ASSEMBLY DIFFERENTIATION IN CAD SYSTEMS

by

David G. Manley

B. S. Industrial Engineering, University of Pittsburgh, 2004

B. A. Japanese, University of Pittsburgh, 2004

Submitted to the Graduate Faculty of

the School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science in Industrial Engineering

University of Pittsburgh

2006

UNIVERSITY OF PITTSBURGH

SCHOOL OF ENGINEERING

This thesis was presented

by

David G. Manley

It was defended on

May 8, 2006

and approved by

Dr. Bopaya Bidanda, Professor, Industrial Engineering

Dr. Michael R. Lovell, Associate Professor, Industrial Engineering

Thesis Advisor: Dr. Bart O. Nnaji, Professor, Industrial Engineering

Copyright © by David G. Manley

2006

ASSEMBLY DIFFERENTIATION IN CAD SYSTEMS

David G. Manley, M.S.

University of Pittsburgh, 2006

This work presents a data model for differentiating and sharing assembly design (AsD) information during collaborative design. Joints between parts are an important aspect of assembly models that are often ambiguous when sharing of models takes place. Although various joints may have similar geometries and topologies, their joining methods and process parameters may vary significantly. It is possible to attach notes and annotations to geometric entities within CAD environments in order to distinguish joints; however, such textual information does not readily prepare models for sharing among collaborators or downstream processes such as simulation and analysis. At present, textual information must be examined and interpreted by the human designer and cannot be interpreted or utilized by the computer; thus, making the querying of information potentially cumbersome and time consuming.

This work presents an AsD ontology that explicitly represents assembly constraints, including joining constraints, and infers any remaining implicit ones. By relating concepts through ontology technology rather than just defining an arbitrary data structure, assembly and joining concepts can be captured in their entirety or extended as necessary. By using the knowledge captured by the ontology, similar-looking joints can be differentiated and the collaboration and downstream product development processes further automated, as the semantics attached to the assembly model prepares it for use within the Semantic Web.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

API	Application Program Interface
ARM	Assembly Relation Model
AsD	Assembly Design
CAD	Computer-aided Design
DAML	DARPA Agent Markup Language
DOF	Degrees of Freedom
OKBC	Open Knowledge Base Connectivity
OIL	Ontology Inference Layer
OWL	Web Ontology Language
RDF	Resource Description Framework
SGML	Standard Generalized Markup Language
SR	Spatial Relationships
SWRL	Semantic Web Rule Language
SWT	Standard Widget Toolkit
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	eXtensible Markup Language

ACKNOWLEDGEMENTS

I would like to thank Dr. Bart Nnaji and the NSF Center for e-Design for providing me with the opportunity to perform this research and other research activities. Without the Center, I would not have had access to the people and resources that have taught me so much about CAD/CAE/CAM systems and engineering design. The insights that I have gained during my time with the Center will prove invaluable in my future engineering endeavors.

I am grateful for the wisdom, mentoring, and support provided by Dr. Kyoung-Yun Kim. His experience and guidance and has not only taught me about the principles of engineering design and development but also much about global cultures and lifestyles. He has taught me much more than what I could have learned in just the lab or classroom.

I would like to thank my family for their continuous love and support. My parents' non-engineering perspective of my work has yielded insights that I wouldn't have necessarily considered. My brothers' mechanical and microelectrical engineering viewpoints have broadened my view of engineering design as well.

Lastly, I would like to thank my fiancée, Marilyn, for being there for me throughout the whole thesis process. Her dedication to seeing me through this whole process motivated me to keep working at my best even when times were difficult. I cannot thank her enough for the many ways that she has helped me with everything.

1.0 INTRODUCTION

It is now commonplace that manufacturers to utilize the Internet and World Wide Web (WWW) technologies to improve time-to-market, supply chain management, and life cycle costs as well as to meet market demands in order to stay competitive. However, there remains a gap between these current market demands and current product development paradigms. For example, it is estimated that the cost of resolving software-related interoperability issues within the US automotive supply chain is approximately $1 billion dollars annually [Brunnermeier 1999]. One possible approach to fill this gap is to seamlessly integrate product development processes into a comprehensive collaborative design environment. To realize such integration, it is necessary to create an assembly design (AsD) data model that includes self-descriptive, semantic information which is sharable and prepares assembly models for downstream product development activities.

Regarding the development of a new AsD data model, the following notions should be considered. Mechanical products rarely consist of a single part. Joints in a mechanical assembly are often a necessary complexity in order to achieve the overall functionality of a product [Brandon 1997]. Assembly, therefore, plays a very important role in manufacturing. For instance, weld assembly accounts for a significant portion of US labor costs, with the number of weld-related workers totaling ten percent of the manufacturing workforce [AWS 2003]. Though welding has had such an impact on US manufacturers, it has been traditionally viewed only as a manufacturing function and has not been fully considered in the broader perspective of product

design and development. Technologies that can enable the systematic sharing of joint and welding design knowledge could potentially reduce costs and improve welding productivity [AWS-EWI 2001]. This implies that a new data model is needed to capture and propagate assembly models throughout the product development process.

To generate a robust assembly model, an understanding of assembly geometry and its functional and manufacturing considerations is necessary. However, current solid modelers and simulation software are not advantageous drivers of robust assembly models since they provide incomplete product definitions, focusing on product geometry, without much consideration of joining parameters and other such constraints. Certainly, geometry is of importance during assembly design, but form and shape characteristics are consequences of the principle physical processes and the design intentions (e.g. joint intent) [Horváth 1998].

Part of the difficulty in sharing joint design knowledge is that computer-aided design (CAD) environments provide little support for such aspects of engineering design. Assembly models may appear to be the same graphically; however, the non-graphical joint designs between the assembly components that differentiate assembly alternative designs are not usually captured in the assembly model. For example, consider the assembly model in Figure 1. This is a typical assembly of a plate with top-hat stiffeners often found on marine vessels. By visual inspection of the model, one cannot determine how the stiffeners are attached to the plate (e.g. welding, adhesive bonding, etc). What's more, even if one could determine the joining process employed through the use of notes and annotations on the assembly model, those notes and annotations are often lost during model sharing. In order to consider the joining processes, such as welding. early in the design phase of product development, one needs to share more than just the geometry of the assembly.

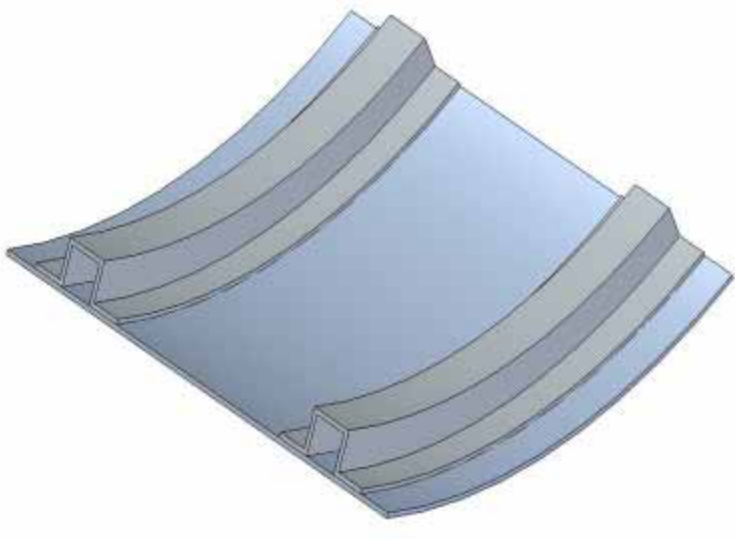


Figure 1. Plate with top-hat stiffeners.

Regarding CAD assembly models, designers do not only need to exchange geometric data as is done among many of today's CAD environments, but also to abstract engineering knowledge about the design and the product development process, including specifications, design rules, constraints, and rationale. As design becomes increasingly knowledge-intensive and collaboration tasks become more frequent, the need for computational frameworks to enable engineering product development by effectively supporting the formal representation, capture, retrieval, and reuse of product knowledge, becomes more critical [Lutters 1997]. That is to say, AsD models that contain more than just data, but also engineering design semantics that enable the systematic storage, transfer, and reuse of knowledge, may improve product development processes.

The primary motivation for modeling assemblies with engineering design semantics is to capture the requirements for the model and the rationale behind its design. Capturing such information facilitates reuse within the organization and supply chain and further enables

integrity constraints on AsD models so that they can be validated. Even more importantly, once a model exists, consistency analysis to discover errors can be preformed in conjunction with other models [Singh 2005].

Consistency checks can be difficult to perform, given the fact that heterogeneous tools and multiple designers are frequently involved in collaborative product development and the designers often use their own terms and definitions to represent a product design. Thus, to efficiently share design information among multiple designers, a designer's intentions should be persistently captured and the semantics of the designer's terms and intents should be interpreted in a consistent manner. Assembly models cannot be interrelated unless one can establish that they share the same universe of discourse. Ultimately, it is the universe of discourse that determines the overlaps between the models and thus makes the interoperation meaningful [Singh 2005]. For this purpose, a new data format is a necessity. Furthermore, the appropriate design knowledge should be provided during all design processes in order to make proper design decisions.

The Semantic Web supports integrated and uniform access to information sources and services as well as intelligent applications by the explicit representation of the semantics buried in an ontology. Ontologies provide a source of shared and precisely defined terms that can be used to describe web resources and improve their accessibility for automated processes. They provide a universe of discourse and a context for data sharing activities.

This paper presents an AsD ontology that is a formal, explicit specification of a shared conceptualization of assembly design modeling with attention given to joining constraints. By utilizing ontology technology, clear relations among assembly components and form features are established and assembly knowledge is organized into product, feature, manufacturing, and

spatial relationship classes. In this work, implicit assembly constraints are explicitly represented using SWRL (Semantic Web Rule Language) and OWL (Web Ontology Language). A meta-model called the assembly relationship model (ARM) [Kim 2004], which was originally developed to capture assembly engineering relations, is enhanced using ontology technology to satisfy collaborative engineering needs. The implicit constraints of assembly engineering relations, spatial relationships (SR), and joining are represented using OWL triples and SWRL rules.

The remainder of this work proceeds in the following manner. Section 2 provides an introductory background on semantic markup languages and technologies. Because there are many layers needed to realize computer-interpretable semantics, the goal of this section is to familiarize the reader with the basic purpose of each layer. Section 3 reviews existing systems that utilize ontology technology and also reviews the ARM proposed in [Kim 2004]. Section 4 introduces the technologies and tools used to develop the AsD ontology and an example browser application for viewing the content of ontology-based assembly models. Section 5 explains the construction of the AsD ontology and the browser application. Lastly, section 6 concludes this work and discusses some possible future extensions of this research.

2.0 BACKGROUND

The following subsections provide the reader with a background on the Semantic Web and associated markup language technologies. It is not expected that the reader of this work will be intimately familiar with these technologies, as they are geared more toward the computer scientist. The following subsections are not intended to provide the reader with a full explanation of how each of the technologies was developed and operates, but rather to provide enough information regarding their roles in adding semantics to a computer environment.

First, a brief overview of the concept of the Semantic Web will be given. Second, a summary of older technologies that attempted to achieve efficient data sharing will be given. Third, eXtensible Markup Language (XML) and its role as a foundation syntax will be discussed. Forth, the Resource Description Framework (RDF) and its function as a linking mechanism among distributed resources will be introduced. Fifth, the definition and role of ontologies as well as their associated markup language, Web Ontology Language (OWL), will be explained. Lastly, the Semantic Web Rule Language (SWRL) will be discussed.

2.1 THE SEMANTIC WEB

The original vision of Tim Berners-Lee's WWW included meta-data above and beyond the current WWW, that is, additional information that was machine interpretable [W3C 1992]. This

was a vision of the Semantic Web, a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. In other words, the Semantic Web is the WWW with inference capabilities.

The point of the Semantic Web is not just to make applications smarter, but also to make data "smarter" [Daconta 2003]. Data can be made "smarter" through the use of semantic technologies, such as concept maps or ontologies. In such concept maps and ontologies, computers will find the meaning of semantic data by following hyperlinks to definitions of key terms and rules for reasoning about them logically. This hyperlinked set of documents and data promotes the development of automated services and software agents, since the mapping of terms puts documents and data within a specific context or universe of discourse [Berners-Lee 2001].

Currently, the problem with performing intelligent tasks, such as web services or the sharing of engineering design models, is that they must rely on proprietary server languages and technologies to perform business logic. The Semantic Web's goal is to provide a mechanism through which computers can perform inferences on documents and data with less human intervention. With the Semantic Web, computers can make the inferences based on hyperlinked vocabularies which explicitly define concepts used in documents. These inferences would allow the computers to go beyond the simple linguistic analyses used by WWW search engines today [Alesso 2005].

At the core of the Semantic Web is the use of syntaxes utilizing URI's which link resources and terms in a document to other documents. What this means is that concepts are not just words within documents, but unique linked resources that provide context for the concepts in the document. The linked structure provides the basic mechanism for computers to infer facts

about a document, even the though the facts are not explicitly created by the author. Figure 2 depicts the pyramid of semantic markup languages recommended by the World Wide Web Consortium (W3C).

By preparing CAD assembly models for the Semantic Web, engineers will be able to share product data more effectively and reliably. If members of a supply chain use differing vocabularies to describe their data, ontologies will provide a mechanism for the computer to discern the equivalence of terms. This implies that models may possibly be shared with little or no human intervention required and with less data lost during the transfer process.

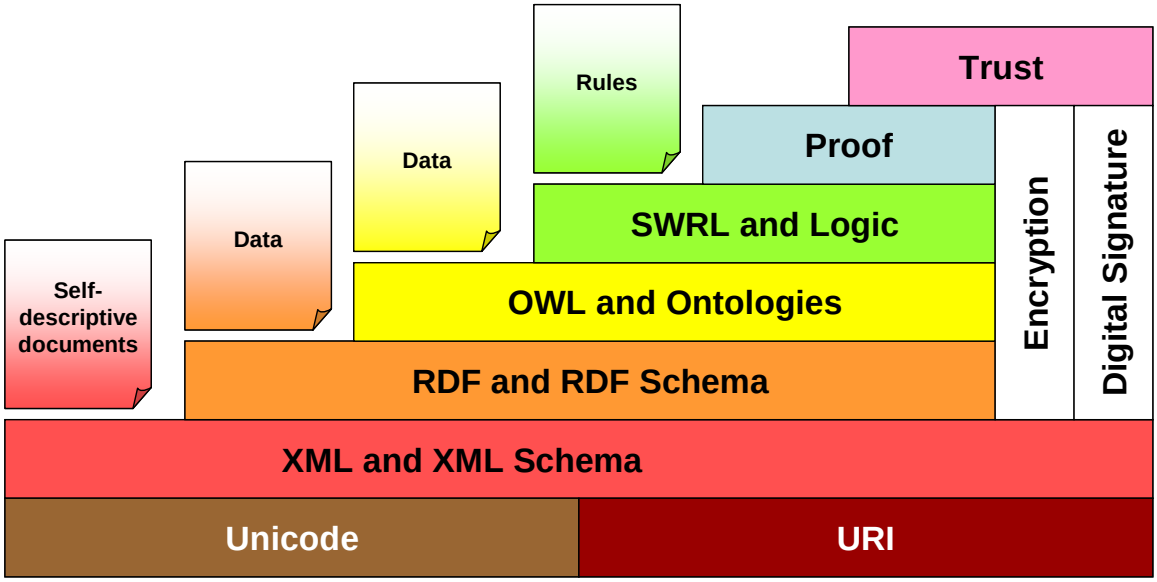


Figure 2. Pyramid of Semantic Web languages (adapted from [Berners-Lee 2000]).

2.2 DISTRIBUTED OBJECTS

Distributed software and data objects have been used for some time. Proprietary versions of distributed objects have been developed by a variety of organizations, such as Portable Distributed Objects (NeXT), Component Object Model (Microsoft), System Object Model

(IBM), and OpenDoc (Apple). Since the above-listed solutions did not solve the difficulties of sharing data, the Object Management Group created Common Object Request Broker Architecture (CORBA) [Alesso 2005]. Although CORBA was intended to be a mutual solution for data object interoperability and sharing, various implementations of CORBA were created and true interoperability was never achieved.

The problem with all of these solutions for distributed data objects is that they are complex to implement, and are binary solutions with compatibility issues. Moreover, they require for the most part a homogenous operating environment. The advantage of the markup languages used to develop the Semantic Web is that they are text-based, allowing heterogeneous systems to interoperate and use the data. XML and Hypertext Transport Protocol (HTTP) are the minimum technologies required to access objects, services, and servers.

As the AsD data model developed, it came to utilize XML as its core syntax and thus it can be transported via HTTP. This greatly improves sharability, as assembly models or portions of assembly models can be linked to or embedded in a larger variety of software objects. Furthermore, since the base technology is simple and clear, the data model can be extended to meet future needs.

2.3 XML & XML SCHEMA

XML (eXtensible Markup Language) is the open standard that allows data to be exchanged between applications and databases over the WWW. It is an interoperable bridge between dissimilar frameworks, such as Microsoft's .NET and Sun's J2EE [Alesso 2005]. XML was derived from SGML (Standard Generalized Markup Language), a powerful yet complex tagging

language for documents. To encourage the use of markup, XML was created as a simplified version of SGML, but maintains SGML's most important features of extensibility, structure, and validity.

In order for data to be shared among applications, it must be formatted so that it has a structure and can be parsed by the recipient application. Since it is possible that the data to be used with a variety of applications and programs, it is necessary for it be represented in a generic structure. Thus, XML allows users to add arbitrary structure to their documents through the use of tags, but says nothing about what the structure means. Clearly, it is advantageous to allow the users to define their own tags to add structure to their data; however, the recipient program writer must know in advance how the document used the tags in order to code the program appropriately. That is the recipient must know the context of the data before it can start using it. The above situation points out that while XML provides data exchange capabilities, it lacks semantics. Data formatted solely in XML can only be used for one specific purpose.

XML Schema is a set of rules that reference XML documents must follow. An XML Schema defines the structure and order of XML documents as well as the data types that are allowed to be used. It is metadata for the XML document that are used to check if the document is valid before transmission and sharing of data takes place. If an XML document is valid programmers can be sure that they can parse the XML document successfully.

The Assembly Relationship Model (ARM) proposed in [Kim 2004] was developed using only XML technologies. While this allowed the data model to be extensible, the data it contained lacked semantics. Thus, applications using the data model had to be built with that specific data model in mind. The AsD data model based on semantic technologies and

ontologies proposed in this work intends the overcome this lack of semantics by using richer technologies, namely, RDF, OWL, and SWRL.

2.4 RDF & RDF SCHEMA

If two organizations wish to share data between their applications using XML syntax alone, it is required that they agree beforehand on a common data structure (set and order of XML tags) before data exchange and processing begins. If the organizations "upgrade" their data to RDF, the two applications can share data using two different data structures, but maintain successful data transfer using the concept of equivalences (e.g., GMAW is equivalent to MIG welding).

RFD (Resource Description Framework) is a markup language, which enables the interconnection of distributed resources and is built on top of XML. Unlike XML which represents data as a tree structure, RDF represents data in the form of a variable number of triplets. A triplet contains a subject, predicate, and object, and thus each triplet represents a statement or fact about something. Essentially, RDF triplets make statements about resources, where resources are any items with an associated URIs (Uniform Resource Identifier) or addresses. Alternatively, a resource is linked via some property (which can be another resource) to a specific value or another resource. Figure 3 depicts graphically the basic structure of an RDF triplet.

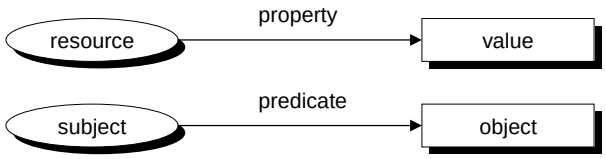


Figure 3. Structure of an RDF triplet.

The triplet structure of RDF is simple, yet very powerful for linking distributed resources. Whereas in XML the data must reside in an ordered tree structure, RDF triplets may reside in any order, yet each individual triplet does not loose its meaning (see Figure 4). The following are four important facts of RDF triplets to be aware of [Alesso 2005]:

1. Each RDF triplet is made up of a subject, predicate, and object.
2. Each RDF triplet is a complete and unique fact.
3. Each RDF triplet is a 3-tuple whose subject is either a reference or a "null", the predicate is a reference, and the object is a reference, "null," or literal value.
4. Each RDF triplet can be joined to other RDF triplets, but still retains its own unique meaning regardless of complexity.

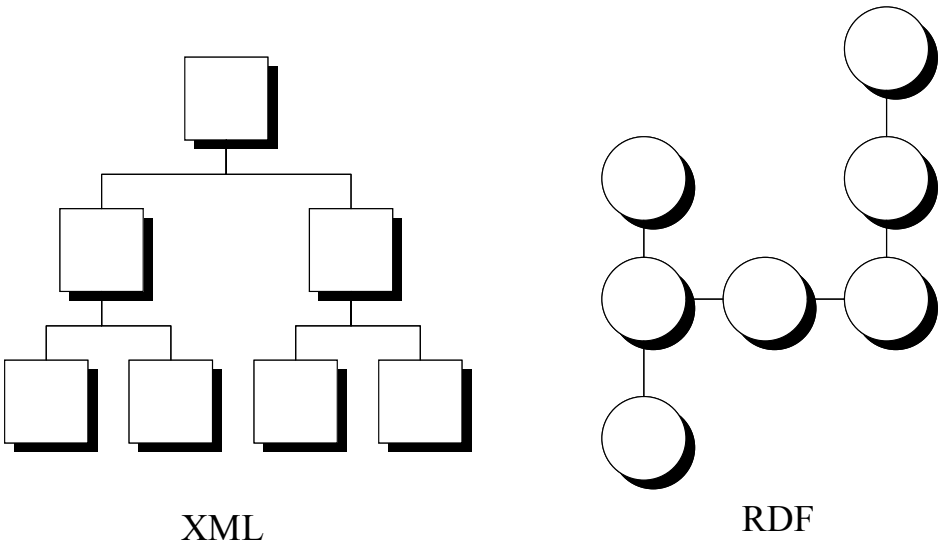


Figure 4. The structure of XML vs that of RDF.

RDF provides a syntax convention for representing the basic semantics of data in a standardized interoperable manner. To represent the meaning of data, the W3C developed RDF Schema (RDFS). The purpose of the RDF Schema is to define class relationships among vocabularies. In turn, such class relationships facilitate searching for specific pieces of data and, in a sense, give the data a context. Classes tell the user or application that which is represented. It is possible for a thing to be a member of many classes and not just reside in a hierarchical relationship. This capability makes RDFS a minimal language for representing ontologies.

Although RDF and RFDS are building blocks for defining a Semantic Web markup language, they lack expressive power. For example, they can't define:

1. The properties of properties
2. Necessary and sufficient conditions for class membership
3. Equivalence and disjointedness of classes

In addition, the only constraints expressible are domain and range constraints of properties. As a result, the semantics remain weakly specified. Each URI defines a new resource. Thus, there may be many resources which represent the same concept. The RDF specifications do not define a mechanism for stating the equivalence of resources; i.e., that multiple resources represent the same conceptual mapping. This is left to higher layers of the language stack, such as OWL [Alesso 2005].

In summary, RDF and RDFS are built on top of XML and provide a means for linking disparate resources which can be anything represented with an URI. RDF goes beyond the tree structure of XML and represents statements in the form of triplets which can be inter-linked with

one another. RDFS applies basic semantics to RDF by defining classes and properties for resources. Although this is the starting point for an ontology language, it lacks the expressiveness to define membership conditions and disjointedness of classes.

With regard to the assembly data model presented in this work, RDF provides the linking mechanism for portions of a data document. For example, model components of a particular mechanical assembly may reside with different vendors at different geographic locations. Use of RDF can link the disparate components together into one assembly model. RDF can also be used to link geometric entities within the assembly model to regulatory codes and standards throughout the WWW so that such information is always available along with the model.

2.5 ONTOLOGIES & OWL

A key problem in achieving interoperability over the Web is recognizing when two pieces of data are related to the same entity, even though different terminologies are being used. OWL may be used to bridge this "terminology gap" [Alesso 2005]. For instance, a program whose purpose is to compare information across two databases must know what two terms are being reference the same entity. Ideally, the program must have a way to discover common meanings for whatever databases it encounters. Ontologies can be used to discern common meanings.

An ontology is a kind of knowledge representation describing a conceptualization of some domain. Ontologies are explicit formal specifications of the terms used in the domain and relations among them [Gruber 1993]; this is a formal, explicit specification of a shared conceptualization. "Conceptualization" refers to an abstract model of some phenomenon in the

world that identifies the relevant concepts of that phenomenon. It specifies a vocabulary, including the key terms, their semantic interconnections, and some rules of inference. In addition, an ontology is a computational model of some elements of the world that are often represented in the form of a semantic network (a graph whose nodes represent concepts or individual objects and whose arcs represent relationships or associations among the concepts). From a collaborative viewpoint, an ontology is an agreement about a shared conceptualization which includes frameworks for modeling domain knowledge. In short, ontologies facilitate "conversations" among software applications through the Internet [Singh 2005]. Ontologies provide a means for dispersed programs to find common meanings among different vocabularies.

Typical ontologies consist of a taxonomy of classes and a set of inference rules. A rule may describe a conclusion that one draws from a premise. It can be a statement processed by an engine or a machine that can make an inference from a given generic rule. By utilizing rules machines can then make inferences about existing knowledge, which is new knowledge derived from already-existing knowledge. By enabling computers to infer new knowledge, it is possible to take some of the burden off human users who will no longer need to weed through information. The computer does not actually understand the information as humans do; nonetheless, it is now possible for computers to manipulate the information in a meaningful way.

The recommended markup language used to define ontologies for use in the Semantic Web is OWL (Web Ontology Language) [W3C-OWL 2004]. OWL grew out of a previous ontology language, DAML+OIL (DARPA Agent Markup Language + Ontology Inference Layer), and was the fruit of efforts on behalf of both the US and the European Union. Essentially, OWL is a standardization of previous knowledge representation efforts used to

extend RDF with a larger vocabulary so that richer logical relationships among concepts could be created.

Table 1. Comparison of markup languages (based on [DAML 2002]).

Capability	XML	RDF	OWL
Bounded lists		Yes	Yes
Cardinality constraints	Yes		Yes
Class expressions			Yes
Class membership conditions			Yes
Data types	Yes		Yes
Defined classes			Yes
Enumerations	Yes		Yes
Equivalence			Yes
Extensibility		Yes	Yes
Formal semantics		Yes	Yes
Inheritance		Yes	Yes
Inference			Yes
Local restrictions			Yes
Necessary and sufficient conditions			Yes
Negation			Yes
Object classes and properties		Yes	Yes
Ordered Data Set	Yes	Yes	Yes
Property/element domain	Yes		Yes
Property/element range	Yes	Yes	Yes
Qualified constraints			
Reification		Yes	Yes
Transitive properties			Yes

OWL ontologies differ from XML Schemas, since OWL ontologies actually model knowledge in a domain whereas XML Schemas are just syntax and a message format. XML statements by themselves do not allow one to draw conclusions about other XML statements.

OWL ontologies differ from RDF Schemas in that OWL ontologies are much more expressive. OWL is built on top of RDF and thus utilizes all of the constructs found in RDF/RFDS, but OWL also adds additional vocabulary in order to represent logical relations, such as properties of properties, class membership conditions, cardinality constrains, and equivalences/disjointness of classes. Table 1 provides a brief comparison of the XML, RDF, and OWL markup languages.

OWL by itself is just a data format using rules and description logic to relate terms. In order to utilize the data in OWL one's application must be built to accept data in OWL format. However, since data in the OWL format is well-marked, it is possible to infer new knowledge from already-existing knowledge without human intervention. To do this, an inference engine is utilized to derive new facts. Figure 5 presents the basic application architecture that utilizes explicit OWL data as well as inferred facts.

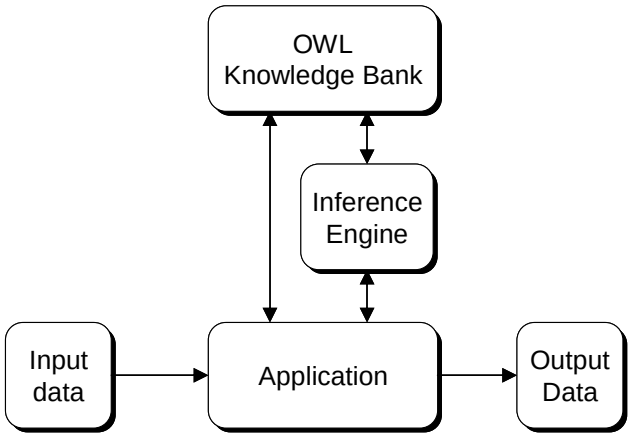


Figure 5. Basic architecture of an application utilizing OWL (adapted from [Alesso 2005]).

2.6 SWRL

While the expressiveness of OWL greatly promotes Semantic Web activity, it does have its limitations. To address this, the Semantic Web Rule Language (SWRL) was created to extend the description logic and axioms of OWL with Horn-like rules. The rules can be used to infer new knowledge from existing OWL knowledge bases [O'Connor 2005]. SWRL is based on a combination of the OWL DL and OWL Lite sublanguages OWL with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language [W3C-SWRL 2004]. Many of OWL's limitations concern the development of OWL properties. As there is no composition constructor, capturing relationships between composite properties is not possible [Horrocks 2005].

SWRL rules are of the form of an antecedent (body) and consequent (head). The basic meaning of the rule may be stated as "if the conditions for the antecedent hold, then the conditions for the consequent must also hold." The head and body consist of a conjunction of one or more atoms. SWRL rules reason about OWL individuals (instances of classes), primarily in terms of OWL classes and properties. For example, a SWRL rule expressing that a person with a male sibling has a brother requires capturing the concepts of 'person' 'male' and 'brother' in OWL [O'Connor 2005].

Golbreich et al. [Golbreich 2005] state in their work that rules are needed in addition to the ontology to capture (1) dependencies between ontology properties, (2) dependencies between ontologies and other domain predicates, and (3) queries. The necessity of rules is of course dependant on the use of the end application utilizing the OWL knowledge base.

In the developed AsD ontology presented in this work, SWRL is used to enforce joining constraints as well as spatial relationship constraints. The knowledge inferred from the rules ensures that the assembly behaves as intended.

3.0 LITERATURE REVIEW

The following subsections review existing systems and frameworks related to engineering and assembly design. Section 3.1 reviews existing ontology-based knowledge systems starting with broad systems and then more narrow, detailed systems for engineering design. Section 3.2 reviews the assembly design formalism and assembly relationship model proposed in [Kim 2004]. The ARM in is reconstructed in this work into an AsD ontology using semantic technologies to improved its use in the Semantic Web.

3.1 EXISTING ONTOLOGY SYSTEMS

The following statements are repeated from section 2 in order to refresh the reader on the broader aspects of the Semantic Web. The original version of Tim Berners-Lee's WWW included meta-data above and beyond the current WWW, that is, additional information that was machine interpretable [W3C 1992]. The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. The point of the Semantic Web is not just to make applications smarter, but also to make data smarter [Daconta 2003]. Data do not/should not reside in application specific databases. Also Data may become smarter through the use of higher semantics from technologies such as concept maps or ontologies.

Ontologies are explicit, formal specifications of the terms in the domain and the relationships among them [Gruber 1993]; formal, explicit specifications of a shared conceptualization. "Conceptualization" refers to an abstract model of some phenomenon in the world that identifies the relevant concepts of that phenomenon. "Formal" refers to the fact that the ontology should be machine-readable [Fensel 2001]. Mizoguchi [Mizoguchi 2003, 2004, 2004a] presented the roles of an ontology as being a common vocabulary, data structure, explication of what is left implicit, semantic interoperability, explication of design rationale, systemization of knowledge, meta-model function, and theory of content.

Ontologies have been developed for a variety of domains, most of them being broad. The broadest of ontologies are the upper-level ontologies that describe common sense-level knowledge. CYC, developed by Cycorp, is a commercial ontology containing over 200,000 terms and assertions. Its goal is to define high-level, common sense-type concepts in a machine-interpretable manner. Potential applications for CYC include online brokering of goods and services, enhanced virtual reality, improved machine translation, improved speech recognition, data mining, and true language processing among others [CYC 2005]. However, since CYC is a high-level ontology, it has not had a strong impact in the mechanical design domain. Nonetheless, in 1999, the National Institute of Standards and Technology (NIST) chose CYC as an ontology for further investigation in the manufacturing domain [Schlenoff 1999]. The results from this investigation lead to the development of Process Specification Language (PSL), which is a language that is sufficiently generic enough to represent discrete manufacturing and construction process data [Gruninger 2003].

Narrower in scope than upper-level ontologies, enterprise-level ontologies attempt to formalize the practices and processes that occur within an organization. The level of concepts is

enterprise specific and is meant to promote knowledge reuse with regard to business decisions and transactions. Enterprise Ontology [Uschold 1998] intends to define the overall activities of an organization. While this ontology takes into account the business aspects of an organization, it does not define engineering activities in detail. Similar to the overall goal of Enterprise Ontology, TOVE [Fox 1992, 1998] is an ontology for enterprise knowledge. The results of TOVE particularly in the domain products and requirements, are closer to the knowledge-intensive tasks of engineering design than Enterprise Ontology, yet they do not capture all detailed forms of mechanical design knowledge.

In the engineering domain, Lin et al. [Lin 1996] developed a Knowledge Aided Design (KAD) system to capture knowledge from engineering tasks, particularly those tasks related to engineering requirements. Issues that they address for the motivation of their work include communication, traceability, completeness, consistency, document creation, and managing change. They used an object-oriented approach to implement their work.

Some ontological research has been applied at both the conceptual and detailed design levels as is explain in the following examples. Kitamura et al. [Kitamura, 2002, 2003, 2004, 2004a] successfully developed an ontology to represent functional design and deployed the ontology into industry. While their work captures information regarding the flow of something (e.g. the flow of fluids or parts in manufacturing), it has limitations when it comes to the capturing of complex mechanical phenomena. Horváth et al. [Horváth 1998] attempted to create an ontology for design features using ontology theory. They classify design concepts in terms of entities, phenomena, and situations. Grosse et al. [Grosse 2005] developed an ontology-based knowledge management system that maintains engineering analysis models in order to facilitate

model sharing and reuse. They formally identified the terms used in generic analysis modeling as well as continuum-based and discrete finite element models.

Of all the ontologies reviewed here, the majority of them have been developed and applied to broader business applications, which do not have the level of detail required for mechanical design. Although some of the aforementioned research has been applied to design, it has limitations when it comes to representing mechanical assemblies and embedded relationships.

3.2 ASSEMBLY DESIGN FORMALISM

This section introduces the AsD formalism from [Kim 2004] that this work's semantic AsD ontology is based on for capturing assembly/joining relations and spatial relationship implications. The formalism allows joining relations to be modeled symbolically for computer interpretation, and can be used to infer mathematical and physical implications for downstream activities. An assembly relation model (ARM), which is a meta-model generated through use of the AsD formalism, is used to exchange AsD information transparently in a collaborative AsD environment, independent of CAD file formats. All geometric entities in the ARM are linked to a related solid model. Through the AsD formalism, a designer specifies spatial relationships, joining methods, and joining conditions between assembly components. Based upon the user's input, assembly and joining relations are generated that are automatically extracted from the assembly and the models, which have mathematically solvable implications.

An assembly relationship indicates which components are assembled and their corresponding joining relationships, while a joining relationship denotes how and by what

method assembly components are joined. The AsD formalism, which the captures assembly/joining relationships of a product assembly, is comprised of five phases: spatial relationship specification, mating feature extraction, joint feature formation and extraction, assembly feature formation, and assembly engineering relationship extraction. The following subsections outline these five phases. For a more detailed explanation of the AsD formalism, please refer to [Kim 2004].

3.2.1 Spatial Relationship Specification

By interactively assigning spatial relationships, the designer can assemble components together to make assemblies and infer the degrees of freedom (DOF) remaining on each component. Each spatial relationship can be interpreted as a constraint imposed on the DOF between interacting features. In other words, any allowable motion must follow a path along the directions specified by the DOF in order to maintain spatial relationships. By assigning spatial relationships, the areas of contact between two assembly components, mating features, can then be defined and extracted from the parts.

3.2.2 Mating Feature Extraction

Frequently, two assembly components do not make contact over their entire surface areas; only portions of each part are in contact. Mating features are derived from these portions. Based on the definitions for feature and form feature in [Liu 1991] and [Shah 1998], a mating feature can be defined as a set of component geometric entities of form features which are needed to relatively locate the parts according to their spatial relationships in the whole product assembly.

Mating features are necessary for representing assembly and joining relationships, as actual assembly operations occur at the mating features. Spatial relationships and corresponding mating features provide fundamental elements used to describe an assembly. Mating feature extraction is a preliminary step for capturing joining information; however, mating features extracted directly from spatial relationship specification, are not sufficient for the representation of joining processes. The next step is the joint feature formation process.

3.2.3 Joint Feature Formation

Joining processes, such as welding, occur as seams between the mating entities. Mating features of the mating entities impose limitations on the representation of special configurations for joining (e.g. weld seams and grooves). To overcome these limitations, a feature called the joint feature is defined as a set of information, including joining methods, groove shapes, joining components and entities, and joining constraints, which is used to represent assembly/joining relations.

The joining entities should be included in the mating features extracted during the previous stage. If a designer specifies a geometric entity, which does not belong to the mating features as a joining entity, then it violates the validity of joining. After joint features are determined, the system is then ready to proceed forward to assembly feature formation.

3.2.4 Assembly Feature Formation

The purpose of assembly feature formation is to group the mating features and joint features together and thus integrate the data embedded at the component design stage with new assembly information for subsequent collaborative design processes, such as virtual prototyping and

simulation, assembly violation detection, process planning, etc. Having designated spatial relationships, mating features, and joint features, the system can then trace back to the component design stage and determine from which form features these mating features originate and what their design specifications are. An assembly feature is defined as a set of assembly information, including form features and joint features associated with mating relations and assembly/joining relations, such that the association includes a set of spatial relationships between mating features, mating bonds, material, remaining degrees of freedom, as well as other constraints implied by the original design intents on the form features. While mating features provide links between the spatial relationships and engineering information necessary for the succeeding processes, assembly features act as media that carry and transmit all information to the downstream steps.

3.2.5 Extraction of Assembly Engineering Relations

Assembly engineering relations of an entire assembly can be extracted from the assembly features after specifying the spatial relationships and joining methods between components. A mating bond is used to explicitly represent the engineering relationships that hold within the entire structure and passes necessary information to subsequent assembly analyses. It provides an efficient design data sharing mechanism in collaborative AsD environments.

The components of a mating bond are shown in Figure 6 with the two dominant types of information being the mating pair and the mating conditions. The mating pair contains two mating features involved in the joining. The inter-feature association of form features related to the assembly is used to record form features, subassemblies, or assemblies in which the form features associate. Using the mating features, the system traces back to its original form features

and inherits the implied constraints. The mating conditions include the assigned spatial relationships, designed DOF, and assembly/joining relations.

As mentioned above, assembly engineering relations of an assembly structure can be extracted based on the mating features and joint features after specifying the spatial relationships and joining methods between assembly components. A mating bond is created once two mating features of different components are selected and positioned with each other and joint features are formatted. Assembly features are organized by a set of one or more mating bonds.

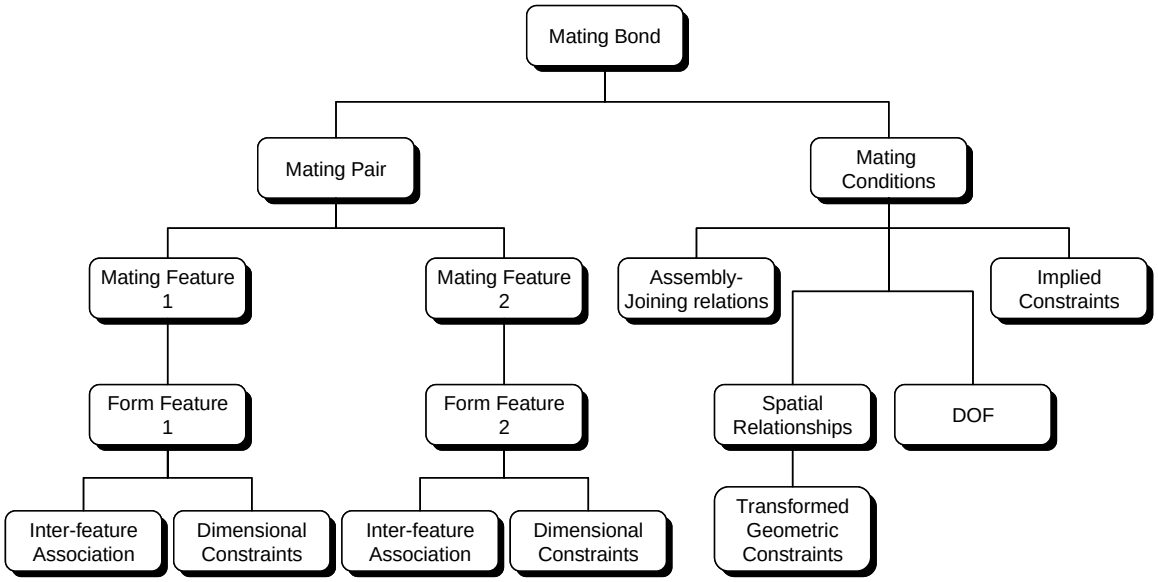


Figure 6. Components of a mating bond (from [Kim 2004]).

After assembly features are generated, intra-feature and inter-feature relationships are captured in an assembly relation model (ARM). The three types of relations mathematically defined in the model are the *belongTo*, *inter-featureAssociation*, and *assembly/joining* relations. BelongTo relations describe which form features belong to which part. Inter-Feature-Association relations indicate how form features of a particular part relate to each other (e.g. a

block feature may contain a hole feature). Assembly/joining relations signify which form features are on separate parts that form joints. The relations in the ARM are linked with a corresponding solid model to form the final AsD model. This explicit representation of assembly engineering relations will be enhanced by using ontology technology in this research work.

3.2.6 Technological Requirements of the AsD Data Model

The AsD formalism was developed to represent AsD and joining information explicitly using XML; however, some assembly/joining concepts were left implicit. It is possible to represent all such concepts explicitly using XML, but not in a universally acceptable manner. By relating concepts through ontology technology rather than by just defining data syntax, assembly/joining concepts can be captured in their entirety or extend as necessary. Furthermore, the higher semantic richness of ontologies allows computers to infer additional assembly/joining knowledge and make that knowledge available to AsD decision makers. By using ontology technology, AsD constraints can be represented in a standard manner regardless of geometry file formats. Such representation will significantly improve integrated and collaborative assembly development processes, including design, analysis, and decision making. Lastly, given that knowledge is captured in a standard way through the use of an ontology, it also can be retrieved, shared, and reused during collaboration.

4.0 DEVELOPMENT TOOLS

The following subsections briefly introduce the tools that were used to develop the AsD ontology. Section 4.1 presents the knowledge base and ontology, Protégé, which was developed at Stanford University. Section 4.2 examines the reasoning engine, Bossam, which was used to derive new facts from existing knowledge in the AsD ontology. Section 4.3 discusses the programming items used to develop a browsing application for the AsD ontology.

4.1 ONTOLOGY DEVELOPMENT TOOLS

For the implementation of the ontology and its rules, the Protégé knowledge and ontology editor was used [Noy 2000]. This tool was originally created to model medical and bio informatics knowledge, but has since gained wide acceptance in the knowledge modeling community as a whole. Currently Protégé is used to model knowledge in many business and scientific communities throughout the world.

Protégé is not just a single application, but an open architecture which supports a variety of plug-ins for knowledge modeling. The two core components of Protégé are the Protégé-frames and Protégé-OWL knowledge editors. The Protégé-frames editor enables users to create frame-based ontologies in accordance with the Open Knowledge Base Connectivity (OKBC)

protocols. The Protégé-OWL editor allows users to develop ontologies according to the W3C-recommended ontology modeling language, OWL.

As was mentioned in earlier sections of this work, the OWL plug-in of Protégé [Knublauch 2004] was used as the editor for this AsD ontology. In addition, the SWRL rule editor plug-in [O'Connor 2005], which resides within the OWL editor, was utilized to create additional Horn-like rules in order to make the AsD ontology more "robust" when a reasoning engine is used. Figure 7 displays the graphical user interface of the Protégé application. In this particular instance, the Jambalaya plug-in is displayed. Jambalaya is a tool used to visualize OWL ontologies using nodes and arcs.

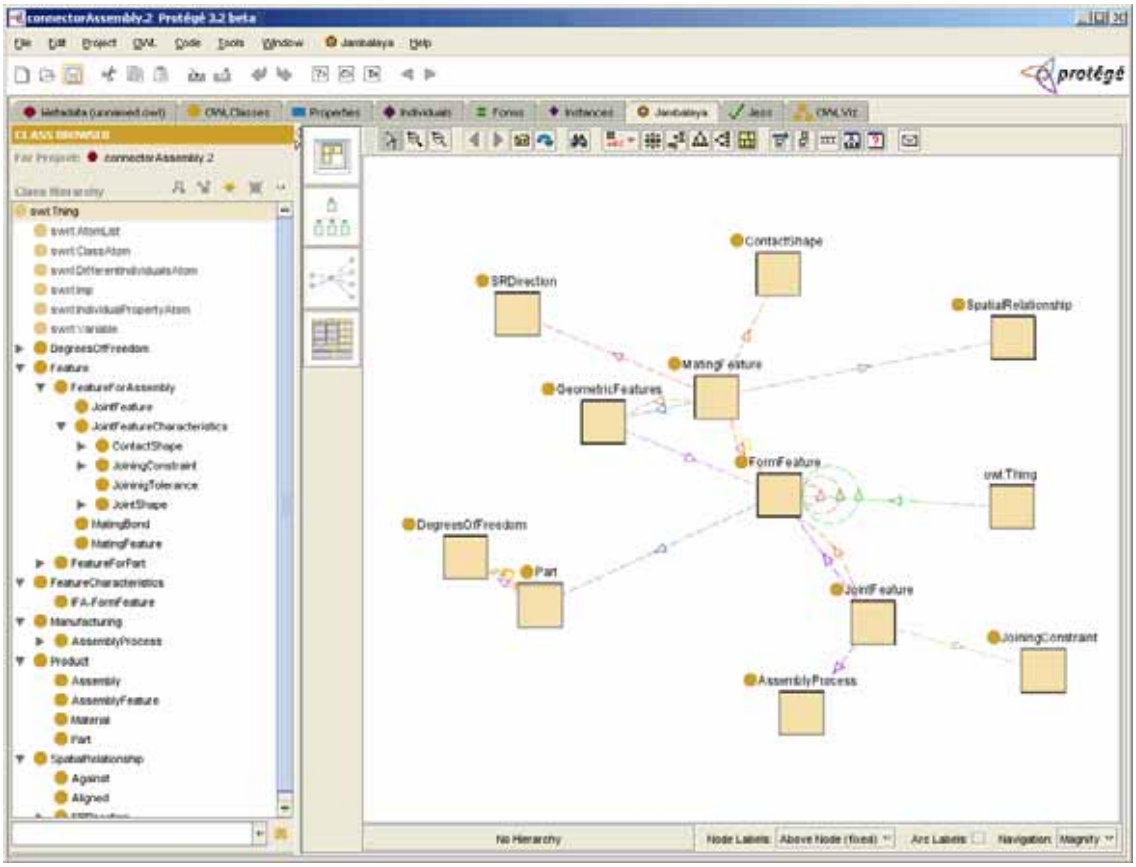


Figure 7. The Protégé user interface showing the Jambalaya plug-in.

4.2 REASONING ENGINE

The Bossam rule and reasoning engine developed by the Korean Electronics and Telecommunications Research Institute [Bossam 2005] was used to infer additional facts from the AsD ontology. As indicated earlier, additional facts within an ontology cannot be inferred without the use of inference engineering that utilizes the rules stored in the ontology.

In the case of this work ,Bossam was the reasoning engine of choice partially because the author was minimally familiar with other reasoning engines before this work began. Other reasoning tools such as Jess [Jess 2005] exist and have their strengths and weaknesses in terms of the logical deductions they make. Of all the reasoning engines available, Bossam is the least restrictive in its reasoning capabilities [Jang 2004].

4.3 APPLICATION DEVELOPMENT

Development of the AsD Browser utilized the Java programming environment. Because of its platform independence, the use of Java allows programmers to develop Semantic Web rules and content for a variety of systems. Moreover, many of the ontology development tools and API's are Java-based, making Java the appropriate language for ontology applications.

The Standard Widget Toolkit (SWT), developed by IBM and the Eclipse Foundation [Eclipse 2005], was used to create the GUI for the application. SWT is an alternative GUI toolkit to Sun's AWT and SWING toolkits. The benefit SWT provides is that it allows Java to utilized the native GUI components of the operating system that the application is running on. This means that the Java application will generally run faster. However, this has no bearing on the core functionality of the application, itself.

The core component used to manipulate the ontology was the Jena application program interface (API), developed by the HP Lab Semantic Web Program [Jena 2005]. Jena is an extensive API that allows users to create, store, manipulate, and reason RDF and OWL documents. In this work, Jena was used to search through and read in existing OWL ontologies as Java abstract data types.

5.0 IMPLEMENTATION

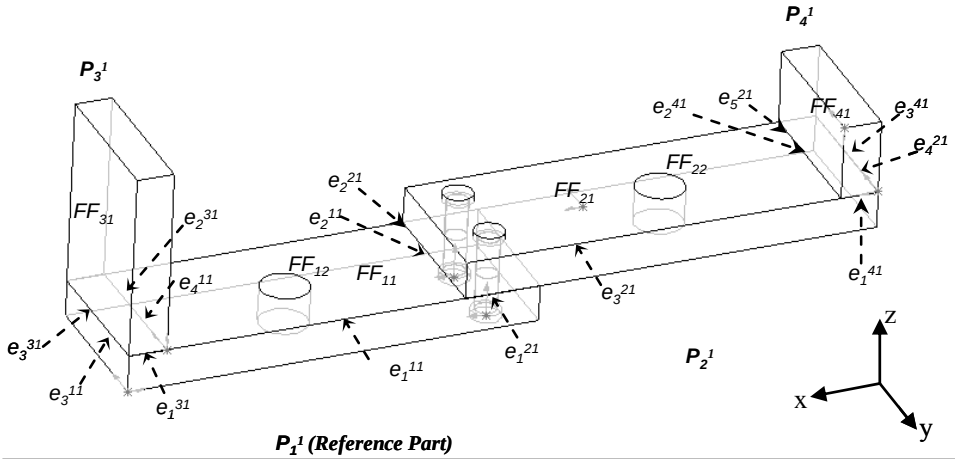
The following subsections detail the implementation of the AsD ontology in the OWL and SWRL syntax, making assembly models more suitable for the Semantic Web. Section 5.1 explains the construction of the data model. Section 5.2 shows how the ontology can be reasoned and used for intent analysis. Section 5.3 explains the development of a Java application used to view and query the assembly models and their semantic content.

5.1 THE ONTOLOGY

This section presents the ontology-based assembly design data model, or AsD ontology. In this data model, product characteristics, assembly characteristics, and joining processes are established in an ontology. Various collaborative designers, while using collaborative design tools, will then be able to access this assembly information by using a semantic query. The AsD ontology has been established based on an ARM, which was described in section 3; the ontology-based ARM serves as a meta-model that provides a link between geometric models and assembly relations. The AsD ontology also provides new facts through the use of the Bossam reasoning engine.

Throughout this section, the example connector assembly model in Figure 8 will be used. The connector assembly has two welded joints as well as one riveted joint. Notation for the

assembly model entities can be found in the figure. The model was originally represented in a pure XML format based on the design formalism in [Kim 2004]. This work enhances this model, using the developed AsD ontology. Note that each ARM form feature is linked to corresponding geometric entities of a design model in a SAT format.



e_i^{jk}	i^{th} edge of FF_{jk} (FFjk is Form Feature of Part (P_j^i) and P_j^i is a part of assembly (A_i))
P_1^1, P_2^1, P_3^1, and P_4^1	plate_a, plate_b, plate_c, and plate_d
FF_{11}	block (length, width, height) = block (L_{11}, W_{11}, H_{11}) = block (110, 40, 10)
FF_{21}	block (L_{21}, W_{21}, H_{21}) = block (110, 40, 10)
FF_{31}	block (L_{31}, W_{31}, H_{31}) = block (50, 40, 10)
FF_{41}	block (L_{41}, W_{41}, H_{41}) = block (20, 40, 10)
FF_{12}	hole (diameter, depth) = hole (DM_{12}, DT_{12}) = hole (12.81, 10)
FF_{22}	hole (DM_{22}, DT_{22}) = hole (12.81, 10)

Figure 8. Connector assembly example.

5.1.1 Ontology Terms

The definitions, possible terms, and concept representation of assembly models are investigated in this work. The investigated terms include Product, Assembly, Assembly Component, Part, Sub-assembly, Assembly Feature, Form Feature, Joint, Joint Feature, Mating Feature, and others. Table 2 shows selected terms and analyzed results. For example, the definition of an assembly feature in engineering design is "a group of assembly information," which includes form

features, joint features, mating relations, assembly/joining relations, spatial relationships, material, engineering constraints, etc.

Table 2. AsD ontology terms.

Terms	Definition for mechanical design domain	Other possible terms within the mechanical design domain	Represent-ation of concept	Content of representation of concept
Product	An item designed, developed, manufactured, and assembled to fill a functional need	Engineered product, mechanical product, consumer product, industrial product	Drawing, description, and specification	Specification of preferences, structure, materials, etc.
Assembly	A group of components brought together under specific conditions to form a self-contained unit and perform functions	Product breakdown structure	Drawing, description, and specification	Specification of components' geometry, spatial relationships, etc.
Part	An individually manufactured component	Component, assembly component	Drawing, description, and specification	Contains geometry, design intent
Assembly feature	A group of assembly information including form features and joint features associated with mating relations and assembly/joining relations, such that the association includes a set of spatial relationships between mating features, mating bonds, material, remaining degrees of freedom, as well as other constraints implied by the original intents on the form features	None	Data structure	Specification of spatial relationships, mating features, mating bonds, material, etc.
Form feature	A set of geometric entities (surfaces, edges, and vertices) together with specifications of the bounding relationship between them and which have engineering/functional implications and/or provide assembly aid, such as a center line of a hole on an object	None	Data structure	Specification of geometry, bounding relationship, and engineering function
Joint	The interface between two or more assembly components and how they are interfaced	None	Drawing, description, and specification	specification of geometry and joining conditions
Joint feature	A set of information including joining methods, groove shapes, joining components and entities, and joining constraints, which is used to represent assembly/joining relations	None	Data structure	Joining components, geometry, and constraints
Mating feature	A set of component geometric entities of form features which are needed to relatively locate the parts according to their spatial relationships in the whole product assembly	None	Data structure	Geometry

An ontology consists of classes, properties, and instances of classes. A class defines a concept. Individuals are instances of classes and are linked to classes via properties. Properties can be used to state relationships between individuals, or between individuals and data values. Based on engineering design terms, classes and their characteristics are defined and properties and instances are extracted. Figure 9 illustrates the hierarchy of AsD ontology classes. In the hierarchy, a class is a subclass of another class where the broadest class is "thing". Note that "thing" is a necessary class for an OWL ontology is inserted by default.

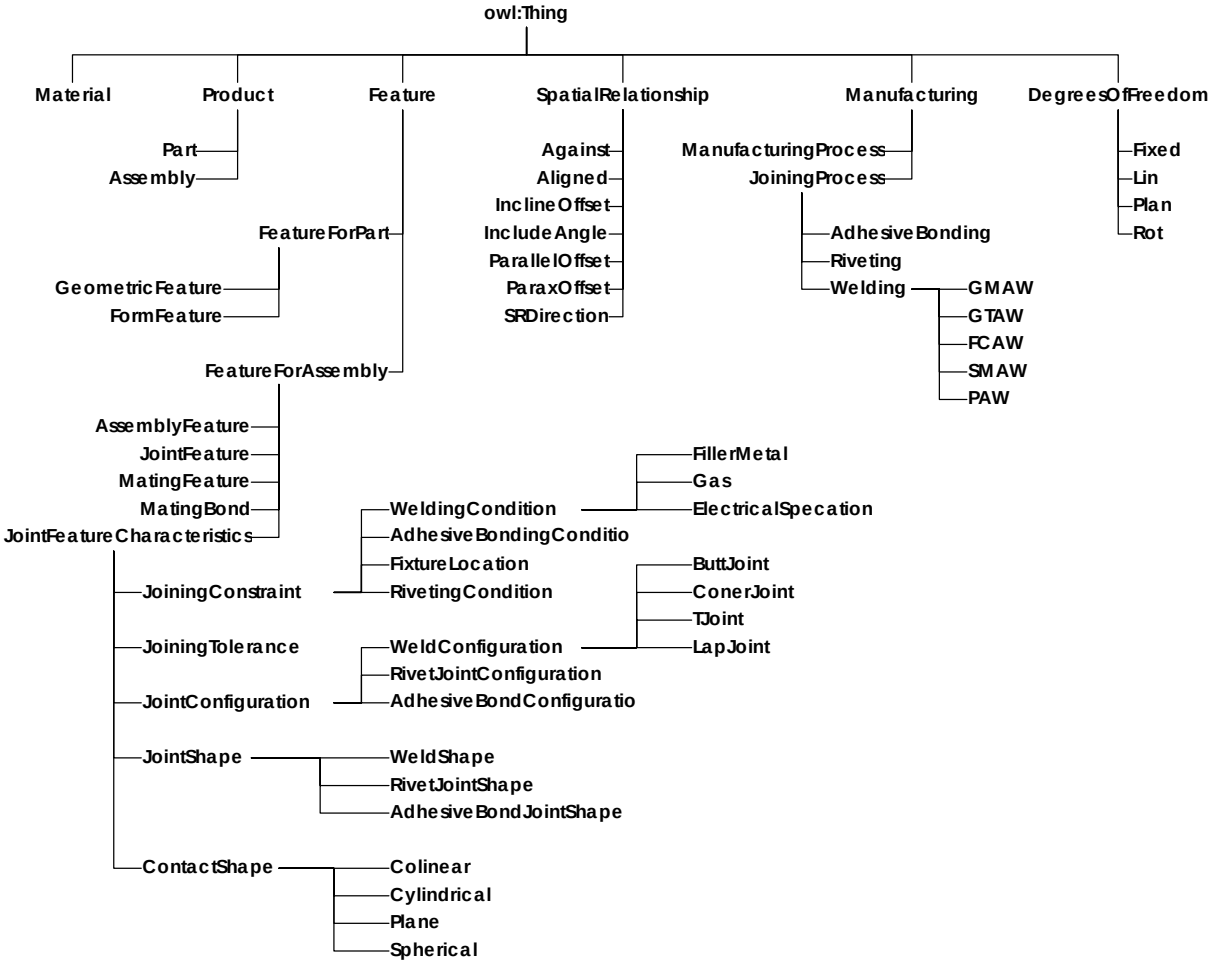


Figure 9. Class hierarchy of the AsD Ontology.

The developed ontology contains detailed classes of welding parameters among many joining processes. By utilizing such a means of capturing joining constraints, the transition from the CAD environment to the CAE environment can be further automated, while the joining information specified on geometric entities remains shareable along with the assembly geometry. Further benefits of capturing joining information in such a manner include the increased sharability of joint specifications among diverse computer programs, which in turn facilitates sharing among distributed collaborators and supply chain members. Welding concepts for the ontology are based on the definitions put forth by the American Welding Society [AWS 2001], the National Institute of Standards and Technology [Rippey 2004], and the work of Yao et al. [Yao 1998]; however, not all definitions and specifications presented by the aforementioned researchers were used in this work.

Table 3 shows example properties that represent class characteristics, domains, and ranges that represent class relationships. For example, the domain of the "belongTo" property is FormFeature and its range is Part.

Table 3. Class properties.

Property	Domain	Range
belongTo	FormFeature	Part
belongToFormFeature	GeometricFeatures	FormFeature
hasMaterial	Part	Material
hasPart	Assembly	Part
referenceDirection	MatingFeature	SRDirection
hasMatingComponent	MatingFeature	FormFeature

As mentioned in the previous section, OWL is used to represent ontology triplets. This language is designed for use by applications that need to process the content of the information

instead of just presenting the information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, Resource Definition Framework (RDF), and RDF Schema (RDF-S), by providing additional vocabulary along with formal semantics. In 2004, the W3C made OWL a recommendation for Semantic Web technology [W3C-OWL 2004]. Although OWL was investigated, the current constraint representation capability of OWL alone was not adequate to implement relationships in assembly design; thus, it was augmented with SWRL rules.

This research shows that SWRL rules can be reasoned to accommodate potential semantic queries and information requests in a collaborative assembly design environment. To generate the AsD ontology and rules, Protégé's SWRL editor is used for this work. Figure 10 illustrates the feature hierarchy its and associated assembly relationships for the connector assembly from Figure 8. Only geometric features that are related to the assembly relationships are included in this figure. For example, form features FF_{11} and FF_{21} have a "inter-featureAssociation" relation. This figure is used to illustrate how the ontology can be reasoned and used in collaborative assembly design.

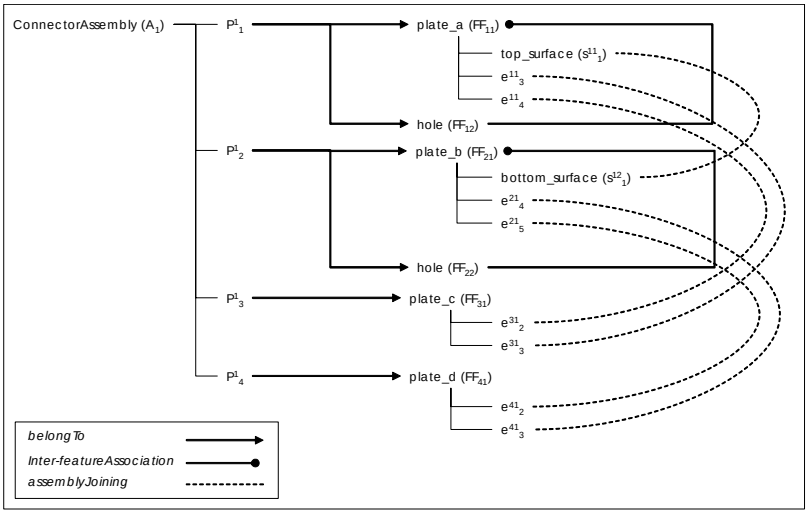


Figure 10. Feature hierarchy of the connector assembly.

In this research, the three types of assembly relationships (i.e. *belongTo*, *inter-featureAssociation*, and *assembly/joining relations*), spatial relationships, and joining relations are represented in OWL and SWRL.

The first core constraints in the AsD ontology are the assembly relationships from form feature to form feature and from form feature to part. These assembly constraints are represented explicitly using OWL triplets and SWRL rules. The basic assumptions of these definitions are 1) **A** is a set of assembly structure, 2) **P** is a set of parts, and 3) **FF** is a set of form features. Function comp( ) is defined as a function that obtains elements of each object. For example, comp(A_i) obtains part $P_i \in$ **P** that constructs assembly $A_i \in$ **A** [Kim 2004].

5.1.2 BelongTo Relations

A belongTo relation defines relations between a part and a form feature. Assembly relations between features as well as between features and parts are defined below. The belongTo relations infer two constraints (C1-1 and C1-2 below).

Table 4. Summary of the belongTo relationship.

Definition of belongTo (from [Kim 2004])
A part P_j^i and a form feature FF_{jk} are said to have a belongTo relation, $B_{jk}^{(i)}$: $FF_{jk} :\rightarrow P_j^i$, k = 1, 2, …, n, **if** $P_j^i \in$ comp(A_i), j = 1, 2, …, m; and $FF_{jk} \in$ comp (P_j^i).
Implied Constraints
C1-1: Every form feature must belong to a part
C1-2: A form feature must not belong to two parts

The inferred constraints can be transformed to two asserted conditions (using Protege). For the C1-1, an asserted condition, "∃ FormFeature belongTo Part" is used and an asserted

condition, "belongTo = 1" is used to represent C1-2. This cardinality condition means the property belongTo has exactly one value.

5.1.3 Inter-featureAssociation Relations

The inter-featureAssociation relation represents the relations among form features. The relational constraint (RC_{pq}) stands for the relationship between two form features in the form feature hierarchy. For example, a block (FF_{jq}) may have a blind hole (FF_{jp}) at a certain location. The distance between the coordinates of the block and the blind hole is a dimensional constraint. Since the block form feature contains the hole form feature (since the block is a parent feature of the hole in the feature decomposition hierarchy), the relational constraint (RC_{pq}) is 0. If FF_{jp} has some form feature (FF_{jq}), then RC_{pq} is 1. When two form features (FF_{jp} and FF_{jq}) do not belong to each other (e.g. two holes in a block), RC_{pq} is 2. The inter-featureAssociation implies three constraints (C2-1 to C2-3 below).

Table 5. Summary of the inter-featureAssociation relationship.

Definition of inter-featureAssociation (from [Kim 2004])
A form feature FF_{jp} and another form feature FF_{jq} are said to have an inter-featureAssociation relation, $I_{pq}^{(j)}$: $FF_{jp} \Leftrightarrow FF_{jq}$, p =1,2, …, n, q =1,2, …, l, **if** $P_j^i \in$ comp (A_i), j = 1, 2, …, m; FF_{jp}, $FF_{jq} \in$ comp (P_j^i) (i.e. FF_{jp}, $FF_{jq} :\rightarrow P_j^i$); DC_r and RC_{pq} are satisfied, where $r \in IDI_{pq}^{(j)}$; and $IDI_{pq}^{(j)}$ is an index set depending upon this pair, FF_{jp} and FF_{jq} .
Implied Constraints
C2-1: The associated form features must not be identical (non-equivalent)
C2-2: The associated form features must belong to tne same part
C2-3: The relational constraint must be represented and included
SWRL Rules (from [Kim 2006])
FormFeature(?x) ^ FormFeature(?y) ^ Part(?z) ^ differentFrom(?x, ?y) ^ belongTo(?x, ?z) ^ belongTo(?y, ?z) ^ RelationalConstraint_0(?a) → inter-featureAssociation(?x, ?y)
FormFeature(?x) ^ FormFeature(?y) ^ Part(?z) ^ differentFrom(?x, ?y) ^ belongTo(?x, ?z) ^ belongTo(?y, ?z) ^ RelationalConstraint_1(?a) → inter-featureAssociation(?y, ?x)
FormFeature(?x) ^ FormFeature(?y) ^ Part(?z) ^ differentFrom(?x, ?y) ^ belongTo(?x, ?z) ^ belongTo(?y, ?z) ^ RelationalConstraint_2(?a) → inter-featureAssociation(?x, ?y)

5.1.4 Assembly/joining relations.

The assembly/joining relations represent the relations between form features that belong to different parts. It implies two constraints (C3-1 and C3-2 below) and the implied constraints are represented using a SWRL rule.

Table 6. Summary of assembly/joining relations.

Definition of assembly/joining relations (from [Kim 2004])
A form feature FF_{gp} and another form feature FF_{hq} are said to have an assembly/joining relation, $\vartheta_{pq}^{(gh)}$: $FF_{gp} \otimes FF_{hq}$, **if** P_g^i and $P_h^i \in$ comp (A_i), g = 1, 2, …, m_1, h = 1, 2, …, m_2; $FF_{gp} \in$ comp (P_g^i), p = 1, 2, …, l_1; $FF_{hq} \in$ comp (P_h^i), q = 1, 2, …, l_2; FF_{gp}, $FF_{hq} \in J$; MF_{r1}, $JF_{r2} \in J$; and DC_{r3} is satisfied, where r1 $\in JMI_{pq}^{(gh)}$, r2 $\in JJI_{pq}^{(gh)}$, and r3 $\in JDI_{pq}^{(gh)}$; and $JMI_{pq}^{(gh)}$, $JJI_{pq}^{(gh)}$, and $JDI_{pq}^{(gh)}$ are index sets depending upon this pair, FF_{gp} and FF_{hq} .
Implied Constraints
C3-1: The associated form features must belong to two non-equivalent parts
C3-2: The associated form features must be a joining pair
SWRL Rule (from [Kim 2006])
FormFeature(?x) ^ FormFeature(?y) ^ Part(?z) ⋀ Part(?a) ^ belongTo(?x, ?z) ^ belongTo(?y, ?a) ^ differentFrom(?z, ?a) ^ isJointPair(?x, ?y) → assemblyJoiningRelationship(?x, ?y)
Inferred Facts:
assemblyJoiningRelationship (FormFeature_11, FormFeature_21)
assemblyJoiningRelationship (FormFeature_31, FormFeature_11)
assemblyJoiningRelationship (FormFeature_21, FormFeature_41)

5.1.5 Spatial Relationship Reasoning

Depending upon the spatial relationships (SR) of two give parts, the DOF can be inferred. This paper shows that the DOF can be inferred by using SWRL rules. The SWRL rules used to represent SRs and DOF and inference examples are described in this section.

1) Two parts, which have and "against" SR between planar surfaces along to z-direction, have DOF of {Plan_Z::Rot_Z}—in other words, plane DOF along the z-direction of the coordinate of the reference part and rotational (rot) DOF within its own coordinate system (z-direction).

Table 7. Planar spatial relationship.

SWRL Rules (from [Kim 2006])
MatingFeature(?x) ^ FormFeature(?y) ^ Part(?z) ^ belongTo(?y, ?z) ^ hasMatingComponent(?x, ?y) ^ hasSpatialRelationship(?x, against) ^ hasContactShape(?x, Plane_Z) ^ ReferenceDirection(?x, Z-Direction) → hasDOF(?z, Plan_Z)
MatingFeature(?x) ^ FormFeature(?y) ^ Part(?z) ^ belongTo(?y, ?z) ^ hasMatingComponent(?x, ?y) ^ hasSpatialRelationship(?x, Against) ^ hasContactShape(?x, Plane_Z) ^ ReferenceDirection(?x, Z-Direction) → hasOwnDOF(?z, Rot_Z)
Inferred Facts
hasDOF(Part_12,Plan_Z_1);
hasOwnDOF(Part_14, Rot_Z_1);
hasDOF(Part_13, Plan_Z_1);
hasOwnDOF(Part_12, Rot_Z_1);
hasDOF(Part_14, Plan_Z_1);
hasOwnDOF(Part_13,Rot_Z_1);

2) Two parts, which have "aligned" SR between collinear lines along the x-direction, have DOF of {Lin_X::Rot_X}. In other words, linear DOF along the x-direction the coordinate system of the reference part, and rotational DOF within its own coordinate system .

Table 8. Aligned spatial relationship.

SWRL Rules (from [Kim 2006])
MatingFeature(?x) ^ FormFeature(?y) ^ Part(?z) ^ belongTo(?y, ?z) ^ hasMatingComponent(?x, ?y) ^ hasSpatialRelationship(?x, Aligned) ^ hasContactShape(?x, Collinear) ^ ReferenceDirection(?x, X-Direction) → hasDOF(?z, Lin_X)
MatingFeature(?x) ^ FormFeature(?y) ^ Part(?z) ^ belongTo(?y, ?z) ^ hasMatingComponent(?x, ?y) ^ hasSpatialRelationship(?x, Aligned) ^ hasContactShape(?x, Colinear) ^ ReferenceDirection(?x, X-Direction) → hasOwnDOF(?z, Rot_X)
Inferred Facts
hasDOF(Part_12,Lin_X_1);
hasOwnDOF(Part_12, Rot_X_1);
hasDOF(Part_13, Lin_X_1);
hasOwnDOF(Part_13, Rot_X_1);
hasDOF(Part_14, Lin_X_1);
hasOwnDOF(Part_14, Rot_X_1);

3) A part, which has rotational DOF and two linear DOF, has fixed designed DOF. The corresponding SWRL rules and facts inferred by the rules are the following:

Table 9. Fixed DOF.

SWRL Rule (from [Kim 2006])
Part(?x) ^ hasDOF(?x, Rot_Z) ^ hasDOF(?x, Lin_X) ^ hasDOF(?x, Lin_Y) ^ hasDOF(?x,Fixed_in) → hasDesignedDOF(?x, Fixed_In)
Inferred Facts
hasDesignedDOF(Part_12, Fixed_1);
hasDesignedDOF(Part_14, Fixed_1);

4) Each joining method also implies a reduction of DOF. For example, a part joined by welding has fixed DOF, which is inferred by welding. The corresponding SWRL rules and facts inferred by the rules are following:

Table 10. Inferred DOF of welding.

SWRL Rule (from [Kim 2006])
JointFeature(?y) ^ FormFeature(?z) ^ Part(?b) ^ isJoiningMethod(?y, Welding) ^ hasJoiningComponents(?y, ?z) ^ belongTo(?z, ?b) → hasJoiningInferredDOF(?b, fixed_In)
Inferred fact
hasJoiningInferredDOF(Part_13, Fixed_1);
hasJoiningInferredDOF(Part_14, Fixed_1);

5) If a part is joined by more than one rivet, the part has fixed DOF. This fact can be inferred by the rivets. The corresponding SWRL rules and facts inferred by the rules follow as:

Table 11. Inferred DOF of riveting.

SWRL Rule (from [Kim 2006])
JointFeature(?y) ^ FormFeature(?z) ^ Part(?b) ^ isJoiningMethod(?y, Riveting) ^ hasJoiningComponents(?y, ?z) ^ belongTo(?z, ?b) ^ RivetingCondition(?c) ^ hasJoiningConstraint(?y, ?c) ^ NumberOfRivet(?c, ?d) ^ swrlb:greaterThanOrEqual(?d, 2) → hasJoiningInferredDOF(?b, fixed_In)
Inferred Facts
hasJoiningInferredDOF(Part_11, Fixed_1);
hasJoiningInferredDOF(Part_12, Fixed_1);

5.2 INTENT ANALYSIS

In assembly design, spatial relationships (SR) can be assigned to achieve the intended DOF of parts relative to one another. These designed SRs should be realized and maintained in assembly design, and eventually in the physical assembly by joining. However, the designed DOF often are not maintained persistently during a distributed collaborative design process [Kim 2004]. Each SR can be interpreted as a constraint imposed on the DOF between the relative mating or interacting features. Given a set of SR, the resultant DOF can be inferred. In other words, any allowable motion for parts has to follow a path along the directions specified by the DOF in order to maintain their spatial relationships.

Each joining method infers specific SRs, and the corresponding DOF are implied by these spatial relationships [Kim 2006]. The designer's original intent imposed on assembly design can be analyzed by comparing the implied DOF and the designed DOF. For example, when designers want to permanently join two plates, they assign SRs to fix those plates. If a designer considers a welded joint and specifies a welding operation as a joining method, then the DOF corresponding to the welding operation can be inferred by using the aforementioned ontology reasoning capability, and used to check whether this welding operation will satisfy the designer's intent for the assembly. The welding operation causes 1) an "against" SR between the mating faces, 2) an "aligned" SR between joining entities on the weld seam, and 3) the two assembly components (two plates) to lose all DOF and become fixed [Kim 2006]. In this case, the specified joining method (welding) fully satisfies the designed DOF.

As an assembly design evolves, the designer specifies a series of SRs and implicitly imposes DOF (i.e. designed DOF). Using the AsD ontology, the designed DOF can be persistently maintained and the information can be used to compare the original assembly design intent with DOF. For example, when designers want to permanently join two plates, they assign SRs to fix those plates. Since the design intent and implication of the joining process are persistently captured in the AsD ontology, the intent analysis can be conducted in a collaborative assembly design environment.

5.3 THE BROWSER APPLICATION

Typically, to obtain the aforementioned query information, corresponding CAD software needs to be acquired or a corresponding CAD file needs to be translated into neutral formats (e.g. STEP and IGES). Often, core assembly information (e.g. assembly and joining relations) is lost through this translation. In the presented framework, assembly entities in the ontology are persistently connected to a corresponding solid model by the ARM, which is format and application independent. In this manner, the designers do not need to know a specific CAD format, and loss of information is relatively low, since assembly information is represented explicitly in the ARM.

The interface of the assembly design browser is shown in Figure 11. Using the interface, designers can search for assembly models that contain relevant assembly information. The browser provides three primary views of the assembly information: a graphical geometric view of the assembly, a hierarchical view of the ontology classes and of the properties that relate all

pertinent assembly model data, and an XML view that shows the raw XML format and structure of the model data. While the hierarchical and XML views are non-typical views for design environments, they do give the user the option to inspect non-visual aspects of the assembly model for such things as joining conditions and material properties. By viewing the hierarchy, users can manually determine which classes are explicitly represented by the model and their sub-class relationships. The XML view gives the user the opportunity to peruse the format in which the model will be transferred from system to system, similar in some respects to viewing the ASCII contents of an IGES file, for example.

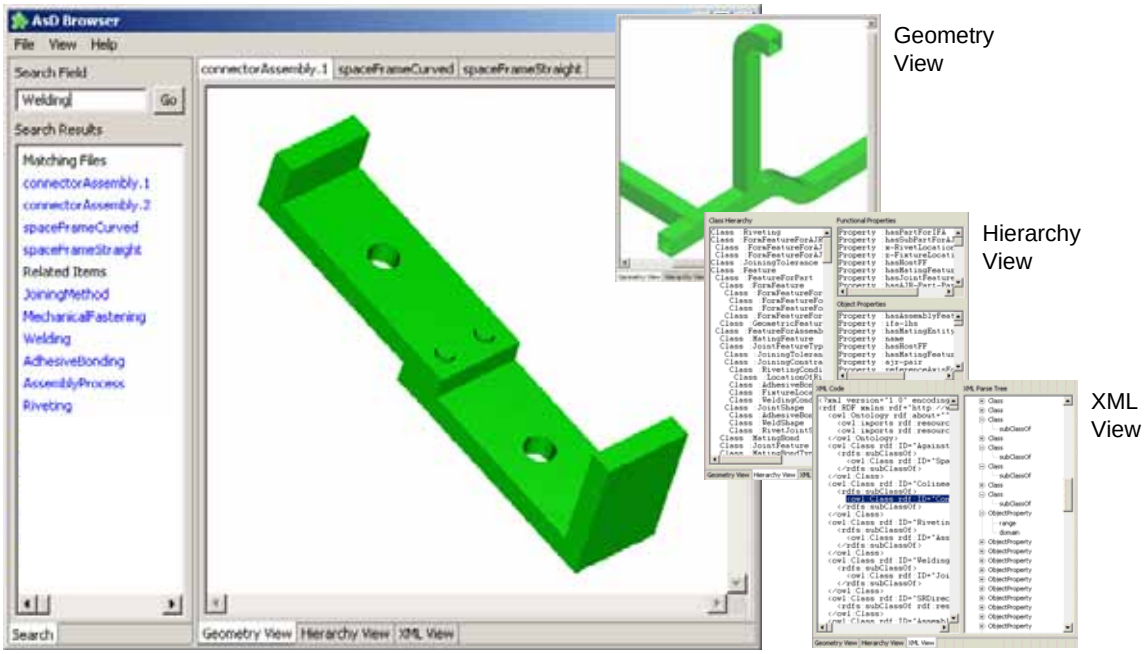


Figure 11. The AsD Browser application GUI.

The assembly design browser searches for relevant assembly entities through the use of a keyword search that examines file names as well as class structure of the associated ontology model. If the search criteria are met, the relevant assembly model returns files that contain the specified assembly information. In addition to the file results that are returned, supplementary

search criteria are also displayed so that the designer may make additional searches for similar items. For example, if a designer searches for assemblies that contain welding information showed in Figure 11, additional search criteria such as riveting or adhesive bonding may be returned by utilizing the ontology model, since they are also joining methods, which is a result of a semantic query.

Relevant terms are found by tracing the properties to related classes. In the case mentioned in the paragraph above, the sibling classes to the welding class were retrieved. In addition to tracing the explicit properties to related classes, it is possible to follow inferred properties generated by a reasoning engine. For example, if one queries for classes related to welding, one could review the explicit intended DOF for the assembly components (hasDesignedDOF) versus the implicit inferred remaining DOF due to the welding joining method (hasJoiningInferredDOF). In short, queries can trigger a reasoning engine to generate additional inferred relations that can be used to locate related classes.

6.0 CONCLUSION

In conclusion, this work has presented a new data model for representing assembly models in CAD environments. The new model uses semantic and ontology technologies to enhance the information and knowledge that can be captured by such a model, as well as the improved access and sharability of such information. Furthermore, by including more abstract engineering information, such as that of joining method selection and joint design, assemblies that graphically appear to be the same geometrically and topologically can now be differentiated by the abstract engineering knowledge that is attached to them (e.g. weld joints and adhesively bonded joints may look similar, but the joining information differentiates them).

The benefits of using semantic technologies to represent CAD assemblies help to promote horizontal and vertical integration within manufacturing organizations and their supply chains. Better horizontal integration, is achieved since the structure of the RDF and OWL languages allow for increased sharing of portions of an assembly model as well as the decentralization of the model due to the links to distributed resources. Better vertical integration is achieved by allowing more abstract engineering information and non-visual joining parameters to be attached to the model. This allows design decisions and information other than that which is represented geometrically to be propagated to downstream activities consistently. Non-visual information, such as the joining parameters at the interfaces between parts, is important for the

analyst to make accurate simulation models and is important for the manufacturing and processing to ensure design feasibility.

This work has produced an initial ontology for representing mechanical assemblies, thus,enabling more selective querying and sharing of engineering designs. As designs become more intricate with increasing complexities at the interfaces between parts, the need for a means to consistently capture and selectively share information in increasingly collaborative design environments becomes more important. It is believed that the AsD ontology developed here is a good starting point.

To be accepted and effectively utilized in the commercial design community, the terms, definitions, and relationships in the AsD ontology must by developed and agreed upon by experts in the domain. It is not very plausible for one person or organization to develop a concept map that the entire design community would agree on. The ontology presented in this work is not intended to be a final solution for a semantic assembly data model, but rather an example of how semantic technologies can be used to enhance assembly models in CAD systems.

Security issues with respect to the sharing of engineering design information were not addressed in this work. Since a greater amount of design information can now be attached to assembly models, there is potentially an increased risk of revealing more proprietary information than before. Before this AsD ontology can confidently be used the necessary security procedures and protocols must be investigated.

This work does not take into consideration computational performance and processing issues that may effect computational resources. Because the new AsD ontology data model is a text-based rather than binary-based data model, file sizes are much larger plain CAD geometry

files. Furthermore, since XML syntax utilizes tags to identify specific pieces of data, the number of tags outnumbers the pieces of data actually present in the assembly model. Nonetheless, it is believed that the benefits of the addition of semantic content to assembly models far outweighs the larger file size and increased processing resources needed to deal with text-based files. The semantic nature of the new assembly models allows resources to be linked in a distributed environment, promoting selective sharing of only portions of a model, rather than the unnecessary sharing of complete models.

As mentioned above, the AsD ontology developed in this work is a starting point for a semantic assembly design model. The most notable limitation at this stage is that instances of assembly models must be generated manually rather than through the used of a CAD environment. There is a need to develop a CAD system that produces entire CAD assembly models in the OWL format or to automatically generate solid geometry, which is linked to a semantic OWL file that contains more abstract engineering information. Being able to generate semantic content in an automated manner is crucial for industry acceptance since manual creation is time-consuming and the majority of design engineers will not have in-depth experience with semantic modeling.

The amount of information and the number of concepts represented in the AsD ontology is not comprehensive and should be extended in order to cover a larger range of design domains. In this work, only several joining methods were considered. Expansion of the number of engineering concepts represented in the AsD ontology is not technically difficult due to the nature of the XML, RDF, and OWL technologies; however, defining the terms and the relationships among those terms is an activity in which many from the design community should

partake. If standard definitions cannot be agreed upon, then semantic and contextual modeling of assemblies cannot be achieved in a widespread manner.

Applications of the AsD ontology need to be investigated to further promote the advantages of semantic assembly modeling. For example, since joining information can be attached to specific entities, analysis and simulation applications should be able to extract that information and set up simulation models in a more automated fashion. What is more, concepts modeled in the AsD ontology can potentially be mapped to concepts in other ontologies, such as the engineering analysis ontology developed by Grosse et. al. [Grosse 2005]. Utilization of concepts from multiple ontologies would be a prime example of the use of semantics to improve product development process productivity.

APPENDIX

SAMPLE OWL & SWRL CODE FOR THE CONNECTOR ASSEMBLY

The first listing of code shows the beginning portion of the OWL ontology file for the AsD model. In this beginning portion, one can see the definition of some OWL classes and properties. Note that the Protégé editor produces the OWL code such that the classes are in alphabetical order.

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
    <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
    <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY daml "http://www.daml.org/2001/03/daml+oil#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY swrlImport "http://www.daml.org/rules/proposal/swrl.owl#" >
    <!ENTITY swrlbImport "http://www.daml.org/rules/proposal/swrlb.owl#" >
]>


<rdf:RDF xmlns="http://www.owl-ontologies.com/unnamed.owl#"
     xml:base="http://www.owl-ontologies.com/unnamed.owl"
     xmlns:swrlbImport="http://www.daml.org/rules/proposal/swrlb.owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:swrl="http://www.w3.org/2003/11/swrl#"
     xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
     xmlns:swrlImport="http://www.daml.org/rules/proposal/swrl.owl#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Ontology rdf:about="">
        <owl:imports rdf:resource="http://www.daml.org/rules/proposal/swrlb.owl"/>
        <owl:imports rdf:resource="http://www.daml.org/rules/proposal/swrl.owl"/>
    </owl:Ontology>
    <owl:Class rdf:ID="AdhesiveBondingCondition">
        <rdfs:subClassOf rdf:resource="#JoiningConstraint"/>
    </owl:Class>
```

```
    <owl:Class rdf:ID="AdhesiveBondJointShape">
        <rdfs:subClassOf rdf:resource="#JointShape"/>
    </owl:Class>
    <owl:Class rdf:ID="Against">
        <rdfs:subClassOf rdf:resource="#SpatialRelationship"/>
    </owl:Class>
    <Against rdf:ID="Against_1"/>
    <owl:Class rdf:ID="Aligned">
        <rdfs:subClassOf rdf:resource="#SpatialRelationship"/>
    </owl:Class>
    <Aligned rdf:ID="Aligned_1"/>
    <owl:Class rdf:ID="Assembly">
        <rdfs:subClassOf rdf:resource="#Product"/>
    </owl:Class>
    <owl:Class rdf:ID="AssemblyFeature">
        <rdfs:subClassOf rdf:resource="#Product"/>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="assemblyJoiningRelationship">
        <rdfs:domain rdf:resource="#FormFeature"/>
        <rdfs:range rdf:resource="#FormFeature"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="AssemblyProcess">
        <rdfs:subClassOf rdf:resource="#Manufacturing"/>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="belongTo">
        <rdfs:domain rdf:resource="#FormFeature"/>
        <rdfs:range rdf:resource="#Part"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="belongToFormFeature">
        <rdfs:domain rdf:resource="#GeometricFeatures"/>
        <rdfs:range rdf:resource="#FormFeature"/>
    </owl:ObjectProperty>
…
```

The second listing of code shows a portion of an instance of the connector assembly.

```
…
<GeometricFeatures rdf:ID="S_11_1">
    <belongToFormFeature rdf:resource="#FormFeature_11"/>
</GeometricFeatures>
<GeometricFeatures rdf:ID="S_21_1">
    <belongToFormFeature rdf:resource="#FormFeature_21"/>
</GeometricFeatures>
<GeometricFeatures rdf:ID="S_21_2">
    <belongToFormFeature rdf:resource="#FormFeature_21"/>
</GeometricFeatures>
<GeometricFeatures rdf:ID="S_31_1">
    <belongToFormFeature rdf:resource="#FormFeature_31"/>
</GeometricFeatures>
<GeometricFeatures rdf:ID="S_41_1">
    <belongToFormFeature rdf:resource="#FormFeature_41"/>
</GeometricFeatures>
…
```

The last listing in this appendix is an example of an SWRL rule that enhances the ontology even further.

```
…
<swrl:Imp rdf:ID="Rule-4">
    <swrl:head>
        <rdf:List>
            <rdf:first>
```

```
                <rdf:Description>
                    <rdf:type rdf:resource="&swrl;IndividualPropertyAtom"/>
                    <swrl:argument2 rdf:resource="#a"/>
                    <swrl:argument1 rdf:resource="#x"/>
                    <swrl:propertyPredicate rdf:resource="#isJoiningMethod"/>
                </rdf:Description>
            </rdf:first>
            <rdf:rest rdf:resource="&rdf;nil"/>
        </rdf:List>
    </swrl:head>
    <swrl:body>
        <rdf:List>
            <rdf:first>
                <rdf:Description>
                    <rdf:type rdf:resource="&swrl;ClassAtom"/>
                    <swrl:argument1 rdf:resource="#x"/>
                    <swrl:classPredicate rdf:resource="#JointFeature"/>
                </rdf:Description>
            </rdf:first>
            <rdf:rest>
                <rdf:List>
                    <rdf:first>
                        <rdf:Description>
                            <rdf:type rdf:resource="&swrl;ClassAtom"/>
                            <swrl:argument1 rdf:resource="#a"/>
                            <swrl:classPredicate rdf:resource="#Riveting"/>
                        </rdf:Description>
                    </rdf:first>
                    <rdf:rest>
                        <rdf:List>
                            <rdf:first>
                                <rdf:Description>
                                    <rdf:type rdf:resource="&swrl;ClassAtom"/>
                                    <swrl:argument1 rdf:resource="#c"/>
                                    <swrl:classPredicate rdf:resource="#RivetingConditio…
                                </rdf:Description>
                            </rdf:first>
                            <rdf:rest>
                                <rdf:List>
                                    <rdf:first>
                                        <rdf:Description>
                                            <rdf:type rdf:resource="&swrl;IndividualProp…
                                            <swrl:argument2 rdf:resource="#c"/>
                                            <swrl:argument1 rdf:resource="#x"/>
                                            <swrl:propertyPredicate rdf:resource="#hasJo…
                                        </rdf:Description>
                                    </rdf:first>
                                    <rdf:rest rdf:resource="&rdf;nil"/>
                                </rdf:List>
                            </rdf:rest>
                        </rdf:List>
                    </rdf:rest>
                </rdf:List>
            </rdf:rest>
        </rdf:List>
    </swrl:body>
</swrl:Imp>
…
```

BIBLIOGRAPHY

[Alesso 2005] Alesso H.P., Smith C. F., *Developing Semantic Web Services*, A K Peters Ltd, 2005.

[AWS 2001] American Welding Society, *A3.0-01: Standard Welding Terms and Definitions*, 2001.

[AWS 2003] American Welding Society, *Vision for Welding Industry*. AWS Report, http://www.aws.org/research/vision.pdf, 2003.

[AWS-EWI 2001] American Welding Society, Edison Welding Institute, *Economic Impact and Productivity of Welding: Heavy Manufacturing Industries*, AWS Report, June 2001.

[Berners-Lee 2000] Berners-Lee T. "Semantic Web – XML2000," World Wide Web Consortium Presentation, http://www.w3.org/2000/Talks/1206-xml2k-tbl, 2000.

[Berners-Lee 2001] Berners-Lee T., Hendler J., Lassila O., "The Semantic Web," *Scientific American*, pp. 34-43, May 2001.

[Bossam 2005] The Korean Electronics and Telecommunications Research Institute, "Bossam Rule/OWL Reasoner," http://mknows.etri.re.kr/bossam, 2005.

[Brandon 1997] Brandon D., Kaplan W.D., *Joining Processes: An Introduction*, John Wiley and Sons Inc, 1997.

[Brunnermeier 1999] Brunnermeier S.B., Martin S.A., *Interoperability Cost Analysis of the US Automotive Supply Chain*, NIST Technical Report, Research Triangle Institute Project Number 7007-03, March 1999.

[CYC 2005] Cycorp, Inc. "The Cyc Knowledge Base," http://www.cyc.com/cyc/technology/whatiscyc_dir/whatsincyc, 2005.

[Daconta 2003] Daconta M.C., Obrst L.J., Smith K.T., *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*, John Wiley and Sons Inc, 2003.

[DAML 2002] DARPA Agent Markup Language, "Language Feature Comparison," http://www.daml.org/language/features.html, 2002.

[Eclipse 2005] The Eclipse Foundation, *Eclipse*, http://www.eclipse.org, 2005.

[Fensel 2001] Fensel D. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*, Springer-Verlag Berlin Heidelberg, 2001.

[Fox 1992] Fox M.S., *The TOVE Project: Towards a Common-Sense Model of the Enterprise*, Enterprise Integration Laboratory Technical Report, 1992.

[Fox 1998] Fox M.S., Gruninger M., "Enterprise modeling," *AI Magazine* pp.109-21, Fall 1998.

[Fuh 2005] Fuh J.Y.H., Li W.D., "Advances in collaborative CAD: the state of the art," *Compute-Aided Design*, vol. 37, pp. 571-581, 2005.

[Golbreich 2005] Golbreich C., Bierlaire O., Dameron O., Gibaud B., "What reasoning support for ontology and rules? The brain case study," *8th International Protégé Conference, Protégé with Rules Workshop*, Madrid, Spain, SMI-2005-1079, 2005.

[Grosse 2005] Grosse I.R., Milton-Benoit J.M., Wileden J.C., "Ontologies for supporting engineering analysis models," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 19, pp. 1-18, 2005.

[Gruber 1993] Gruber T.R., "A translation approach to portable ontology specification," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220, 1993.

[Gruninger 2003] Gruninger M., Sriram R.D., Cheng J., Law K., "Process specification language for project information exchange," *International Journal of Information Technology in Architecture, Engineering, and Construction*, vol. 1, pp. 307-28, 2003.

[Horrocks 2005] Horrocks I., Patel-Schneider P.F., Bechhofer S., Tsarkov D., "OWL Rules: A Proposal and Prototype Implementation," *Journal of Web Semantics*, vol. 3, no. 1, pp. 23-40, 2005.

[Horváth 1998] Horváth I., Pulles J.P.W., Bremer A.P., Vergeest J.S.M., "Towards an ontology-based definition of design features," *SIAM Workshop on Mathematical Foundations for Features in Computer Aided Design, Engineering, and Manufacturing*, 1998.

[Jang 2004] Jang M., Sohn J.C., "Bossam: an extended rule engine for the web," *Proceedings of RuleML 2004*, LNCS3323, pp. 128-38, 2004.

[Jena 2005] HP Labs Semantic Web Program "Jena: A Semantic Web Framework for Java," http://jena.sourceforge.net, 2005.

[Jess 2005] Sandia National Laboratories, "Jess: the rule engine for the Java platform," http://www.jessrules.com/, 2005.

[Kim 2004] Kim K.Y., Wang Y., Muogboh O.S., Nnaji B.O., "Design formalism for collaborative assembly design," *Computer-Aided Design*, vol. 36, pp. 449-71, 2004.

[Kim 2006] Kim K.Y., Manley D.G., Yang. H., "Ontology-based Assembly Design for Collaborative Product Development," Submitted to *CAD*, April 2006.

[Kitamura 2002] Kitamura Y., Sano T., Namba K., Mizoguchi R., "A functional concept ontology and its application to automatic identification of functional structures," *Advanced Engineering Informatics,* vol. 16, no. 2, pp. 145-63, 2002.

[Kitamura 2003] Kitamura Y., Mizoguchi R., "Ontology-based description of functional design knowledge and its use in a functional way server," Expert *Systems with Application*, vol. 24, no. 2, pp. 153-66, 2003.

[Kitamura 2004] Kitamura Y., Mizoguchi R., "Ontology-based systematization of functional knowledge," *Journal of Engineering Design*, vol. 15, no. 4, pp. 327-51, 2004.

[Kitamura 2004a] Kitamura Y., Kashiwase M., Masayoshi F., Mizoguchi R., "Deployment of an ontological framework of function design knowledge," *Advanced Engineering Informatics*, vol. 18, no. 2, pp. 115-27, 2004.

[Knublauch 2004] Knublauch H, Fergerson R.W., Noy N.F., Musen M.A., "The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications," *Third International Semantic Web Conference*, Hiroshima, Japan, 2004.

[Kurland 2003] Kurland R., "NX systems engineering powers the product lifecycle," TechniCom Inc Technical Article, 2003.

[Lee 1999] Lee K., *Principles of CAD/CAM/CAE Systems*. Addison Wesley Longman Inc, 1999.

[Lin 1996] Lin J., Fox M.S., Bilgic T., "A requirement ontology for engineering design," *Concurrent Engineering Research and Applications*, vol. 4, no. 3, pp. 279-91, 1996.

[Liu 1991] Liu H.C., Nnaji B.O., "Design with spatial relationships," *Journal of Manufacturing Systems*, vol. 10, no. 6, 1991.

[Lutters 1997] Lutters D., Streppel A.H., Kals H.J.J., "The role of information structures in design and engineering processes," *3rd Workshop on Product Structuring*, 1997.

[Mizoguchi 2003] Mizoguchi R., "Tutorial on ontological engineering part 1: introduction to ontological engineering" *New Generation Computing*, vol. 21, no. 4, pp. 365-84, 2003.

[Mizoguchi 2004] Mizoguchi R., "Tutorial on ontological engineering part 2: ontology development tools and languages," *New Generation Computing*, vol. 21, no. 4, pp. 61-96, 2004.

[Mizoguchi 2004a] Mizoguchi R., "Tutorial on ontological: engineering part 3 advanced course of ontological engineering," *New Generation Computing*, vol. 22, no. 2, pp. 193-220, 2004.

[Noy 2000] Noy N.F., Fergerson R.W., Musen M.A.. "The knowledge model of Protege-2000: Combining interoperability and flexibility." 2th *International Conference on Knowledge Engineering and Knowledge Management*, Juan-les-Pins, France, 2000.

[O'Connor 2005] O'Connor M.J., Knublauch H., Tu S.W., Musen M.A., "Writing Rules for the Semantic Web Using SWRL and Jess," *8th International Protégé Conference, Protégé with Rules Workshop*, Madrid, Spain, SMI-2005-1079, 2005.

[Rippey 2004] Rippey, W.G., *NISTIR 7107: A Welding Data Dictionary*, National Institute of Standards and Technology Technical Document, 2004.

[Schlenoff 1999] Schlenoff C., Ivester R., Libes D., Denno P., Szykman S., *NISTIR 6301: An analysis of existing ontological systems for applications in manufacturing and healthcare*, NIST Technical Report, 1999.

[Shah 1998] Shah J.J., Rogers M.T., "Functional requirements and conceptual design of the feature-based modeling system," *Computer-Aided Engineering Journal*, vol. 5, pp. 9-18, 1998.

[Singh 2005] Singh M.P., Huhns M.N., *Service-Oriented Computing: Semantics, Processes, Agents*, John Wiley & Sons Ltd, 2005.

[Uschold 1998] Uschold M., King M., Moralee S., Zorgios Y., "The enterprise ontology." *Knowledge Engineering Review*, vol. 13, Special Issue on Putting Ontologies to Use, 1998.

[W3C 1992] World Wide Web Consortium. "World wide web: summary," http://www.w3.org/Summary.html, 1992.

[W3C-OWL 2004] World Wide Web Consortium. "OWL: Web Ontology Language Overview," http://www.w3.org/TR/owl-features, 2004.

[W3C-SWRL 2004] World Wide Web Consortium. "Semantic Web Rule Language: Combining OWL and RuleML," http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/, 2004.

[Yao 1998] Yao Z., Bradley H.D., Maropoulos P.G., "An aggregate weld product model for the early design stages," *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, vol. 12, pp. 447-61, 1998.