Assembly Design Ontology for Service-Oriented Design Collaboration

Kyoung-Yun Kim[1], Hyungjeong Yang[2] and David G. Manley[3]

[1]Wayne State University, kykim@eng.wayne.edu
[2]Chonnam National University, hjyang@chonnam.ac.kr
[3]University of Pittsburgh, dgmst11@pitt.edu

ABSTRACT

This paper presents an Assembly Design (AsD) ontology that explicitly represents AsD constraints and infers any remaining implicit ones. By relating concepts through ontology technology rather than just defining data syntax, assembly and joining concepts can be captured in their entirety or extended as necessary. Ontologies allow assembly and joining constraints to be represented in a standard manner regardless of geometry file formats. Such representation will significantly improve service-oriented design collaboration. The developed AsD ontology is tested using a realistic mechanical assembly and it is shown how the ontology can be used to capture design rationale and analyze the design intents. In conclusion, the significance of ontology for realizing lean and selective assembly design information sharing is discussed.

Keywords: Assembly design ontology; Assembly relation model; Design collaboration.

1. INTRODUCTION

Discrete product manufacturers are under pressure from customers to move away from the traditional make-to-stock production model to a build-to-demand model. The impact of Internet technologies has accelerated the pace of product development and product aftermarket service. Industries now realize that the best way to reduce life cycle costs is to evolve a more effective product development paradigm using the Internet and web-based technologies [1]. Yet, there remains a gap between these current market demands and current product development paradigms. One possible approach to fill this gap is to seamlessly integrate product development processes into a collaborative environment. To realize such integration, it is necessary to realize an assembly design (AsD) model that includes self-descriptive, semantic information.

Designers are no longer merely exchanging geometric data, but knowledge about design and the product development process, including specifications, design rules, constraints, and rationale. As design becomes increasingly knowledge-intensive and collaborative, 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 [2][3].

To generate a robust assembly model, an understanding of assembly geometry and its physical effects is prerequisite. However, current solid modelers and simulation software are not advantageous drivers of robust assembly models since they provide incomplete product definitions and are not able to act according to the semantic content of the models. The reason for this is that, traditionally, the geometric model was in the center of product models. Of course, geometry is of importance during assembly design, but the morphological characteristics are consequences of the principle physical processes and the design intentions (e.g., joint intent) [4][5].

This paper presents an AsD ontology that plays a formal, explicit specification of a shared conceptualization of assembly design modeling. By utilizing ontology technology, clear relations among assembly components and form features are established and assembly knowledge is systemized in product, feature, manufacturing, and spatial relationship classes. In this paper, 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) [6], 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 by using OWL triples and SWRL rules. The AsD ontology also captures design rationale including joint intent and SR. This work can be used to query AsD information selectively as well as transparently. The AsD ontology is tested using a realistic mechanical assembly and shows how the ontology can be used to capture design rationale and analyze design intent. In conclusion, the significance of the ontology for realizing semantic assembly design information sharing, called *Semantic Assembly Design Modeling* (SADM) environment is discussed.

2. BACKGROUND

2.1 Ontologies

Ontologies are explicit formal specifications of the terms in the domain and relations among them [7]; a formal, explicit specification of a shared conceptualization. "Conceptualization" refers to an abstract model of some phenomenon in the world, which identifies the relevant concepts of that phenomenon. "Formal" refers to the fact that the ontology should be machine-readable [8]. Mizoguchi [9] presented the roles of ontologies as a common vocabulary, common 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. The broadest of ontologies, are the upper-level ontologies such as CYC, developed by Cycorp [10], that describe common sense-level knowledge. Narrower in scope than upper-level ontologies, enterprise-level ontologies attempt to formalize the practices and processes that occur within an organization. Examples of such ontologies include Enterprise Ontology [11] and TOVE [12]. The narrowest ontologies have been developed to represent conceptual and functional engineering information [13] as well as design features [4].

Of the ontologies reviewed, many of them have been developed and applied to broader business applications, which do not have the detail required at the engineering mechanical design level. Although some research has been applied to design, it has limitations on representing mechanical assemblies.

2.2 Service-oriented Design Collaboration

Integrated product engineering requires collaboration of various engineering tools from various disciplines such as mechanical design, electrical design, materials, manufacturing, quality, marketing, maintenance, and regulations. This integrated collaboration can be realized in a service-oriented collaboration environment through the Internet [6][14]. Instead of looking at various engineering tools from a traditional computation viewpoint, service-oriented design focuses on the engineering implication of design tools at an abstract level. This approach assures openness of collaborative engineering systems. From this perspective, the Internet is no longer a simple network of computers, rather it is a network of potential services.

To utilize service-oriented architecture (SOA) in design collaboration, services should be specified from the functional aspect of service providers. To make an existing tool available online or to build a new tool for such a system, associated services should be defined explicitly. The service transaction among service providers, service consumers, and the service manager within service-oriented design is illustrated in Fig. 1. Assembly design systems in this architecture should be able to produce assembly information that can be propagated and shared with various other service stakeholders and tools. Thus, this new assembly design paradigm that integrates assembly design and other product development activities requires assembly models that use self-descriptive modeling semantics.

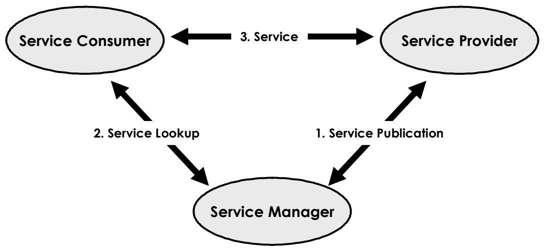

Fig. 1. Service triangular relationship.

2.3 Existing Assembly Design Formalism

Related to assembly design formalisms, Deneux [15] discussed the necessity of assembly features in the design of complex assemblies. Holland and Bronsvoort [16] proposed the concepts of a single-part feature model and an assembly feature model for assembly modeling. However, their assembly feature concepts, which considered only assemblies with mechanical fastening, cannot be employed to assembly modeling requiring various other joining processes. Whitney, et al. [17] proposed a formalism for assembly design and focused on only fully constrained assemblies and subassemblies. Even though they presented a general methodology for assembly design, many SRs in actual mechanical assemblies, e.g., between two cylindrical surfaces and between spherical surface and other surfaces were not addressed. Also, the effects on SR from joining processes were not discussed.

Kim et al. [6] has developed AsD formalism based on SRs [18][20] and features. SRs are described briefly in the next sub-section. The formalism represents assembly and joining relations and generates a meta-model called ARM (assembly relation model) to explicitly represent the relations in XML format. However, their formalism still has implicit constraints and relationships, which cannot be interpreted by a computer in the Semantic Web and service-oriented design environments. These implicit relationships and constraints are necessary for various downstream design activities, including transparent assembly and joining analysis in a collaborative assembly modeling environment. Thus, this work will enhance the AsD formalism using ontology technology. The AsD ontology developed will explicitly represent AsD design constraints and the computer will be able to infer any remaining implicit ones. By relating concepts through ontology technology rather than just defining data syntax, assembly and joining concepts can be captured in their entirety or extended as necessary. Furthermore, the higher semantic richness of ontologies allows computers to infer additional assembly/joining knowledge and make that knowledge available to decision makers. In this paper, the three types of assembly relationships (i.e., belongTo, inter-featureAssociation, and assembly/joining relations), SRs, and the joining relations are represented in an ontology and are used as a core model in semantic assembly design modeling.

2.3.1 Spatial Relationship (SR)

SRs were first proposed by Ambler and Popplestone [18] in 1975 to describe the relative positions of parts in their final state by specifying feature relationships among them. SRs include *against*, *coplanar*, *fits*, *parax*, *lin*, *rot*, and *fix*. In the work of Ambler and Popplestone, SRs are concentrated on the configuration of a part. Liu and Nnaji [20] focused on the mechanical assembly specifications as well as the configuration of a product, so that SRs can be applied to general assemblies and are capable of accepting the design specifications. They defined design with SRs not only for inferring the assembly positions, but also to capture designer's intentions. Each SR constrains the degrees of freedom (DOF) of motion between the mating entities. For example, two faces are said to be against if the two faces are touching at some point and normal vectors of those faces are in opposition where they touch. Any combination of two features can possess this property.

The types of DOF are classified as follows; *lin_n*: linear translation along n axis, where, n contains a fixed point and a vector; *rot_n*: rotation about n axis, where, n contains a fixed point and a vector; *cir_n*: translating along a circle with n axis, where, the fixed point of n is the center of the circle and vector of n is perpendicular to the circle; *plane_n*, *cyl_n*, and *sph*: translating along a planar, cylindrical, spherical surface. The DOF of a part are expressed as {DOF of moving within the coordinate system of the relative moving part :: DOF of moving within its own coordinate system}, with respect to the other mating parts of the assembly [20]. For a plane_z_a DOF, a body may move on a planar surface along two lin. When a new plane_z_b is introduced, the remaining DOF are derived by intersecting these two planes. The intersection of surface DOF is available only when a plane is involved; circle_n is the result of intersecting plane_z_a with cyl_n (or sph_n) together. In the intersection of two rotational DOF, say rot_z_a and rot_z_b, if they share the same rotational axis then rot_z_a or rot_z_b remains; if not they cancel each other. Liu and Nnaji [20] presented some general reduction rules for DOF.

3. ASSEMBLY DESIGN ONTOLOGY

3.1 Notes on Implementation

For this research, the Semantic Web Rule Language (SWRL) is used. This language is based on a combination of the OWL DL and OWL Lite sublanguages of the Web Ontology Language (OWL) with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language. SWRL is intended to be the rule language of the Semantic Web. It includes a high-level abstract syntax for Horn-like rules in both the OWL DL and OWL Lite sublanguages of OWL. The World Wide Web Consortium (W3C) has acknowledged receipt of a Member Submission from the National Research Council of Canada, Network Inference, and Stanford University. The submission has been made in association with the Joint US/EU ad hoc Agent Markup Language Committee. The SWRL submission package contains three components in addition to the principal prose document: (1) an RDF Schema partially describing the RDF Concrete Syntax of SWRL; (2) an OWL ontology partially describing the RDF Concrete Syntax of SWRL; (3) an XML Schema for the SWRL XML Concrete Syntax [21]. OWL, which is a recommendation for Semantic Web technology by the World Wide Web Consortium (W3C) [22], was investigated; however, the constraint representation capability of OWL alone was not adequate to implement an AsD ontology that can be fully utilized in the SADM environment. In this research, SWRL is used to define AsD terms and their relationships. Also, SWRL rules have been implemented so that the rules can be reasoned to accommodate potential semantic queries/information request in the SADM environment. To generate the AsD ontology and rules, the SWRL editor in Protégé was used for this work.

3.2 Components of the Assembly Design Ontology

The definitions, possible terms, and concept representation of assembly were investigated thoroughly. Based on this investigation, the terms of the AsD ontology were carefully defined. Fig. 2 illustrates the hierarchy of AsD ontology classes. Within the hierarchy, a class is a subclass of another class.

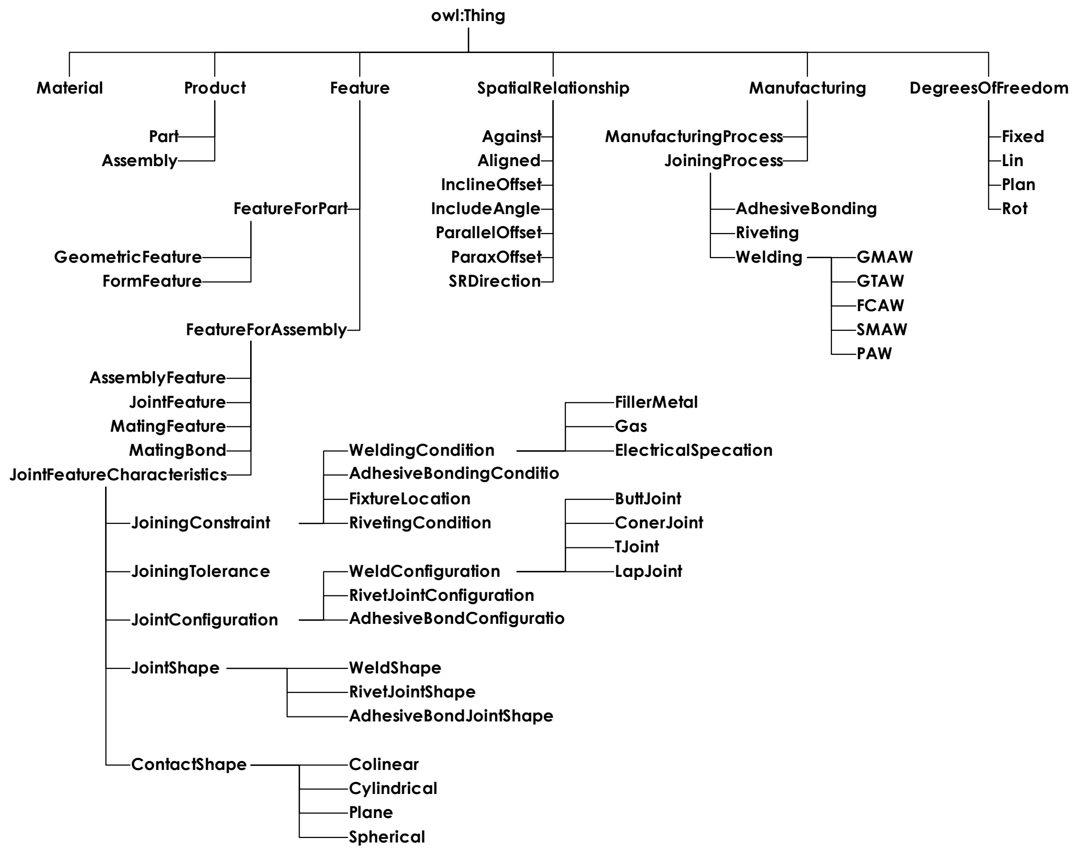

Fig. 2. AsD ontology class hierarchy.

An OWL 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 (e.g., material). Properties can be used to state relationships between individuals or between individuals and data values. Classes such as JointFeatureCharacteristics are introduced in this work to represent relations in assembly models. With these supplementary classes, properties, and SWRL rules illustrated in the next section, the assembly design relationships and constraints are developed. This paper focuses especially on the realization of semantic query that can provide appropriate assembly design information by SWRL reasoning for service-oriented design.

3.3 AsD Constraint Representations in SWRL

3.3.1 Relationships in Assembly Design

The first core constraints in the AsD ontology are assembly relationships *form feature to form feature* and *form feature to part*. The assembly constraints are represented explicitly using OWL triples and SWRL rules. In this section, three assembly relations are described to illustrate how the assembly relationships can be represented in the AsD ontology.

The basic assumptions of these definitions are 1) part (P) is a member of assembly (Θ) (i.e., $P \in \Theta$) and 2) form feature (FF) is a member of part (P) (i.e., $FF \in P$)

1) *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). The inferred constraints can be transformed into two asserted conditions. For C1-1, an asserted condition, "∃ FormFeature belongTo SubPart" is used and an asserted condition, "belongTo = 1" is used to represent C1-2. This OWL cardinality condition means the property belongTo has exactly one value.

Implied Constraints

- C1-1: Every form feature (FormFeature) must belong to a part (SubPart)
- C1-2: A form feature (FormFeature) must not belong to two parts (SubPart)

2) *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 of assembly (A_j). 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 (the block is a parent feature of the hole in the feature decomposition hierarchy), their relational constraint (RC_{pq}) is 0. If FF_{jp} has some form feature (FF_{jq}), 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).

Implied constraints

- C2-1: The associated form features must not be identical (non-equivalent)
- C2-2: The associated form features must belong to same part
- C2-3: The relational constraint must be represented (included)

SWRL rules

- FormFeature(?x) ∧ FormFeature(?y) ∧ Part(?z) ∧ differentFrom(?x, ?y) ∧ belongTo(?x, ?z) ∧ belongTo(?y, ?z) ∧ RelationalConstraint(?a) ∧ sameAs(?a, 0) ∧ inter-featureAssociation(?x, ?y)
- FormFeature(?x) ∧ FormFeature(?y) ∧ Part(?z) ∧ differentFrom(?x, ?y) ∧ belongTo(?x, ?z) ∧ belongTo(?y, ?z) ∧ RelationalConstraint(?a) ∧ sameAs(?a, 1) ∧ inter-featureAssociation(?y, ?x)
- FormFeature(?x) ∧ FormFeature(?y) ∧ Part(?z) ∧ differentFrom(?x, ?y) ∧ belongTo(?x, ?z) ∧ belongTo(?y, ?z) ∧ RelationalConstraint(?a) ∧ sameAs(?a, 2) ∧ inter-featureAssociation(?x, ?y)
- FormFeature(?x) ∧ FormFeature(?y) ∧ Part(?z) ∧ differentFrom(?x, ?y) ∧ belongTo(?x, ?z) ∧ belongTo(?y, ?z) ∧ RelationalConstraint(?a) ∧ sameAs(?a, 2) ∧ straddleRelation (?x, ?y)

To represent the implied constraints, SWRL rules are used as shown above. Based on the relational constraint (RelationalConstraint), the inter-featureAssociation is defined. The sequence of variables in the inter-featureAssociation represents the feature relations. When the two form features do not belong to each other (i.e., when the relational constraint is 2), a new property (i.e., straddleRelation) is needed. The name of property is based a mereo-topological term.

3) *Assembly/joining Relations*: The assembly/joining relations represent the relations between form features that belong to different parts. They imply two constraints (C3-1 and C3-2) and the implied constraints are represented by SWRL rules.

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

- FormFeature(?x) ∧ FormFeature(?y) ∧ Part(?z) ∧ Part(?a) ∧ belongTo(?x, ?z) ∧ belongTo(?y, ?a) ∧ differentFrom(?z, ?a) ∧ isJointPair(?x, ?y) → assemblyJoiningRelationship(?x, ?y)

3.3.2 Constraints in Spatial Relationships and Joining

The implied constraints associated to SRs, DOF, and joining operations have been investigated in this paper. The examples of implied constraints and associated SWRL rules are listed in Tab. 1. These rules are utilized in this paper to understand design intents and joining implications.

4. ASSEMBLY DESIGN INTENT ANALYSIS

In assembly design, SRs can be assigned to achieve intended DOF. 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 distributed collaborative design processes. This paper shows how the designed DOF are persistently maintained by the developed AsD ontology and that the information can be used to compare the original assembly design intent with DOF implied by specific joining methods.

Domain	Example of implied constraints	Example of SWRL rules
SR and DOF	■ Two parts, which have against SR between planar surfaces along the z-direction, have DOF of {Plan_Z::Rot_Z} (e.g., plane (plan) DOF along the z-direction of the coordinate of the reference part and rotational (rot) DOF within its own coordinate system (z-direction)).	MatingFeature(?x) ∧ FormFeature(?y) ∧ Part(?z) ∧ belongTo(?y, ?z) ∧ hasMatingComponent(?x, ?y) ∧ Against(?a) ∧ hasSpatialRelationship(?x, ?a) ∧ Plane_Z(?b) ∧ hasContactShape(?x, ?b) ∧ Z-Direction(?d) ∧ ReferenceDirection(?x, ?d) ∧ Plan_Z(?c) → hasDOF(?z, ?c)
		MatingFeature(?x) ∧ FormFeature(?y) ∧ Part(?z) ∧ belongTo(?y, ?z) ∧ hasMatingComponent(?x, ?y) ∧ Against(?a) ∧ hasSpatialRelationship(?x, ?a) ∧ Plane_Z(?b) ∧ hasContactShape(?x, ?b) ∧ Z-Direction(?d) ∧ ReferenceDirection(?x, ?d) ∧ Rot_Z(?c) → hasOwnDOF(?z, ?c)
	■ Two parts, which have aligned SR between collinear lines along the x-direction, have DOF of {Lin_X::Rot_X} (e.g., linear (lin) DOF along the x-direction the coordinate of the reference part and rotational (rot) DOF within its own coordinate system (x-direction))	MatingFeature(?x) ∧ FormFeature(?y) ∧ Part(?z) ∧ belongTo(?y, ?z) ∧ hasMatingComponent(?x, ?y) ∧ Aligned(?a) ∧ hasSpatialRelationship(?x, ?a) ∧ Collinear(?b) ∧ hasContactShape(?x, ?b) ∧ X-Direction(?d) ∧ ReferenceDirection(?x, ?d) ∧ Lin_X(?c) → hasDOF(?z, ?c)
		MatingFeature(?x) ∧ FormFeature(?y) ∧ Part(?z) ∧ belongTo(?y, ?z) ∧ hasMatingComponent(?x, ?y) ∧ Aligned(?a) ∧ hasSpatialRelationship(?x, ?a) ∧ Colinear(?b) ∧ hasContactShape(?x, ?b) ∧ X-Direction(?d) ∧ ReferenceDirection(?x, ?d) ∧ Rot_X(?c) → hasOwnDOF(?z, ?c)
Designed DOF	■ A part, which has a rotational DOF (rot) and two linear DOF (lin), has fixed designed DOF	Part(?x) ∧ Rot_Z(?y) ∧ Lin_X(?z) ∧ Lin_Y(?a) ∧ hasDOF(?x, ?y) ∧ hasDOF(?x, ?z) ∧ hasDOF(?x, ?a) ∧ Fixed(?b) → hasDesignedDOF(?x, ?b)
DOF inferred by a joining method	■ A part joined by welding has fixed DOF, which is inferred by welding	Welding(?x) ∧ JointFeature(?y) ∧ FormFeature(?z) ∧ Fixed(?a) ∧ Part(?b) ∧ isJoiningMethod(?y, ?x) ∧ hasJoiningComponents(?y, ?z) ∧ belongTo(?z, ?b) → hasJoiningInferredDOF(?b, ?a)
	■ A part joined by more than one rivet has fixed DOF, which is inferred by the rivets	Riveting(?x) ∧ JointFeature(?y) ∧ FormFeature(?z) ∧ Fixed(?a) ∧ Part(?b) ∧ isJoiningMethod(?y, ?x) ∧ hasJoiningComponents(?y, ?z) ∧ belongTo(?z, ?b) ∧ RivetingCondition(?c) ∧ hasJoiningConstraint(?y, ?c) ∧ NumberOfRivet(?c, ?d) ∧ swrlb:greaterThanOrEqual(?d, 2) → hasJoiningInferredDOF(?b, ?a)

Tab. 1. Constraint representation for SRs, DOF, and joining operations.

SRs are specified /imposed during the assembly design process. As described in the previous sections, each SR can be interpreted as a constraint imposed on the DOF between relative mating or interacting features. Given a set of SRs, the resultant DOF can be inferred. In other words, any allowable motion of parts must follow a path along the directions specified by the DOF in order to maintain their SRs. Fig. 3 illustrates how SRs implied by joint design can be used for a designer's intent analysis. As shown in the figure, each joining method infers specific SRs and the corresponding DOF are implied by these SRs. The designer's original intent imposed on assembly design can be analyzed by comparing the implied DOF and the designed DOF. For example, a designer wants to permanently join two plates and he/she assigns spatial relationships to fix those plates. If the 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 and used to check whether this welding operation will satisfy the designer's intent on 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 loose all DOF and become fixed. If the designed DOF is fixed by assigning series of SR, the specified joining method (welding) fully satisfies the designed DOF. Some joining methods may either under-constrain or over-constrain the DOF on an assembly.

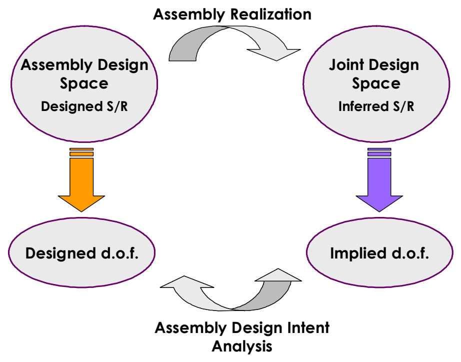

Fig. 3. Assembly design intent analysis.

5. DEMONSTRATION OF ASSEMBLY DESIGN ONTOLOGY

5.1 Case Study

As a case study, a connector assembly (Fig. 4) is used to demonstrate the knowledge capturing capability of the AsD ontology employed in an assembly design browser. In this example, plate_a and plate_b are joined by two button rivets. Top_surface of plate_a and bottom_surface of plate_b have against relationships. The rivets are aligned along centerline of holes at the location that designer specified. Plate_a and plate_c are joined with Gas Metal Arc Welding (GMAW) and their mating feature entities are top_surface of plate_a and bottom_surface of plate_c. Similarly, plate_b and plate_d are joined by using GMAW. Tab. 2 shows notations used for this case study. Datum planes in JC_5 and JC_8 are reference planes that represent the designated tolerance limits of welding deformation. The datum planes are offset from the structure by the tolerance limits allowed. Fig. 5 shows feature hierarchy.

Notation	Description
e_i^{jk}	ith 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)

Tab. 2. Notation for connector assembly.

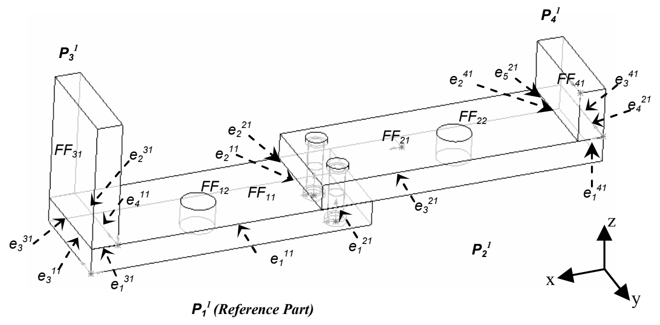

Fig. 4. Connector assembly with two welded joints and one pin joint.

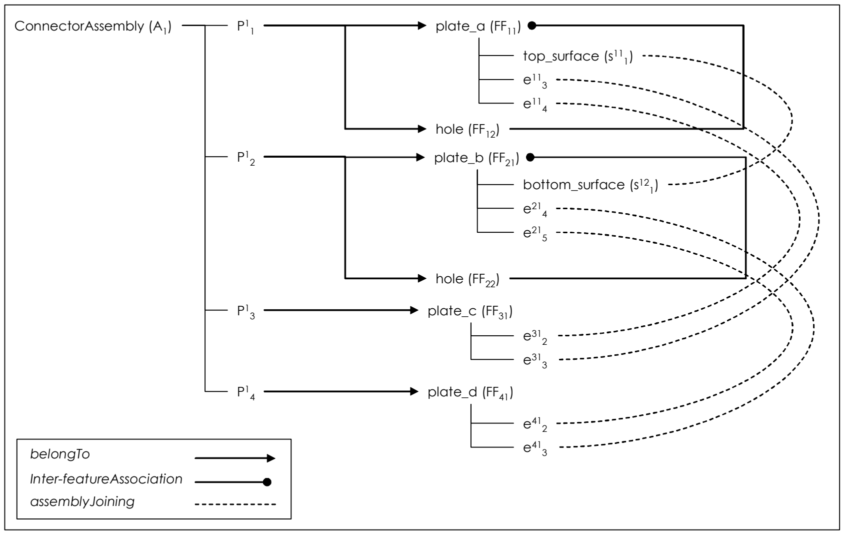

Fig. 5. Feature hierarchy and assembly relationships in the connector assembly.

	Anticipated result	Result from reasoning
Designed DOF	■ Part_12 has Fixed DOF ■ Part_13 has Fixed DOF ■ Part_14 has Fixed DOF	■ fact ns_0:Fact24442 is ns_2:hasDesignedDOF(ns_2:Part_12,ns_2:Fixed_1); ■ fact ns_0:Fact24440 is ns_2:hasDesignedDOF(ns_2:Part_13,ns_2:Fixed_1); ■ fact ns_0:Fact24441 is ns_2:hasDesignedDOF(ns_2:Part_14,ns_2:Fixed_1);
DOF inferred by a welding method	■ DOF of Part_13 has been constrained as Fixed by welding (based on the reference part Part_11) ■ DOF of Part_14 has been constrained as Fixed by welding (based on the reference part Part_12)	■ fact ns_0:Fact30124 is ns_2:hasJoiningInferredDOF(ns_2:Part_13,ns_2:Fixed_1); ■ fact ns_0:Fact30123 is ns_2:hasJoiningInferredDOF(ns_2:Part_14,ns_2:Fixed_1);
DOF inferred by a riveting method	■ DOF of Part_11 has been constrained as Fixed by two rivets ■ DOF of Part_12 has been constrained as Fixed by two rivets (based on the reference part Part_11)	■ fact ns_0:Fact37293 is ns_2:hasJoiningInferredDOF(ns_2:Part_11,ns_2:Fixed_1); ■ fact ns_0:Fact37294 is ns_2:hasJoiningInferredDOF(ns_2:Part_12,ns_2:Fixed_1);

Tab. 3. Reasoning result of the AsD ontology for the connector assembly.

5.2 Assembly Design Ontology, SWRL Reasoning, and Design Intent Analysis

In this paper, Bossam is used as a SWRL rule inference engine. Bossam is an extended forward-chaining rule engine developed by Korean Electronics and Telecommunications Research Institute (ETRI). Bossam supports OWL inferencing and is accompanied with a set of OWL Lite/DL inference rules. It also supports SWRL/OWLX, SWRL/RDF and RuleML. Tab. 3 shows examples of the anticipated and actual results of the AsD ontology reasoning, particularly for assembly intent analysis. The AsD ontology is reasoned with valid given conditions for a specific query. If a query doesn't satisfy the conditions, the query is not reasoned. If the query has incorrect syntax, the SWRL editor checks the validity. When the query has the correct syntax, but does not satisfy the conditions, the query is just passed and the reasoned result is null.

When a designer wants to permanently join two plates (P_1^1 and P_1^3) and he/she assigns SRs to fix those plates. As in the connector assembly, if the 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 SWRL reasoning and used to check whether this welding operation will satisfy the designer's intent on the assembly. The welding operation causes fixed DOF. In the connector assembly, the specified joining method (welding) fully satisfies the designed DOF (Tab. 3)

In other cases, some joining methods may either under-constrain or over-constrain the DOF on an assembly. As in the connector assembly, the two plates (P_1^1 and P_1^2) are intended to be joined and their DOF are fixed by assigning a series of SRs. The intended DOF can be reasoned as shown in Tab. 3. If a designer wants to join the two plates by applying one cylindrical rivet at a position, the intended DOF (fixed) is under-constrained. In a riveting operation, the end of the rivet shank is deformed after upsetting. However, after upsetting, the assembly can still have rotational DOF, if there is enough tangential (rotational) force applied to the two plates. When two rivets are used to join the assembly components, the DOF of components are fully constrained. Note that more than two rivets can increase structural rigidity, even though DOF of the assembly are over-constrained and joining cost and time are increased. The proper number of rivets should be determined by assembly operation analysis.

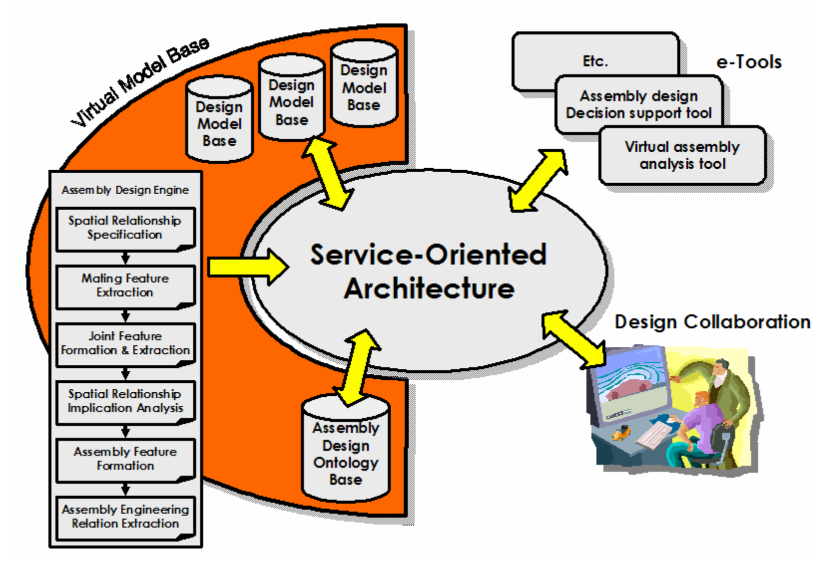
Virtual Model Base
Design Model Base
Design Model Base
Design Model Base
Etc.
e-Tools
Assembly design Decision support tool
Virtual assembly analysis tool
Assembly Design Engine
Spatial Relationship Specification
Mating Feature Extraction
Joint Feature Formation & Extraction
Spatial Relationship Implication Analysis
Assembly Feature Formation
Assembly Engineering Relation Extraction
Service-Oriented Architecture
Design Collaboration
Assembly Design Ontology Base

Fig. 6. Semantic assembly design modeling framework.

6. CONCLUSION AND FUTURE WORKS

This paper presented an AsD ontology that plays as formal, explicit specification of a shared conceptualization of assembly design modeling. This paper enhanced the previously developed AsD formalism by using ontology technology. The AsD ontology developed explicitly represents AsD design constraints and the computer can infer remaining implicit constraints (e.g., assembly relationships, SRs, and joining relations). By relating concepts through ontology technology rather than just defining data syntax, assembly and joining concepts can be captured in their entirety or extended as necessary. Furthermore, the higher semantic richness of ontologies allows computers to infer additional assembly/joining knowledge and make that knowledge available to decision makers. By using ontology technology, assembly and joining constraints can be represented in a standard manner regardless of geometry file formats. Such representation significantly improves service-oriented design collaboration. 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.

To understand what kind of AsD information should be represented in ontology and shared in distributed design collaboration, a survey was conducted with product design and development engineers at automotive industries including Ford Motor Company and frequently asked questions were obtained and classified. Most notably, the survey clearly indicated that assembly relation and assembly method/operation are core information that should be shared in distributed design collaboration.

In recognizing the fact that product development requires tremendous amounts of information and corresponding decisions, it is obvious that the reasoning behind every decision can hardly be transferred together with the information. Consequently, the need for feedback and communication among product development stakeholders increases, which may lead to extremely complex and uncontrollable flows of information in product development collaboration. However, providing that product information generated by the different collaborators is attached to an overall and widely accessible model, this situation may change considerably. Instead of "pushing" information from one collaborator to another, participants can "pull" the information they require and are given access to. As shown in this paper, by utilizing ontology technology, clear relations among assembly components and form features are established and assembly knowledge can be systemized in product, feature, manufacturing, joining, and analysis classes. This semantically valid information provides a great foundation to realize a lean and selective assembly design information sharing environment. The authors are currently developing a new information sharing paradigm, called Semantic Assembly Design Modeling (SADM). Fig. 6 illustrates a framework for the SADM. Typically, product development collaboration requires a secured network and infrastructure. SADM has to be implemented in such a secure data network, which is developed through intensive research. In this framework, design collaborators and e-tools (e.g., virtual assembly tool and assembly design decision support tool) at remote locations can request assembly information from the AsD ontology in service-oriented design collaboration. Since the AsD ontology systemizes assembly knowledge, the e-tools and collaborators can retrieve assembly information selectively and transparently via semantic query.

7. REFERENCES

[1] 2005 Engineous International Symposium & Workshop, Novi, MI. USA, October 10-12, 2005

[2] Lutters D, Streppel AH, Kals HJJ. The role of information structures in design and engineering processes. 3rd Workshop on Product Structuring 1997.

[3] Szykman, S., Sriram, R. D. and Regli, W. C., The role of knowledge in next-generation product development systems, *J Computing and Information Sc in Engr*, Vol. 1, 2001, pp 3-11.

[4] Horváth, I., Pulles, J. P. W., Bremer, A. P. and 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.

[5] Kurland, R., NX systems engineering powers the product lifecycle, *TechniCom Inc technical article*, 2003.

[6] Kim, K. Y., Wang, Y., Muogboh, O. S. and Nnaji, B. O., Design Formalism For Collaborative Assembly Design Environment, *Computer-Aided Design*, Vol. 36 No. 9, 2004, pp 849-71.

[7] Gruber, T. R., A Translation Approach to Portable Ontology Specification, *Knowledge Acquisition*, Vol. 5, No. 2, 1993, pp 199-220.

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

[9] Mizoguchi, R. Tutorial on Ontological Engineering Part 1: Introduction to Ontological Engineering, *New Generation Computing*, Ohm-Sha & Springer, Vol. 21, No. 4, 2003, pp 365-84.

[10] Cycorp, Inc., http://www.cyc.com/cyc

[11] Uschold, M., King, M., Moralee, S., and Zorgios, Y., The Enterprise Ontology, *The Knowledge Engineering Review*, Vol. 13, Special Issue on Putting Ontologies to Use, 1998.

[12] Fox, M. S., and Gruninger, M., Enterprise Modelling, *AI Magazine*, AAAI Press, Fall 1998, pp 109-21.

[13] Kitamura, Y., Kashiwase, M., Masayoshi, F. and Mizoguchi, R., Deployment of an Ontological Framework of Function Design Knowledge, *Unpublished manuscript for Advanced Engineering Informatics*, 2004.

[14] Nnaji, B. O., Wang, Y. and Kim, K. Y., Service-oriented architecture for integrated e-design and realization of engineered products, *International Forum on Design for Manufacture and Assembly*, Providence, RI. USA, June 22-23, 2004.

[15] Deneux, D., Introduction to assembly features: an illustrated synthesis methodology, *J Intelligent Manufacturing*, Vol. 10, 1999, pp 29-39.

[16] van Holland, W. and Bronsvoort, W. F., Assembly features in modeling and planning, *Robotics and Computer Integrated Manufacturing*, Vol. 16, 2000, pp 277-94.

[17] Whitney, D. E., Mantripragada, R., Adams, J. E. and Rhee, S. J., Toward a theory for design of kinematically constrained mechanical assemblies, Int J Robotics Research, Vol. 18, No. 12, 1999, pp 1235-48.

[18] Ambler, A. P. and Popplestone, R. J., Inferring the positions of bodies from specified spatial relationships, *Artificial Intelligence*, Vol. 6, No. 2, 1975.

[19] Liu, T. L., A coordinated constraint-based modeling and design advisory system for mechanical components and assemblies, Ph. D. dissertation, University of Massachusetts Amherst, 1997.

[20] Liu, H. C. and Nnaji, B. O., Design with spatial relationships, *Jr. of Manufacturing Systems*, Vol. 10, 1991.

[21] http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/

[22] World Wide Web Consortium, OWL web ontology language guide, http://www.w3c.org/TR/owl-guide.