NRC Publications Archive Archives des publications du CNRC

Ontology-based interoperation model of collaborative product development

Sun, Hongbo; Fan, Wenhui; Shen, Weiming; Xiao, Tianyuan

This publication could be one of several versions: author's original, accepted manuscript or the publisher's version. / La version de cette publication peut être l'une des suivantes : la version prépublication de l'auteur, la version acceptée du manuscrit ou la version de l'éditeur.

For the publisher's version, please access the DOI link below. / Pour consulter la version de l'éditeur, utilisez le lien DOI ci-dessous.

Publisher's version / Version de l'éditeur:

<https://doi.org/10.1016/j.jnca.2011.02.012>

Journal of Network and Computer Applications, 35, 1, pp. 132-144, 2012-01-01

NRC Publications Record / Notice d'Archives des publications de CNRC:

<https://nrc-publications.canada.ca/eng/view/object/?id=99986c53-40c3-4b1c-bc00-3a9555f5ec56>

<https://publications-cnrc.canada.ca/fra/voir/objet/?id=99986c53-40c3-4b1c-bc00-3a9555f5ec56>

Access and use of this website and the material on it are subject to the Terms and Conditions set forth at

<https://nrc-publications.canada.ca/eng/copyright>

READ THESE TERMS AND CONDITIONS CAREFULLY BEFORE USING THIS WEBSITE.

L'accès à ce site Web et l'utilisation de son contenu sont assujettis aux conditions présentées dans le site

<https://publications-cnrc.canada.ca/fra/droits>

LISEZ CES CONDITIONS ATTENTIVEMENT AVANT D'UTILISER CE SITE WEB.

Questions? Contact the NRC Publications Archive team at

PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca. If you wish to email the authors directly, please see the first page of the publication for their contact information.

Vous avez des questions? Nous pouvons vous aider. Pour communiquer directement avec un auteur, consultez la première page de la revue dans laquelle son article a été publié afin de trouver ses coordonnées. Si vous n'arrivez pas à les repérer, communiquez avec nous à PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca.

Ontology-based Interoperation Model of Collaborative Product Development

Hongbo Sun^{1,2}, Wenhui Fan¹, Weiming Shen², Tianyuan Xiao¹

¹National CIMS Engineering Research Centre, Tsinghua University,
BeiJing, China;

² Centre for Computer-assisted Construction Technologies, National Research Council,
London, Ontario, Canada

Sunhongbo02@tsinghua.org.cn

Abstract

The reuse of existing systems is an important objective of HLA (High Level Architecture)-based collaborative product development systems. However, in order to reuse an existing system, its interoperation interface has to be modified to comply with the objective and interaction representations defined in a corresponding FOM (Federation Object Model). Such modifications imply added time and effort, which diminishes the efficiency of system reuse in collaborative product development. This paper presents a heavy-weighted ontology-based construction method for interoperation models to support the reuse of subsystems in various collaborative contexts. In this method ontologies are used to specify the semantics of object classes and interaction classes in subsystems in a formal and computer readable fashion. In doing so, a FCA (Formal Concept Analysis) -like construction method is introduced to establish the original interoperation ontology from scratch. An automatic transforming method from a SOM (Simulation Object Model) into interoperation ontology is also described to make it easy for existing HLA-based systems to adopt this approach. Then a consistency verification method is introduced to guarantee the consistency of the interoperation ontologies. A case study is used to demonstrate the feasibility of the proposed method. As a human- friendly modeling method, compared with existing interoperation modeling methods, the proposed method is more flexible, efficient and reliable.

1. Introduction

In increasingly saturated markets, innovation and continuous product development are essential for enterprises to survive. Adopting collaborative product development makes full use of several independent development systems, and enhances their abilities at the same time [1]. Collaborative product development systems involve processes like CAD modeling, simulation and optimization, require data like CAD digital models, CAE analysis and optimization results [2,3]. Collaborative product development is a typical distributed collaboration of heterogeneous systems. When several independent systems need to be integrated with others, mutual understanding is always a real challenge [4]. In a typical High Level Architecture (HLA)-based collaborative product development system, Federation Object Models (FOM) perform as a mutual understanding media and Simulation Object Models (SOM) are used as a capacity describing entity of subsystems [5]. But they are usually established manually. Their consistency and completeness cannot be guaranteed and they are not easy to understand, even for professional experts.

Ontology is the semantic basis of communication among domain entities. It can be used in automatic reasoning, knowledge representation and knowledge reuse [6]. Ontology-based approaches

have been used to resolve the problem of heterogeneous data and information integration [7]. They can be used to share knowledge among systems or users, and to make a knowledge-based inference or reasoning. As Rathnam [8] mentioned, the HLA FED (FEDeration) files provide insight into the development of a Meta model for capturing simulation concepts in an ontology. This allows facilities to adopt ontology in HLA-based collaborative product development. At the same time, the automatic transformation mechanism in ontology that contains axioms will greatly reduce the changes of federate simulation codes. That is to say, after ontology is used in these developments, there is a promising potential to save the work load. Ontologies have become so popular that they provide a shared and common understanding of a domain that can be communicated between people and application systems [9]. Recently, there have been an increasing number of research projects applying ontological techniques in the context of product development [10, 11, 12]. But none of these projects directly addresses the issues of utilizing ontology techniques to describe HLA-based subsystem capabilities that are compatible with the existing HLA-based collaborative product development systems.

This paper is about developing a novel method to construct interoperation models of HLA-based collaboration product development in a human-friendly, efficient and consistent way and to support multi-disciplinary collaborations. To do so, a FCA (Formal Concept Analysis) -based interoperation ontology modeling method is specially tailored for ontology construction of HLA-based collaborative product development from scratch. When it comes to the systems which already have the capability of describing SOMs, a finite automata machine for transferring them into collaborative ontology is established in this paper to make it easier for existing HLA-based platforms to adopt this approach. Then, a consistency verification method is introduced to guarantee the consistency of the interoperation models constructed for collaborative product development.

This paper is organized as follows. Section 2 reviews the related work. In section 3 the theoretical foundations are shown in a formalized format. Section 4 presents the detailed modeling methods. Section 5 takes a typical collaborative product development unit as an example to demonstrate the feasibility of the proposed methods. Discussions and a future plan are in section 6.

2 Related Work

Nowadays more and more simulation functions are added into collaborative product development [13, 14]. The design of a product can be deemed as a multi-step process in which a set of design goals and requirements are transformed into a functional system. Simulation functions help these systems to fulfill their design goals and to add to their potential values. The key merits of simulation in the context of product development can be listed as follows [15, 16, 17]:

- To explore and focus the solution space corresponding to a given design or optimization problem at a reasonable cost.
- To reduce uncertainty about the system and reduce the changes of design failure.
- To predict and analyze system behavior in a well established environment.
- To test and guide the development of different parts or subsystems so that they can run harmoniously under the condition of meeting the design goals.

When simulation is added to a collaborative product development environment, there are always several subsystems in the same environment with independent design goals. And these subsystems may

follow different design or running rules according to the given professional field. Under this circumstance, distributed and federated systems are often very useful.

HLA is the well known architecture for distributed simulation [18], but using HLA in collaborative product development is more than just applying HLA as the overall architecture of these systems. In fact, there is a lot of research work required to apply HLA into all of the collaborative product development systems [13, 19, 20]. Unfortunately, besides simulation there are still many other functions in these systems and the design goals of these systems are not always limited to the area of simulation, so there are some new features.

- The independence of every subsystem should be well maintained and when integrated they can collaborate harmoniously and efficiently.
- Every subsystem has its own representation method. When they need to perform the collaboration, the difference of representation semantics cannot be eliminated by formalization means. Although HLA OMT well defines the format of interaction classes and object classes for simulation, it cannot solve the semantic problem raised by different knowledge representation methods.
- Simulation may not be the only design goal of system integration, and other design goals include design optimization and visualization. Simulation can be one step in fulfilling these goals.
- During the time the whole system is running, simulation may take place several times. That means reuse ability is required during a system run, and a reuse method that involves recompiling is not applicable here.
- Time advance strategy is not the kernel of the whole system integration, and at the same time synchronization and mutual understanding are of greater importance.
- The collaboration sponsor is always changing and there can be more than one collaboration task at the same time. So it is not recommended to stop one subsystem to wait for its recompiling. The preparation of new collaborations must be more efficient and automatic.

From that mentioned above, it can be seen that though one of the most important goals of HLA is the reuse of existing simulation systems and resources it is not enough for HLA-based collaborative product development. Furthermore, according to the architecture and aim of the HLA framework, the simulation codes need to change and recompile to fulfill the goal of upcoming collaborative simulation jobs. However, when applying HLA in collaborative product development these changes add cost and time, and they also greatly affect the efficiency of such kinds of collaborations.

Ontology technology has been successfully used for mutual understanding among several independent systems. The ontology-based approach may play an important role to address these issues. The target of this research is to explore a novel SOM construction method which employs an efficient way to describe the capability of each subsystem and takes full advantage of ontology technologies.

Ontology construction research usually focuses on ontology learning [21]. And the main semi-automatic construction methods used now are category theory and the FCA method. Other than the well known set theory, some researchers start their work from category theory. Category theory is based on graphics and the key concepts are easier to reuse than those in sets. Unfortunately, category

theory is not easy to understand. That is to say, it is not easily accepted by end users. And the simplest *Set* category (formed by all sets and full functions among them) cannot describe the relations of concepts in heavy weighted ontology. What the collaborative product development interoperation ontology needs is *Med* category (formed by all exchangeable groupoids and homomorphic mappings among them). However, even for the experts with a very good mathematics background, seeking the commutation graphics and cones - is too complex. . FCA is a method used for ontology discovery, sorting and display [22]. All the concepts involved with their generalization and specializations form a conceptual lattice. This lattice is the analysis and structurization of domain knowledge. The essential of generating a conceptual lattice from a formal background can be deemed as a conceptual clustering procedure [6]. It is more easily accepted by the end users, hence this paper chooses a FCA-like method to construct interoperation ontology from scratch.

An ontology-based FOM development framework has been proposed to introduce ontology into HLA-based simulations [8]. The Meta model corresponding to the OMT (Object Model Template) is known as World Ontology, simulation (federate) ontologies as SONT and federation ontology as FONT. The SONT specifies the object attribute architecture corresponding to a given federate simulation. The FONT specifies a common representation for all objects and interactions that are shared among different federates, and captures the relationships between the federate and common representations. But unfortunately, the ontology they adopted is light weighted ontology. That is to say, no axiom information is included to support further inferring processes. And construction of SONT is based on well defined World Ontology, thus the flexibility and independence of every subsystem is lost. Therefore, the construction method proposed in this paper is more independent and easier to apply.

3. Theoretical Foundations

Can we use ontology as a mutual understanding media for different disciplinary systems? Theoretically speaking, we can. Because on one hand the objective of collaborative product development at one time is unique; every established model is describing some aspects of the same thing. On the other hand, the concept lattice (Galois lattice) theory [23] provides a solid foundation for this method.

3.1 Definitions

Because the algebraic system defined on the concept set of collaborative product development and the partial order relations of these concepts has the same upper bound and lower bound, it can be deemed as a concept lattice [24]. The mutual understanding models of collaborative product development have a common source, a given product model, and any model involved is specialized in some aspects. They also share a common Meta data, binary stream, and any datum collaboration required is a given parse of a binary fragment. That is to say, the partial order relations, such as part-of or inherit-from, have a common ancestor, the product (T), and all the products have a common ancestor, *Thing*. Also the minimum original concept ($\perp$) defined under these partial orders are *binary characters*; it is also the public descendants of these concepts. Based on this, the paper defines related concepts are as follows:

Definition 1: collaborative product development ontology

$$O ::= (C, H_C, R_C, H_R, M, R_M, A)$$

Collaborative product development ontology O is defined as a seven tuple. C denotes a collaboration concept set of collaborative product development. H_C defines a set of partial orders on concept set C ; they give the inherit relations among the concepts involved. The concepts set and inherit relations defined on that set form a directed acyclic graph (DAG) whose source is the given model of a collaborative product and whose sink are binary fragments. R_C denotes a set of non-inherit partial order relations on concept set C ; these partial order relations correspond to concept attributes. H_R defines inherit relations on partial order relation set R_C . M is a series of collaborative product Meta ontology concepts; they give a series of inheritable instances of R_C . R_M denotes a set of partial order relations under M ; they describe the relations among elements in a Meta ontology set, and they are also the basis for collaborative product ontology reasoning. A defines a set of axioms among an ontology concept set and a Meta ontology relation set; they provide the major premises of collaborative product development ontology reasoning.

3.2 Mathematical Properties

Ordinarily, $a \wedge b$ denotes the maximum lower bound of $\{a, b\}$, and $a \vee b$ represent the minimum upper bound of $\{a, b\}$, that is $a \wedge b = \text{GLB}\{a, b\}$, $a \vee b = \text{LUB}\{a, b\}$. In a common sense, this paper uses $a_1 \wedge a_2 \wedge \dots \wedge a_n$ to denote the maximum lower bound of set $\{a_1, a_2, \dots, a_n\}$ and $a_1 \vee a_2 \vee \dots \vee a_n$ represents its minimum upper bound.

Theorem 1: operations $\wedge, \vee$ on collaborative product development ontology conception lattice $\langle C, \leq \rangle$ have properties as follows

- Idempotent law:** any $a \in C$, there exist $a \wedge a = a$, $a \vee a = a$
- Commutative law:** any $a, b \in C$, there exist $a \wedge b = b \wedge a$, $a \vee b = b \vee a$
- Associative law:** any $a, b, c \in C$, there exist $(a \wedge b) \wedge c = a \wedge (b \wedge c)$,
 $(a \vee b) \vee c = a \vee (b \vee c)$
- Absorption law:** any $a, b \in C$, there exist $a \wedge (a \vee b) = a$, $a \vee (a \wedge b) = a$

Theorem 2: suppose $\langle C, \leq \rangle$ is a conception lattice of given collaborative product development ontology, $\geq$ is the reverse of relation $\leq$. Any $a, b, c, d \in C$, there exist

- **reflexivity:** $a \leq a$; $a \geq a$
- **anti-symmetry:** $a \leq b$ and $b \leq a \Rightarrow a = b$; $a \geq b$ and $b \geq a \Rightarrow a = b$
- **transitivity:** $a \leq b$ and $b \leq c \Rightarrow a \leq c$; $a \geq b$ and $b \geq c \Rightarrow a \geq c$
- $a \wedge b \leq a$; $a \vee b \geq a$; $a \wedge b \leq b$; $a \vee b \geq b$
- $c \leq a$ and $c \leq b \Rightarrow c \leq a \wedge b$; $c \geq b$ and $c \geq a \Rightarrow c \geq a \vee b$
- $a \leq b \Rightarrow a \wedge b = a \Rightarrow a \vee b = b$
- $a \leq b$ and $c \leq d \Rightarrow a \wedge c \leq b \wedge d$;
 $a \leq b$ and $c \leq d \Rightarrow a \vee c \leq b \vee d$
- **rank preservation:** $a \leq b \Rightarrow a \wedge c \leq b \wedge c$; $a \leq b \Rightarrow a \vee c \leq b \vee c$
- **distribution inequality:** $a \vee (b \wedge c) \leq (a \vee b) \wedge (a \vee c)$;
 $a \wedge (b \vee c) \geq (a \wedge b) \vee (a \wedge c)$

- **norm inequality:** $a \leq c \Rightarrow a \vee (b \wedge c) \leq (a \vee b) \wedge c$

The theorems above are important sources of basic axioms required by collaborative product development ontology reasoning.

4 Ontology Modeling

Although the target of this paper is only to introduce heavy weighted ontology into HLA-based collaborative product development to describe the capability of federates, no one can take for granted that it is very easy work. The main reason is that before that target will be fulfilled there are a lot of preparations needed and the whole collaboration mechanism will change to take full advantage of ontology technologies. To uncover these preparations, a two layer ontology application mechanism is first demonstrated here. Then Meta ontology construction is introduced in this section. It defines the reasoning rules and templates. Based on the Meta ontology, the domain ontologies can be constructed in two different ways: FCA and transformation from the SOM files. Finally, after establishing the interoperation ontologies, a description logic-based consistency checking method is also introduced in this section.

4.1 Ontology Construction Mechanism

As Figure 1 shows, this paper presents a heavy weighted ontology-based method for interoperation modeling which not only include common attributes and algebraic characters but also axioms that describe domain semantics and support the reuse of subsystems in multiple collaborative contexts.

Figure 1 Two Layer Ontology Construction Mechanism

In order to guarantee the portability and to keep the independency of ontology from its application environment, formal semantics and abstract ontology are employed here. The formal semantics is used

as the representation means of ontologies and axioms, which focus more on the parsing of oriented concepts than application semantics. And the abstract ontologies in the abstraction layer are founded for general purposes and should cover all cases over time. When combining the content of abstract ontologies with application context which are described in the application scenario, they turn into applicable ontologies. In this way, inferring in the knowledge-based system can involve context axioms. And an abstract ontology may map into several applicable ontologies according to different contexts.

Abstract ontology has three main parts: domain knowledge base, Meta ontology knowledge and axiom knowledge. The domain knowledge base is an inferable knowledge library composed of domain knowledge and axioms. Meta ontology describes the definition and inheritance relationship of data types, and the templates of ontology with attributes. Axiom knowledge includes transformation axioms, general axioms and domain axioms. Transformation axioms define the semantic transformation equivalent relations, and are stored in the Meta ontology as rules. General axioms are the logic representations of axioms defined under the conceptual lattice in collaborative product development. These axioms are described in section 3.2. Domain axioms are professional laws or rules of a given domain, or user defined axioms. These axioms are stored in the domain knowledge base in the format of description logic.

In the inferring process, the context of the application scenario is recognized by the initial instances and the facts. The instances are used to match the roles of description logic. In a given scenario, an applicable ontology from a group of abstract ontologies forms the interoperation context of collaborative product development.

After abstract ontologies are initialized by the given interoperation context, the ontologies turn into applicable ontologies in the application layer. The applicable ontologies include three parts: domain ontology, Meta ontology and application axioms. The concepts and relations in the domain ontology are derived from the domain knowledge base by context free mappings. And the other parts of domain ontology are derived from abstract ontologies according to the interoperation context. The application axioms are represented as a series of rules or constrains.

This ontology modeling mechanism separates general information from the application environment. As time passes, the abstract ontologies can become more and more complete to cover most cases of some professional domains used in collaborative product development. This accumulating construction method is easier and more applicable than seeking a complete ontology at the outset. At the same time, the application ontologies possess all the advantages of heavy weighted ontologies. And with the development of abstract ontologies, more and more knowledge can be reused in this kind of collaboration.

4.2 Meta Ontology

The inferring of collaborative interoperation ontology is based upon Meta data. Meta ontology (MO) is the ontology which stores the fundamental knowledge of this kind of inferring. It contains basic transformation data types and necessary general axioms.

Figure 2 Collaborative Product Development Meta Ontology

The left side of Figure 2 is the concept model of collaborative product development Meta ontology and the relations among its factors, and the right side is the demonstrated structure of this ontology constructed by Protégé 3.4.

Factors of the Meta ontology include Meta class, Meta attributes, Meta data type, axioms and rules. These factors are composed of five entity categories: objects, attributes, axioms, data types and transitions. All the entities belong to Meta classes, described by multiple attributes. Meta attributes follow Minsky's [25] frame theory. Every attribute connects two Meta classes; *domain* denotes the Meta class which the Meta attribute belonged to and *value* represents the instances in the Meta attribute range.

The Meta classes form the templates of ontology construction and the Meta attributes are the templates of ontology attributes. The inheritance relation of ontology is represented by *sub-class-of* between ontology concepts, and the inheritance relation among attributes is denoted by *sub-class-of* slot in the attribute templates. The ownership of their attributes is represented by attribute pointers of concepts. The Meta data type defines the basic types which are associated with the hardware and the common intercommunication data types in collaborative product development. In fact, all the attribute values are represented by instances of given data types. Data type describes complex, customer defined data types (simple data type, enumeration data type, array data type, fixed record data type, and variant record data type), and these data types are composed from basic data types. General axioms store the content of *Theorem 1* and *Theorem*

2 in the form of description language. And transition rules define the equivalent transition rules or constraints among data types in the CLIPS (C Language Integrated Production System) way.

On the basis of Meta ontology, interoperation ontology of collaborative product development describes interoperation models in an object oriented fashion so that it is easy to inherit and extend. And this approach is compatible with the definition of HLA OMT (Object Model Template) object classes and interaction classes.

The attributes of Meta object class contain the description information of object classes in HLA object model identification tables, such as relative information about object model sponsors includes the name of the object model, version, creation date, the aim of creation, features, application domain and sponsor organization. As well, in Meta object classes there are also attributes that connect parent classes and child classes, private class attributes and instances of the Meta object class. All the ranges are composed of instances of other Meta classes. One range may correspond to one or multiple Meta classes. And each of the Meta object classes can have multiple private attributes, and every private attribute complies with the condition "Domain belongs to Meta classes, and range is the instances of Meta classes" [**Error! Bookmark not defined.**].

The Meta attribute class frame concept of object entities contains private attributes *Belongingto* and *UsingDatatype* to represent *Domain* and *Range*. The instances of Meta attribute class must be part of the Meta object classes. The range of *UsingDatatype* attribute is the instance of data type Meta classes, meaning the value of these attributes can be basic data types or complex user defined data types. Attribute *Cardinality* gives the amount of frames in a given attribute Meta class. The range of attribute *UsingRelation* is the instances of relation Meta classes, and transition relations between attributes are described by the instance of corresponding relations. And this attribute provides a graphical search path for a minimum spanning tree-based transformer.

The data type Meta classes abstract all the instances of data types, for example the basic data types as Byte, Int, Float, and String. And these basic data types are defined in the Meta ontology; they can combine to form complex data types which fulfill the requirements of collaborative product development. In the data type Meta classes, there still exists the attribute *UsingRelation* to automatically transit attributes by the data type transitions, for instance the transition between attributes Galon and Liter is done by data type transition. The transition Meta classes store the concrete methods of attributes transitions or data type transitions.

Besides these Meta classes, there are transitions among basic quantity units and data types in Meta ontology.

4.3 Domain Ontology

There are two approaches to construct domain ontologies. One approach is abstract ontology from the application scenario by FCA or adding appropriate content to SOMs. This section introduces the FCA approach to create domain ontology for collaborative product development interoperation models and the next section describes a semi automatic approach to transfer SOMs to domain ontologies.

1. Generate Formal Background. Analyze the domain context; look up verbs and their direct

objects. The objects are concepts in the given domain. Then establish syntax trees based on these concepts. This paper supposes that the relations found contain the complete information of that category in the given context.

2. Formalize the verbs and their objects abstracted. Classify them according to the dictionary. Transfer verb “xx” to “xx-able”, so that they look more like attributes and the concepts of a different layer in the conceptual lattice and are easier to understand.

3. Analysis of the context; group the objects and then construct them into abstract structural concepts. Finally, build the conceptual lattice from the formal background.

4. Through directly removing the bottommost members (data type definitions) of the conceptual lattice we can get a series of partial order relations. These formal concepts construct basic and extended concepts of interoperation model ontologies.

5. Add domain axioms in the format of description logic, in order to support inferring in the ontology fusion processes. The basic factors of these domain axioms include concepts, roles and constructors. The concepts represent the entity set. The roles denote binary relations between entities. Constructors compose atomic concepts and roles into complex concepts and roles. The constructor set determines the representation capability of a given description logic.

6. Add domain bridge axioms in the format of description logic. These axioms include relation bridge axioms and concept bridge axioms. The bridge axioms describe the interconnections between two given domains. Furthermore, bridge equivalent relation axioms and bridge equivalent concept axioms are the foundation of ontology described collaborative product development interoperation models alignment and merging.

4.4 SOM Transformation

To an existing HLA- based collaborative product development system, every participant subsystem (federate) usually has its own SOM description. Abstracting ontology from the application scenario takes time and is very costly. Furthermore, from the viewpoint of reusability, an automatic ontology construction method which takes full advantage of existing SOM is more applicable here.

The main contents of a SOM template includes:

Object model identification table: connect object model with important identification information.

Object class structure table: record namespaces of federates and federation object classes and describe their inheritable relations.

Interaction class structure table: record namespaces of federates and federation interaction classes and describe their inheritance relations.

Attribute table: denote the features of object attributes in the scope of a federate or federation.

Parameter table: denote the features of interaction parameters in the scope of a federate or federation.

Dimension table: represent the dimensions used to filter instance attributes and interactions.

Time representation table: describe the representation of time value.

User-supplied tag table: denote the tags used in the HLA Services.

Synchronization table: describe the representation and data types in the HLA synchronization service.

Transportation type table: describe the transportation mechanism used.

Switches table: store initial settings of the RTI (Run Time Infrastructure).

Data type tables: store detailed information of data representation in object models.

Notes table: store some comments.

FOM/SOM lexicon: store all the objects, attributes, interactions and parameters defined in the HLA object model.

According to the content above, this paper proposes a transformation process as follows.

1. Take identifiers from the object model identification table, and add the concepts like the name of a given collaborative product, object, interaction, and management to form the basic concept set of ontology-based collaborative product development interaction models.
2. Make the concept correspond to the name of a given collaborative product, and add three child concepts *object*, *interaction* and *manager*. At the same time, append common fields, such as *development date*, *sponsor* to these concepts. Then retrieve the object class structure table, and according to the object class structure, establish inheritance relations of concepts which are corresponding to the object classes. In the same way, establish the inheritance relations of concepts which are corresponding to interaction classes by retrieving the interaction class structure table.
3. Retrieve the attribute table; add attributes to a concept in the interoperation ontology according to ownership to the corresponding object class. In the same way add parameters to concepts by retrieving the parameter table.
4. Retrieve the data type table and, according to Meta data types, establish definitions of data types as an independent concepts hierarchy in collaborative product development interoperation ontology.
5. Add child concepts *dimension*, *time representation*, *synchronization* and *transportation type* to the product root, then retrieve corresponding tables to get detailed information about these concepts.
6. Redefine the content of the user supplied tag table and switches table as user defined data types in collaborative product development interoperation ontology.
7. Append domain axioms in the format of description logic, in order to support inferring in collaborative product development interoperation ontology fusion.

Also, the content of FOM/SOM lexicon can be retrieved from collaborative product development interoperation ontology. The common content of the note table is appended to the root concepts of the object class, and the semantics part is also described by ontology. Furthermore, because this paper adopts heavy weighted ontologies, the context of the application scenario can also be taken into consideration.

According to IEEE Standard 1516.2, this paper develops a grammar to describe SOM files.

Grammar 1. Simulation Object Model

- 1 $S \rightarrow O_M$
 - 2 $O_M \rightarrow \langle \text{objectModel } \delta \text{ name} = \delta \delta \text{ OID}_M \text{ TT}_A \text{ S}_Y \text{ T}_S \text{ D}_T \text{ N}_O \epsilon$
 - 3 $O \rightarrow \langle \text{objects} \rangle O_{CS} \epsilon$
 - 4 $O \rightarrow \langle \text{objects} \rangle \epsilon$
 - 5 $O_{CS} \rightarrow O_C O_{CS}$
 - 6 $O_{CS} \rightarrow O_C$
 - 7 $O_C \rightarrow \langle \text{objectClass name} = \delta \delta A_S O_{CS} \epsilon$
 - 8 $O_C \rightarrow \langle \text{objectClass name} = \delta \delta A_S \epsilon$
 - 9 $O_C \rightarrow \langle \text{objectClass name} = \delta \delta O_{CS} \epsilon$
 - 10 $O_C \rightarrow \langle \text{objectClass name} = \delta \delta \epsilon$
 - 11 $A_S \rightarrow A A_S$
 - 12 $A_S \rightarrow A$
 - 13 $A \rightarrow \langle \text{attribute name} = \delta \delta \text{ dataType} = \delta \delta \text{ dimensions} = \delta \text{ transportation} = \delta \delta \epsilon$
 -
 - 94 $\epsilon \rightarrow />$
 - 95 $\epsilon \rightarrow </\delta >$
 - 96 $\delta \rightarrow (\text{Any string does not contain terminal})$
-

ϵ and δ are deemed as terminals. And because the strings like “ $\langle \text{objectClass name} = \delta \delta$ ” can be recognized as one string, this paper also looks at these strings as independent terminals.

Based on rule 2, a gross deterministic finite state automaton (Figure 3) can be built to describe this grammar.

Figure 3 Gross Deterministic Finite State Automaton

Rule 3 to rule 13 describe the grammar of the objects part in the SOM files. To avoid redundant statements, this paper takes the object block of the SOM file as an example to demonstrate the transformation process.

The augmented grammar of the object block is listed as follows:

Grammar 2. Augmented grammar of the object block

- 3' $O' \rightarrow O$
 - 3 $O \rightarrow \langle \text{objects} \rangle O_{CS} \epsilon$
-

-
- 4 $0 \rightarrow \langle \text{objects} \rangle \epsilon$
 - 5 $0_{cs} \rightarrow 0_c 0_{cs}$
 - 6 $0_{cs} \rightarrow 0_c$
 - 7 $0_c \rightarrow \langle \text{objectClass name} = \delta \delta A_S 0_{cs} \rangle \epsilon$
 - 8 $0_c \rightarrow \langle \text{objectClass name} = \delta \delta A_S \rangle \epsilon$
 - 9 $0_c \rightarrow \langle \text{objectClass name} = \delta \delta 0_{cs} \rangle \epsilon$
 - 10 $0_c \rightarrow \langle \text{objectClass name} = \delta \delta \rangle \epsilon$
 - 11 $A_S \rightarrow A A_S$
 - 12 $A_S \rightarrow A$
 - 13 $A \rightarrow \langle \text{attribute name} = \delta \delta \text{dataType} = \delta \delta \text{dimensions} = \delta \text{transportation} = \delta \delta \rangle \epsilon$
-

According to the rules mentioned above, the reachable item sets and the transitions between them can be found. Then the state machine can be described as Figure 4.

Figure 4 State Machine of the object block

Denoting “<objects>” as α , “<objectClass name= $\delta \delta$ ” as β and “<attribute name= $\delta \delta$ dataType= $\delta \delta$ dimensions= δ transportation= $\delta \delta$ ” as γ , table 1 demonstrates the parsing table of this grammar.

Table 1 Parsing table for the object block

<i>state</i>	<i>action</i>					<i>goto</i>				
	α	β	γ	ϵ	$\$$	O	O_C	O_{CS}	A_S	A
2 ₀	s ₂					2 ₁				
2 ₁					acc					
2 ₂		s ₆		s ₃			2 ₅	2 ₄		
2 ₃					r4*					
2 ₄				s ₇						
2 ₅		s ₆		r6*			2 ₅	2 ₈		
2 ₆		s ₆		s ₁₁			2 ₅	2 ₁₀	2 ₉	2 ₁₂
2 ₇					r3*					
2 ₈				r5*						
2 ₉		s ₆		s ₁₆			2 ₅	2 ₁₇		
2 ₁₀				s ₁₄						
2 ₁₁		r10*								
2 ₁₂			s ₁₃	r12*					2 ₁₅	2 ₁₂
2 ₁₃			r13*							
2 ₁₄		r9*								
2 ₁₅				r11*						
2 ₁₆		r8*								
2 ₁₇				s ₁₈						
2 ₁₈		r7*								

The interoperation ontology is built during the parsing process. The attached actions are mentioned in table 2.

Table 2 Attached Parsing actions

action	Attached action
s ₂	Add “ <i>objects</i> ” to product root
s ₃	End “ <i>objects</i> ” concept hierarchy
s ₆	Create a new child concept under current concept
s ₇ , s ₁₁ , s ₁₄ , s ₁₆ , s ₁₈	Current concept change to its parent concept
s ₁₃	Create a new property for current concept

In this way, a deterministic finite state automaton is built to automatically transform the SOM file into an interoperation ontology. Compared with the existing interoperation ontology construction method, this method is easier to apply in HLA-based collaborative product

development systems. It also maximally reuses existing knowledge and is compatible with legacy systems.

4.5 Consistency Checking

No matter how to establish collaborative product development interoperation ontology, there is a potential for inconsistency. The inconsistency can include terminology structure inconsistency, conception redundancy and trivial equivalent relation (synonym). Since the conceptual lattice is the foundation of collaborative product development interoperation ontology, the conflict of concept hierarchy will damage the structure of the lattice. Terminology structure inconsistency must be eliminated before further usage of the interoperation ontology. Conception redundancy refers to the attributes of concepts that are not compatible, and result in no instances existing. The trivial equivalent relations refer to the one to one equivalent concepts in the same interoperation ontology. To avoid redundant computing in ontology integration processes, no trivial equivalent relations should exist.

Collaborative product development interoperation ontology checking is a semi automatic inferring process in a description logic described ontology, which discovers equivalent concepts, proposes suggestions to the wrong terminology layer and redundant concepts. Formalized descriptions are the foundation for discovering terminology layer misplacing. Since description logic cannot denote equivalent concepts [26], the result of this process is more like the suggestions to adjust ontology rather than directly changing the structure of a given ontology. The confirmation of domain expert is also needed here, and then the modification can be made manually.

Description Logic (DL) is also named as terminology logic or conception representation language. It is an object based formal knowledge representation method, a predictable subset of first order predicate logic and it provides predictable inferring abilities. The main factors of DL are *Concept* and *Role*. A typical DL system is composed of four components: constructor set (describe concepts and roles), TBox assertion set, ABox assertion set and inferring mechanism on TBox and ABox.

The TBox is an axiom set which describes domain structures. The ABox is an axiom set which describes individual facts. Classification in the TBox is meant to place a new concept into the conception structure. If every concept is placed appropriately, there are no conflicts in the conception hierarchy of the TBox. The appropriate place for a concept is the layer between the special concept which implicates it and the most common concept it contains. The assertions in the ABox should be non-exclusive to each other, and form a compatible system with the TBox terminology. And consistent checking can be made by the examination of implication relations among the concepts hierarchy.

To formally represent checking methods of implication and assertion compatibility, a formal operator is introduced. Besides concepts and roles, there still is included an interpretation function $\cdot^I$ and a nonempty collection Δ^I (interpretation domain). $\cdot^I$ corresponds to every concept A to $A^I = \Delta^I$, and every role R to binary relation $R^I = \Delta^I \times \Delta^I$. The Attributive Language Complement (ALC) is the most basic description logic which adopts five operators: conjunction $\sqcap$, disjunction $\sqcup$, not $\neg$, existential quantifier $\exists$ and universal quantifier $\forall$.

After the formal operator is introduced, the inferring of description logic can be formally operated. The most important inferring process is consistent checking of the description logic. To TBox T ,

consistency checking between concept C and T is to examine if there exists model I of T which satisfies $C^I \neq \emptyset$. To ABox A , consistency checking means examining if there is any model of A . To knowledge base (T, A) , consistency checking is made to guarantee that there is only one A model, and there exists one T model based on A .

The consistency checking of description logic is also called satisfiability checking. If the terminology, assertion set and conception attributes in the knowledge base result in satisfiability, they are compatible and there exists a model I which can satisfy C . That is to say, there is no redundant concept in the given concept set.

In TBox T , detecting $C \subseteq D$ is to check if $C^I \subseteq D^I$ is satisfied by all models I of T . If the answer is yes, the classification of TBox is correct.

All inferring problems of description logic can be transformed into consistency checking [27]. So this paper reaches implication checking by consistency checking, and furthermore determines the correctness of the conception structure of the TBox, then makes a judgment of the conception structure of ontologies.

The Tableaux algorithm proposed by Schmidt and Smolka [28] can perform satisfiability checking (consistency checking) of ALC concepts in Polynomial Time. This algorithm employs corresponding rules of four operators: conjunction $\sqcap$, disjunction $\sqcup$, existential quantifier $\exists$ and universal quantifier $\forall$. The expressions of conception consistency can be inferred by these rules. When the result is empty $\emptyset$, conception consistency is not satisfied.

There are a lot of reasoning engines which realize Tableaux algorithm, such as FaCT++ [29], Pellet [30], Racer Pro [31]. Because Racer Pro can work with Protégé harmony, and automatically generate suggestions to adjust the conception hierarchy of ontologies, this paper adopts the reasoning engine of Racer Pro to perform consistency checking.

By transformation, this paper starts from consistency checking the description system by the Tableaux algorithm and reaches redundant concepts checking of a given ontology and conception structure checking. The satisfiability checking of concepts from a different conception layer can discover redundant concepts and concept misplacement, and satisfiability checking of concepts from the same conception layer can perform synonym checking by inference from *theorem 2*.

Inference 1: Suppose C_i and C_j are two different concepts of the same collaborative product development interoperation ontology, if there exist $C_i \subseteq C_j$ and $C_j \subseteq C_i$, then C_i and C_j are synonyms.

By all the means mentioned above, this method can eliminate all the inconsistencies of collaborative product development interoperation ontology.

6 Case Demonstration

This case is designed to demonstrate the feasibility of the proposed construction method of HLA-

based collaborative product development interoperation models. In this case, a Meta ontology is first established to provide a template for interoperation models according to standard IEEE 1516, which includes ontology structure, data types and mandatory attributes. Domain ontologies can be deemed as instances of this Meta ontology. Then two ontologies are constructed by the FCA method based on this Meta ontology. To demonstrate the proposed method, another domain model is described by an XML file. According to the transformation rules mentioned above, it was transformed to the third ontology. Compared with constructing domain models only by negotiations, the FCA method, with the help of Meta ontology, can accelerate the negotiation process by standard workflows, and ontologies can keep more details of the negotiation than XML files. To XML described domain models, the proposed transformation method can almost automatically transform these models to domain ontologies. But domain ontologies can save more semantic content than old style XML files.

A typical collaborative product development unit can be described with three main parts: scene creator, localization server and decision support system. The scene creator generates scenes for collaboration jobs. The localization server keeps tracking important entities and the decision support system makes final decisions or suggestions about future collaboration steps. The collaboration information among these three independent systems lies in the scene provided by the Scene Creator, localization information supported by the Localization Server and instructions from the Decision Support System. In this paper, the Localization Server is supposed to be a real time tracking system, so it does not need instructions.

Figure 5 A Typical Collaborative Product Development Unit

The Meta ontology is built as Figure 6 shows. The top concept is *product*, which indicates the objective of the future collaboration. According to standard IEEE 1516, there are nine basic categories of domain models: *datatypes*, *dimensions*, *interactions*, *manager*, *notes*, *objects*, *synchronizations*, *tags*, *time* and *transportation*. For convenience, a new category of *metaDefinitions* also add as the child

concept of *product*. Category *datatypes* include *BasicDataRepresentations*, *simpleDatatypes*, *fixedRecordDataTypes*, *enumeratedDataTypes*, *arrayDataTypes* and *variantRecordDataTypes*. *BasicDataRepresentations* have 14 instances: *HLAoctet*, *HLAfloat32BE*, *HLA32LE* etc. *simpleDatatypes* have three instances: *HLAASCIIchar*, *HLAbyte* and *HLAunicodeChar*, which are composed of the instances of *BasicDataRepresentations*. And the content of other data types can be told by their names.

The root of all interaction classes is *HLAinteractionRoot*, which is the child concept of *interactions*;and then *interactionManager* and *Federate*. To any *Federate*, there are four kinds of interactions: *Adjust*, *Report*, *Request* and *Service*. There are four basic sub-concepts of Concept *Adjust*: *ModifyAttributeState*, *SetExceptionLogging*, *SetServiceReporting* and *SetTiming*. For *Report*, there are 14 sub-concepts; *Request*, 10 sub-concepts and *Service*, 32 sub-concepts.

Concept *manager* has two sub-concepts of *interactionManager* and *objectManager* in HLABased collaborative product development systems. They are also the descent concepts of *interactions* and *objects*.

The sub-concepts of *metaDefinitions* are *alternatives*, *Capabilities*, *endianings*, *enumerators* and *fields*. The instances of these concepts give value options for attributes of some Meta ontology attributes. And the transformation rules are partly defined on the instances of concept *metaDefinitions*.

The root of all object classes is *HLAobjectRoot*, which is the child concept of *objects*;and then *objectManager* and *Federate*.

Ontology inferring is mostly used in the later stages of ontology construction for collaborative product development interoperation models, such as ontology fusion, ontology maintenance and ontology evolution. So in this case, the general axioms storage and transition rules are not demonstrated. This content will be reported in another paper related to ontology maintenance in a hierarchical federated collaborative product development system.

When constructing domain ontologies, this Meta ontology is used as a template. After related concept and relations from FCA analysis are put in proper positions, domain ontology is finished.

Figure 6 Demo Meta-Ontology

By the FCA method, ontologies of the Scene Creator and the Decision Support System have been established manually as Figure 7 shows.

The Scene Creator used here is the BIM (Building Information Management) System, which can generate a virtual environment according to any valid ifc (Industry Foundation Classes) files with the IFCViewer. To compute a formal background, a serial of real operation workflows are described in natural language first. These workflows form the domain context of the BIM system. After analyzing this domain context step by step as the FCA mentioned, the BIM ontology is reached.

New added sub-concepts of *HLAobjectRoot* are *human*, *IFCBuilding* and *IFCViewer*. Among these, *IFCBuilding* is the key concept. The concepts *human* and *IFCViewer* are just used here to give a formal definition of *IFCBuilding* in the format of Description Language. *IFCBuilding* includes *BuildingElementProxy*, *IFCBuildingElement*, *MultiStoreyBuilding* and *Space*. *BuildingElementProxy* is something that can indicate states of the *IFCBuildingElement*, such as *Hygrometer* and *Thermostat*. *IFCBuildingElement* consists of some basic components of one building, such as *BasicWall (Without Openings)*, *Floor*, *Furnishing_Element*, *Opening*, *Roof* and *Stair*. *MultiStoreyBuilding* has more than one *storey*.

In the same way, the FMM (Facility Maintenance Management) system ontology is constructed. The newly added sub-concepts of *HLAobjectRoot* are *Building*, *BuildingDescription*, *BuildingElement*, *Point* and *Project*. *Building* is about the construction, such as *BrickWood*, *Frame*, *Steel*, *SteelReinforcedConcrete* and *Other*. *BuildingDescription* is something describing part of the whole building, such as indicator (*Condition*), *Location* and *Map*. *BuildingElement* can be divided into two categories, *Asset* and *Equipment*.

The consistency checking of these ontologies is done by the embedded checking functions of Protégé.

These two ontologies prove the feasibility of the FCA method. Compared with the negotiation method to construct domain models, computing formal background is also a boring and time-consuming work. But when ontology and FCA are adopted, there are some advantages. FCA gives guidance for negotiations. Ontology helps keep the consistency of the domain models, and can store more content than text files.

Figure 7 Demo Ontology of FCA Method

If domain models are already written in the style of SOM files, such as the following XML file (as Figure 8), The proposed SOM transformation method is much useful here.

The *HLAInteractionRoot* is extended by concept *correctiveAction*. Concept *correctiveAction* has two classes: *event* and *InternalEvent*. *InternalEvent* includes *LowBattery* and *Moving*.

The *HLAObjectRoot* is extended by the concepts *Asset*, *AssetOwner*, *Coordinate*, *Location*, *Map* and *Tag*. *Tag* is the equipment which can generate signals for location information, which can be classified into *ActiveTag* and *InactiveTag*. Every *Asset* is related to at least one *Tag*. *Asset* can be *ArrangeAsset* or *ClassifiedAsset*. *Coordinate* is the representation of a *point* in 2-D, 2 and half Dimension or 3-D environment. *Location* is the description of *Asset*.

```

<?xml version="1.0" ?>
<!DOCTYPE objectModel (View Source for full doctype...)>
- <objectModel DTDversion="1516.2" name="Localization" type="SOM" version="1.0 Alpha" date="2010-04-28" purpose="An object model
  for localization system" appDomain="Location tracking" sponsor="National Research Council, Canada" pocName="Mr. Hongbo Sun"
  pocOrg="CCCT" pocPhone="1-519-430-7021" pocEmail="hongbo.sun@nrc.gc.ca" references="http://10.10.18.156:8080/asset-
  manager-web/services/LocatorAPIService?wsdl" other="See AeroScout User Guide for more information">
- <objects>
- <objectClass name="HLAobjectRoot" sharing="Neither">
  <attribute name="HLAprivilegeToDeleteObject" dataType="NA" updateType="NA" updateCondition="NA"
    ownership="NoTransfer" sharing="Neither" dimensions="NA" transportation="HLAreliable" order="TimeStamp" />
+ <objectClass name="Manager" sharing="PublishSubscribe" semantics="">
  <objectClass name="Asset" sharing="PublishSubscribe" semantics="Mobile things to be tracked">
    <attribute name="AssetID" dataType="HLAinteger64BE" updateType="NA" updateCondition="NA"
      ownership="DivestAcquire" sharing="PublishSubscribe" dimensions="NA" transportation="HLAreliable"
      order="TimeStamp" semantics="Identify of given asset" />
    <attribute name="AssetName" dataType="HLAASCIIstring" updateType="Conditional" updateCondition="Owner wants"
      ownership="DivestAcquire" sharing="PublishSubscribe" dimensions="NA" transportation="HLAreliable"
      order="TimeStamp" semantics="Name of given asset" />
    <attribute name="AssetDescription" dataType="HLAASCIIstring" updateType="Conditional" updateCondition="Owner
      wants" ownership="DivestAcquire" sharing="PublishSubscribe" dimensions="NA" transportation="HLAreliable"
      order="TimeStamp" semantics="Description of given asset" />
    <attribute name="AssetImage" dataType="BLOB" updateType="Conditional" updateCondition="Owner wants"
      updateConditionNotes="3" ownership="DivestAcquire" sharing="PublishSubscribe" dimensions="NA"
      transportation="HLAreliable" order="TimeStamp" semantics="Image of given asset" />
    <attribute name="AssetCreateDate" dataType="Date" updateType="NA" updateCondition="NA" ownership="DivestAcquire"
      sharing="PublishSubscribe" dimensions="NA" transportation="HLAreliable" order="TimeStamp" semantics="Creation
      date of given asset" />
  - <objectClass name="ArrangedAsset" sharing="PublishSubscribe" semantics="The assets which are owned by some
    organization.">
    <attribute name="Owner" dataType="AssetOwner" updateType="Conditional" updateCondition="Change asset
      ownership" ownership="DivestAcquire" sharing="PublishSubscribe" dimensions="NA" transportation="HLAreliable"
      order="TimeStamp" semantics="Owner of given asset" />
  </objectClass>
  - <objectClass name="ClassifiedAsset" sharing="PublishSubscribe" semantics="The assets which are classified by some
    groups.">
    <attribute name="Group" dataType="HLAASCIIstring" updateType="Conditional" updateCondition="Change asset
      group settings" ownership="DivestAcquire" sharing="PublishSubscribe" dimensions="NA"
      transportation="HLAreliable" order="TimeStamp" semantics="Group of given asset" />
  </objectClass>
- </objectClass>
- </objects>
- </objectModel>

```

Figure 8 Demo SOM

By this method, all the elements in the SOM files can be properly transformed. And after taking full advantage of the protégé plug-in, the consistency of this ontology was well kept. The interoperation ontology of the localization system is shown in Figure 9, which is established by the proposed automata.

Figure 9 Localization System Ontology Demo

From the demonstration of interoperation ontologies, it is shown that they can represent the knowledge of every professional domain well and have good potential to support future collaborative operations in a HLA-based collaborative product development system.

7 Discussion

Although ontology is a well-known method to resolve mutual understanding among heterogeneous systems, it still needs more work before handling real industrial problems.

The objective of this research is to establish a semantic-based environment supporting HLA-based collaborative product development. There are still many other issues in this research area, including dynamic adjustment of collaborative ontology, federation ontology; ontology fusion; evolution of Meta Ontology, and so on. This paper proposes an ontology-based construction method for interpretation models of HLA-based collaborative product development, which is the first step of the whole research.

This research is arranged in several steps: first, establishing domain models by ontologies; second, fusing them into a collaboration ontology to form a mutual understanding for future collaborations; third, maintaining a collaboration ontology during collaborations by integrating the ontology tool with RTI (RunTime Infrastructure); fourth, designing a mechanism for collaboration knowledge reuse; finally, developing an ontology-based federated integration for CPD. This paper pays particular attention to the first step, and future papers will address the other issues.

Before HLA-based collaboration starts, FED files are a central part of the necessary preparations. Negotiation is a common way to form SOM and FOM files, especially FOM files. This kind of negotiation is always boring and time-consuming. Ontology fusion can greatly improve the efficiency of the preparation for future collaborations, and the proposed method is the basic work for ontology fusion. Furthermore, by embedded consistency checking functions of Protégé, the quality of the domain models is improved.

In the proposed case, three systems have similar complexity. To write the XML file (961 lines) spent nearly one month, and to establish the other two ontologies only took one week each. The workload for negotiation and formal background computation are almost the same. But to organize the result proves different. There are also some FED file construction tools, such as LabWorks, which usually provide user friendly interfaces to fill the negotiation results in the proper position of text files. However, it is not helpful for the organization of interconnected knowledge.

Improvement of the preparations of HLA-based CPD looks at two aspects: one is that the proposed method supports the ontology fusion process, which can greatly improve efficiency; the other is that ontology is a better method to organize knowledge than other means.

In this research, the Meta ontology modeling for domain models always presents itself as a multi-disciplinary problem. It is not easy to find a general way of establishing Meta models for all systems from different disciplines. So in this paper, the Meta ontology will only focus on HLA-related information and general axioms. When constructing new domain ontologies, Meta ontology presents itself as a template. Some duplicate information does not need repeating.

When collaboration ontology is used in collaborations, from time to time, federates can join in or resign from a collaboration. In such a circumstance, the question surfaces of how to maintain the efficiency and consistency of the collaboration ontology. Ontology maintenance is the issue and it will be reported in another paper.

The key limitation of this method is that it is only suitable for a HLA-based CPD environment. When it is extended to a general CPD environment, some new problems will be raised. Having no FED files is the main challenge of this research. It is because in a typical HLA-based CPD environment, SOM and FOM files are well-known for their mutual understanding among heterogeneous systems. They also have well-established standards to follow. They provide a solid ground for future research. However, it is very hard to find an alternative in the general CPD environment. Lack of RTI can be another challenge. In a typical HLA-based CPD environment, RTI is the distributed middleware which connects all federates (heterogeneous systems), and is in charge of the communication among these systems. The next research steps will be to focus on how to integrate ontology tools into RTI. If there is no such software, the next steps are extremely hard to complete.

8 Conclusion

In HLA-based collaborative product development, the most difficult issue is not to establish a collaborative system, but to adaptively adjust interface codes of existing systems and to negotiate among multiple disciplinary domains [32]. Existing methods always assume that the ontology already exists. However, ontology construction itself is a big challenge and the construction of professional domain ontologies for the coming HLA-based collaborations have their own characteristics differing from general purpose ontologies, including the emphasis on functional semantics, implicit consistency and HLA compatibility. This paper proposed a novel ontology construction method to establish interoperation ontologies as a representation method of every federate involved in HLA-based collaborative engineering environments. And this kind of environment is composed of several independent subsystems with different professional domains. The main part of this method is three algorithms: an FCA-like construction algorithm, an automatic SOM transformation method and a consistency checking algorithm.

The proposed method enjoys several remarkable advantages which are more suitable for HLA-based multidisciplinary collaborations. First, this method is built on firm theoretical foundations and formalization definitions. It is more predictable when used. Second, it reduces the workload of the domain experts when preparing collaboration among several independent professional domains by introducing automatic transformation. And for the same reason, it improves the efficiency of preparation for future collaborations. Third, different from most other ontology integration methods which use literature distance, this method employs heavy weighted ontology to construct ontologies. Axioms, and bridge axioms are all taken into consideration. It does well in keeping the knowledge of every professional domain. Fourth, since ontology is used in this method, the reuse of resources and expandability of existing systems are greatly enhanced. Finally, the construction method of collaborative ontology also helps to accumulate knowledge and furthermore to build relatively complete models of the same object from different aspects. The applicability of this method has been shown by a concrete case demonstration.

9 Acknowledgment

This work is supported by Chinese National High-tech Research and Development Program (863 Program, Grant No. 2009AA110302) and Chinese Nature Science Foundation (Grant No. 60874066).

References

- [1] W. Shen, Q. Hao and W. Li, Computer supported collaborative design: Retrospective and perspective, *Computers in Industry*. Vol. 59 (2008), 855–862.
- [2] W. Fan and T. Xiao, Collaboration Integrated Platform of Design, Simulation and Optimization for Complex Products, *SCIENCE & TECHNOLOGY REVIEW*, China. vol. 226(4),15-24.
- [3] H. Sun, T. Xiao, S. Tang, Research on Federation-Based Pragmatic Integration Framework, *Proceedings of 2009 World Congress on Computer Science and Information Engineering (CSIE 2009)*. Los Angeles/Anaheim, USA, Apr. 2009.
- [4] X. Bu and N. Zhao, Information Integration for Collaborative Product Development with an Ontology-Based Approach, *International Symposium on Information Engineering and Electronic Commerce*, 2009. IEEC '09. Ternopil, Ukraine, May. 2009, 510-514.
- [5] IEEE Computer Society. IEEE standard for modeling and simulation (M&S) high level architecture (HLA)- object model template (OMT) specification(IEEE Std 1516.2- 2000). NewYork: The Institute of Electrical and Engineers, 2001.
- [6] W. Zhou, Z. Liu and H. Chen, A Survey of the Research about Both FCA and Ontology. *Computer Science*, China. 2006, vol. 33(2),8–12.
- [7] F. Fürst and F. Trichet. Axiom-based ontology matching: a method and an experiment, research report, Mar. 2005.
- [8] T. Rathnam, Using Ontologies to Support Interoperability in Federated Simulation, Thesis for Master's degree, Georgia Institute of Technology. Aug. 2004.
- [9] W.M. Cheung and P.G. Maropoulos, A Novel Knowledge Management Methodology To Support Collaborative Product Development, *Digital Enterprise Technology: SESSION 2*, Springer US: New York, US, Sep. 2007, 201-208.
- [10] M. Ciocoiu, M. Gruninger, and D.S. Nau, ontologies for Integrating Engineering Applications, *Journal of Computing and Information Science in Engineering*, 2001, vol. 1(1), 12-22.
- [11] A.J. Duineveld, R. Stoter, M.R. Weiden, B. Kenepa, and V.R. Benjamins, WonderTools?—A comparative study of ontological engineering tools, *International Journal of Human-Computer Studies*, 2000, vol. 52, 1111-1133.
- [12] H.K. Lin and J.A. Harding, An Ontology Driven Manufacturing System Engineering Moderator for Global Virtual Enterprise Teams, *Proceedings of the 1st International Conference on Manufacturing Research (ICMR)*, Glasgow, UK, 2003, 365-370.
- [13] S. Tang, T. Xiao and W. Fan, A collaborative platform for complex product design with an extended HLA integration architecture. *Simulation Modelling Practice and Theory*. Sep. 2010, vol. 18(8),1048-1068.
- [14] Y. Yang, L. Song and X. Zhang, Organization-Oriented Simulation of Collaborative Product Development Process Based on Designer's Agent Model, *11th International Conference on Computer Supported Cooperative Work in Design*, 2007. CSCWD 2007. Melbourne, Australia, Apr. 2007, 309-314.
- [15] J.P. Harrison, B. Christensen, J. Bianco and M. Gulli, Virtual collaborative simulation environment for integrated product and process development, *Proceedings of 5th IEEE International Symposium on High Performance Distributed Computing*, 1996, Syracuse, US, Aug. 1996, 19-22.
- [16] J. Ji, D. Zhang, S. Li, B. Wu and B. Chen, Complex Product Collaborative Design and Simulation Based on Product Master Model Technology, *2008 International Conference on Intelligent Computation Technology and Automation (ICICTA)*. Hunan, China, Oct. 2008, vol.1, 1012–1016.
- [17] T. Xiao and W. Fan, HLA based Integrated Platform for Collaborative Design, Simulation and Optimization, *JOURNAL OF SYSTEM SIMULATION*, China. 2008, vol.20(13),3542-3547.
- [18] J. Huang, J. Hao, X. Zhao and J. Gong, Basic Study on RTI Interoperability, *Second International Conference on Computer Modeling and Simulation*, 2010. ICCMS '10. Sanya, China, Jan. 2010, vol.2, 556-559.
- [19] H. Zhang and D. Chen, An approach of Multidisciplinary Collaborative Design for Virtual Prototyping in Distributed Network Environment, *10th International Conference on Computer Supported Cooperative Work in Design*, 2006. CSCWD '06. Nanjing, China, May. 2006, 1-6.

- [20] S.J.E. Taylor, L. Behli, X. Wang, S.J. Turner and J. Ladbroke, Investigating distributed simulation at the Ford Motor Company, Proceedings. Ninth IEEE International Symposium on Distributed Simulation and Real-Time Applications, 2005. DS-RT 2005. Montreal, Canada, Oct. 2005, 139-147.
- [21] J. Yu, and Y. Dang, Review on Ontology Integration, Computer Science, China, 2008, 35 (7) 9-14.
- [22] X. Peng and W. Zhao, An Incremental and FCA-Based Ontology Construction Method for Semantics-Based Component Retrieval, Seventh International Conference on Quality Software, 2007. QSIC '07. Portland, US, Oct. 2007, 309-315.
- [23] R. Wille, Concept lattices and conceptual knowledge systems. Computers & Mathematics with Applications, vol. 23 (1992), 493-515.
- [24] K. Qu, J. Liang, J. Wang and Z. Shi, The algebraic properties of Concept Lattice, Journal of Systems Science and Information, 2004, 2(2) 271-277.
- [25] M. Minsky, A Framework for Representing Knowledge. The Psychology of Computer Vision. P.H. Winston(Ed.). New York: McGraw-Hill, 1974, 211-277.
- [26] Y. Zhang and S. Li, Research on Formal Categorical Ontologies, Computer Science, China. vol. 33(9), 1-3.
- [27] F. Donini, M. Lenzerini, D. Nardi and A. Schaerf, Reasoning in description logics, In G. Brewka editor, Principles of Knowledge Representation and Reasoning, Studies in Logic, Language and Information, CLSI Publications, 1996, 193 — 238.
- [28] V. Haarslev and R. Moller, “RACER system description”, In Proc. of the Int.Joint Conf. on Automated Reasoning (IJCAR'2001), vol. 2083 of Lecture Notes in Artificial Intelligence, 2001, Springer, 701-705
- [29] OWL:FaCT ++. <http://owl.man.ac.uk/factplusplus/> (Accessed Sept.1, 2006)
- [30] Pellet OWL Reasoner. <http://www.mindswap.org/2003/pellet/index.shtml> (Accessed Sept.1, 2006)
- [31] Racer Systems GmbH & Co. KG. <http://www.racer-systems.com/index.phtml> (Accessed Sept. 1, 2006)
- [32] B. Li, X. Chai, G. Xiong, W. Zhu, C. Quan, H. Zhang and X. Wang, Research and Primary Practice on Virtual Prototyping Engineering of Complex Product, JOURNAL OF SYSTEM SIMULATION, China. Mar. 2002, vol.14 (3), 336-341.