{hasParent}\cdot\text{sketch1}.$$

This representation means that the instance *base-extrude1* as shown in Fig. 6 is an instance of the concept *Base-extrude* and has *sketch1* as its parent.

In the PSRL ontology, the equivalent concept for *BaseExtrude* is *BaseExtrudedFeature*. Therefore, the instance *base-extrude1* is represented in the PSRL ontology as

$$\text{BaseExtrudedFeature}(\text{base}-\text{extrude1}) \sqcap \text{hasParent}\cdot\text{sketch1}.$$

The concept *extrusion* from the UG ontology is equivalent to the concept *BaseExtrudedFeature* in the PSRL. Therefore, the individual *BaseExtrudedFeature(base-extrude1)* translates to the following in UG:

$$\text{Extrusion}(\text{base}-\text{extrude1}) \sqcap \text{hasParent}\cdot\text{sketch1}.$$

In all of the above three representations, *sketch1* is an individual that represents an instance of the concept *Sketch*, i.e., *Sketch(sketch1)*.

The symbol *BaseExtrudedFeature(base-extrude1)* represents an individual created after the translation of all properties and their values from SolidWorks ontology to the PSRL ontology. In the above representation, only the name of the instantiating concept has been depicted. This is only a partial component of the complete instance information. It should be noted that there is more information required for a complete representation of an instance.

Thus, a determination of semantic equivalences between concepts across ontologies enables translation of instances of those concepts from one software to another. The implementation in this paper demonstrates only the determination of equivalent concepts and corresponding translation of instances from one ontology to another. The ability to translate instances from the one application ontology to another application ontology implies the ability to translate a physical CAD model amongst the corresponding software applications. The actual translation of CAD model from one system to another will require the development of a syntactic translator (Section VII-A) from the CAD system to the ontological representation of the CAD software.

X. Discussion

This paper focused on the use of product ontologies for the determination of semantic maps to enable better data exchange. However, the intermediate PSRL and the method of its development can form the basis of the following potential applications in collaborative product development.

1) The extensibility of the core PSRL enables it to provide a basis for the development of new ontologies for new applications. A unified view of the domain experts in product development during the process of developing knowledge-based systems can be generated. Different levels of abstraction of knowledge can be represented using the same ontological base. A formal logic base in DL will also enable tractable reasoning for queries into the knowledge-based systems.
2) An explicit ontology can be used as a formal metadata for semantic searches on a repository of product models. Along with shape, all other information that is included in a product representation can be effectively utilized for better matches to a query. This will enable better re-use of previous designs and corresponding knowledge.

Web Ontology Language (OWL) [27] is an evolution over $\mathrm{DAML} + \mathrm{OIL}$. The work mentioned in this paper was started prior to the availability of a stable version of OWL. Therefore, $\mathrm{DAML} + \mathrm{OIL}$ has been used for all representation.

It should be noted that description logics such as $\mathrm{DAML} + \mathrm{OIL}$ may not be able to completely represent all the information that is required for the complete representation of a product. First-order logic such as Knowledge Interchange Format (KIF) [14] is best suited for a complete representation, although at the expense of computational efficiency in reasoning [19]. Restriction to the domain of Description Logics may be impossible if a very low (e.g., geometric entities such as points and lines) level of abstraction is to be achieved. At this level, we encounter more types of restrictions (e.g., asymmetry and the need to use variables) on concepts and relations that cannot be represented within the domain of DL. Efforts such as the proposed OWL rule language [28] use additional rule layers on top of the description logics in order to enhance expressiveness. Such extra expressiveness, however, impacts on the characteristics of the languages. We have restricted higher level product knowledge (features) within the domain of DL for the determination of semantic equivalence maps. Detailed translation can be achieved by using existing detailed definitions such as ISO 10 303. We may be forced to use a non-DL interface if semantic maps need to be determined at the lowest level of representation. Issues that need to be resolved during determination of such (not restricted to DL) semantic maps is not within the scope of our work.

XI. Conclusion and Future Work

An increasing trend toward product development in a collaborative environment has resulted in the use of various software tools to enhance the product design. This requires a meaningful representation and exchange of product data semantics across different application domains. This paper presented an ontology-based methodology to enable semantic interoperability of product information. We define the building blocks of an ontology (PSRL) for an intuitive and comprehensive representation of product information. Formal description logic is to encode the PSRL. This provides the PSRL with the following features:

- **Application Independence and Dependence**. The PSRL is developed using a standards-based approach of analyzing application ontologies that need to exchange semantics. Thus, it is application-independent. The core PSRL can be updated for new application-specific features because it can incorporate new concepts and relations. This extensibility of the PSRL provides the potential to enable integration of various existing and new application domains.
- **Expressiveness**. The PSRL is based on description logics and is therefore fairly expressive. The formal language provides a computer-readable format for successful automation of semantic interoperability.

In addition, the formal approach used to define the PSRL facilitates modifications to the representation while maintaining its validity and consistency. The use of mathematical logic along with the standards-based approach provides a sound procedure for the determination of semantic equivalences between the application ontology and the intermediate PSRL. Such semi-automatic determination of semantic maps will provide a correct input to and thus complement the use of well-developed translation standards (such as ISO 10 303) for the physical data translation.

The implementation using the W3C standards will enable the use of technologies that support data management over the Internet. This will aid in better collaboration in the design process.

The approach toward semantic interoperability (in particular, the approach to determine semantic equivalences) presented in this paper, is based on the assumption of the existence of exactly equivalent interpretations across two ontologies. In order to enable complete semantic interoperability, additional aspects of semantics and data exchange need to be considered. Some of these are as follows:

- determination of best semantic equivalences for terms that have semantic similarities with other terms but no exact equivalences;
- development of tools and procedures to prevent loss of information during the translation of information from the larger PSRL ontology to the smaller application ontology;

- development of procedures specific to product development environment to enable translation of complete product information along with efficient reasoning procedures.

Our method of representation of the semantic equivalence matrix will be inappropriate to represent inexactness in the semantic mappings. Currently, the semantic equivalence matrix is only capable of specifying one-to-one mappings. If the mappings are asymmetrical, the matrix becomes highly complex and multidimensional. Therefore, there is a need to develop better methods to represent the semantic equivalence matrix to enable efficient automation.

Methods to successfully enable semantic interoperability will form the basis of seamless communication and thereby enable better integration of computing environments in collaborative product development.

ACKNOWLEDGMENT

The authors would like to thank Prof. P. Hinman from the Department of Mathematics at the University of Michigan for his helpful discussions, comments, and suggestions in Section VII. They would also like to thank M. Gruninger from NIST for his insight and expertise. They also thank all of the reviewers who helped to improve this publication through their feedback.

DISCLAIMER

Certain commercial equipment, instruments, or materials are identified in this paper in order to facilitate understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, and PLM Alliance, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

REFERENCES

[1] S. Szykman, R. Sriram, C. Bochenek, J. W. Racz, and J. Senfaute, "Design repositories: Next-generation engineering design databases," *IEEE Intell. Syst.*, vol. 15, no. 3, pp. 48–55, May/Jun. 2000.

[2] S. Ray, "Interoperability standards in the semantic web," *ASME J. Comput. Inform. Sci. Eng.*, vol. 2, no. 1, pp. 65–71, Mar. 2002.

[3] T. Berners-Lee, J. Hendler, and O. Lassila, The semantic web, in Scientific American, May 2001. [Online]. Available: http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

[4] M. Ciocoiu, D. Nau, and M. Gruninger, "Ontologies for integrating engineering applications," *ASME J. Comput. Info. Sci. Eng.*, vol. 1, no. 1, pp. 12–22, Mar. 2001.

[5] S. Szykman, S. Fenves, W. Keirouz, and S. Shooter, "A foundation for interoperability in next-generation product development systems," *Comput. Aided Design*, vol. 33, no. 7, pp. 545–559, Jun. 2001.

[6] J. Owen, *STEP: An Introduction*. Winchester, U.K.: Information Geometers, 1993.

[7] M. J. Pratt, "Extensions of the standard ISO 10 303 (STEP) for the exchange of parametric and variational CAD models," in *Proc. 10th Int. IFIP WG5.2/5.3 PROLOMAT Conf.*, Trento, Italy, 1998, Paper no. 49.

[8] Y. Oh, S.-H. Han, and H. Suh, "Mapping product structures between CAD and PDM systems using UML," *Comput. Aided Design*, vol. 33, no. 7, pp. 521–529, June 2001.

[9] OMG Unified Modeling Language Specification (2003, Mar.). [Online]. Available: http://www.omg.org/cgi-bin/doc?formal/03-03-01

[10] S. Szykman, R. Sriram, and W. Regli, "The role of knowledge in next-generation product development systems," *ASME J. Comput. Inform. Sci. Eng.*, vol. 1, no. 3, pp. 3–11, Mar. 2001.

[11] The Process Specification Language (2005, Mar.). [Online]. Available: http://ats.nist.gov/psl/

[12] K. Y. Kim, S. H. Chae, and H. W. Suh, "An approach to semantic mapping using product ontology for CPC environment," in *Proceedings of the 10th ISPE International Conference on Concurrent Engineering Research and Applications*, J. Cha, R. Jardim-Gonçalves, and A. Steiger-Garção, Eds. Madeira Island, Portugal: A. A. Balkema, Jul. 2003, pp. 291–298.

[13] C. Dartigues, "Product data exchange in a cooperative environment," Ph.D. dssertation, Univ. of Lyon 1, Lyon, France, 2003.

[14] M. Genesereth and R. Fikes. (1992) Knowledge Interchange Format (Tech. Rep. Logic-92-1). Stanford Univ., Stanford, CA. [Online]. Available: http://www-ksl.stanford.edu/knowledge-sharing/kif/#manual

[15] E. J. Barkmeyer, A. B. Feeney, P. Denno, D. W. Flater, D. E. Libes, M. P. Steves, and E. K. Wallace. (2003, Feb.) Concepts For Automating Systems Integration (Tech. Rep. NISTIR 6928). National Inst. of Standards and Technol., Boulder, CO. [Online]. Available: http://www.nist.gov/msidlibrary/doc/AMIS-Concepts.pdf

[16] T. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," in *Formal Ontology in Conceptual Analysis and Knowledge Representation*, N. Guarino and R. Poli, Eds. Deventer, The Netherlands: Kluwer, 1993.

[17] M. Uschold and M. Gruninger. (1996) Ontologies: Principles, Methods, and Application (Tech. Rep. AIAI-TR-191). Univ. of Edinburgh, Edinburgh, U.K.. [Online]. Available: http://www.aiai.ed.ac.uk/project/ftp//documents/1996/96-ker-intro-ontologies.ps.gz

[18] F. Baader and W. Nutt, "Basic description logics," in *The Description Logic Handbook: Theory, Implementation, and Applications*, F. Baader, D. Calvanese, D. McGuiness, D. Nardi, and P. F. Patel-Schneider, Eds. Cambridge, U.K.: Cambridge Univ. Press, 2003, [Online] Available: http://www.inf.unibz.it/~franconi/dl/course/dlhb/dlhb-02.pdf, pp. 43–95.

[19] A. Borgida, "On the relative expressiveness of description logics and predicate logics," *Artif. Intell.*, vol. 82, no. 1–2, pp. 353–367, 1996.

[20] J. Hendler and D. L. McGuinness, "The DARPA agent markup language," *IEEE Intell. Syst.*, vol. 15, no. 6, pp. 67–73, Nov. 2000.

[21] A Model-Theoretic Semantics for DAML + OIL (2001, Mar.). [Online]. Available: http://www.daml.org/2001/03/model-theoretic-semantics

[22] F. Baader and W. Nutt, "Description logic terminology," in *The Description Logic Handbook: Theory, Implementation, and Applications*, F. Baader, D. Calvanese, D. McGuiness, D. Nardi, and P. F. Patel-Schneider, Eds. Cambridge, U.K.: Cambridge Univ. Press, 2003, pp. 485–495.

[23] National Design Repository (2005, Mar.). [Online]. Available: http://www.designrepository.org/

[24] P. Hinman, *Mathematical Logic and Foundations of Mathematics*. Ann Arbor, MI: Grade A Notes, 2002.

[25] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens, "OilEd: A reasonable ontology editor for the semantic web," in *Proc. KI2001, Joint German/Austrian Conf. Artificial Intelligence*. Vienna, Austria, Sep. 2001, pp. 396–408.

[26] Jena Semantic Web Toolkit (2005, Mar.). [Online]. Available: http://www.hpl.hp.com/semweb/jena2.htm

[27] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, OWL Web Ontology Language Reference. W3C Recommendation, Feb. 10, 2004. [Online]. Available: http://www.w3.org/TR/owl-ref.

[28] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, May 21, 2004. [Online]. Available: http://www.w3.org/Submission/SWRL/.

Lalit Patil (S'05) received the B.S. degree from Visvesvaraya Regional College of Engineering, Nagpur, India, the M.S. degree in mechanical engineering from Indian Institute of Technology, Bombay, and he is currently working toward the Ph.D. degree in mechanical engineering at the University of Michigan, Ann Arbor.

His current research interests include interoperability and the application of ontologies in PLM.

Debasish Dutta received the Ph.D. degree from Purdue University.

He co-directs the IGERT program at the National Science Foundation, Division of Graduate Education. He is currently on leave from the University of Michigan, Ann Arbor, where he is a Professor of mechanical engineering and Director of the PLM Alliance. His current research is in global product development and lifecycle management.

Ram Sriram (SM'00) received the B.S. degree from the Indian Institute of Technology, Madras, and the M.S. and Ph.D. degrees from Carnegie Mellon University, Pittsburgh, PA.

He is currently leading the Design and Process Group, Manufacturing Systems Integration Division, National Institute of Standards and Technology, Gaithersburgh, MD, where he conducts research on standards for interoperability of computer-aided design systems and on healthcare informatics. Prior to that, he was on the engineering faculty (1986–1994) of the Massachusetts Institute of Technology (MIT), Cambridge, and was instrumental in setting up the Intelligent Engineering Systems Laboratory. At MIT, he initiated the MIT-DICE project, which was one of the pioneering projects in collaborative engineering. He has coauthored or authored more than 175 papers, books, and reports in computer-aided engineering, including 13 books. He was a founding coeditor of the *International Journal for AI in Engineering*.

Dr. Sriram was the recipient of the Presidential Young Investigators Award from the National Science Foundation in 1989.