Ontology maintenance in a hierarchical federated collaborative product development environment

Sun, H.; Fan, W.; Shen, W.; Xiao, T.; Chen, X.

https://doi.org/10.1109/CSCWD.2011.5960073

Ontology maintenance in a hierarchical federated collaborative product development environment

NRCC-54426

Sun, H.; Fan, W.; Shen, W.; Xiao, T.; Chen, X.

August 2011

A version of this document is published in / Une version de ce document se trouve dans:
15th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2011), Lausanne, Switzerland, June 8-10, 2011, pp. 1-9

Ontology Maintenance in a Hierarchical Federated Collaborative Product Development Environment

Hongbo Sun[1], Wenhui Fan[1], Weiming Shen[2], Tianyuan Xiao[1], Xin Chen[1]
[1] *National CIMS Engineering Research Centre, Tsinghua University*
Beijing, China, 100084
[2] *Centre for Computer-assisted Construction Technologies, National Research Council*
London, Ontario, Canada, N6G 4X8
sunhongbo02@tsinghua.org.cn

***Abstract*—This paper presents a novel approach aiming to dynamically maintain the collaboration ontology in the execution of an ontology-based federated collaborative product development system when a federate joins or has resigned from a given federation. The proposed approach includes two algorithms: ontology maintenance (+) and ontology maintenance (-), corresponding to joining and resigning situations. It adopts an axiom-based deduction ontology fusion strategy, and takes heavy-weighted ontologies into consideration. It can find all the explicit and derived inter-ontology relations, and furthermore it reaches the active upper bounds of implicit equivalent inter-ontology relations searching. This paper also discusses some implementation issues on the basis of TH_RTI, a RTI (Run Time Infrastructure) version developed by National CIMS ERC, Tsinghua University. The proposed approach has great potential to improve the efficiency of ontology-based federated collaboration executions, reduce the work load for adaptive adjustment of ever-existing platforms, and enhance the applicability and flexibility of collaborative product development systems.**

***Keywords*—Collaborative product development (CPD), HLA - High Level Architecture, Ontology Maintenance.**

I. INTRODUCTION

With the rapid advancement of information and communication technologies, globalized businesses face extremely complicated operations and, as such, require greater ability to solve the problems introduced by them. Adopting CPD makes full use of several independent development systems, and enhances their ability at the same time [1]. CPD systems often include functions, such as collaborative design, collaborative simulation and collaborative optimization. They require data and information like CAD digital models, CAE analysis and optimization results [2]. These requirements also accelerate the need for dynamic cooperation of existing computing resources in order to work together harmoniously. Since simulation is a key characteristic of CPD, HLA (High Level Architecture) has been adopted as the basic architecture for these kinds of integrations [3].

However, a HLA federation also possesses several shortcomings that limit its usage. First, its main purpose is to solve problems in the realm of collaborative simulation. When this method is adopted by product development research areas, other than simulation, a lot of new challenges remain, for example the charging method, resources utilization, task scheduling, task immigration and fault tolerance. Second, it does not touch upon the latest technologies such as service science, dynamic self adaptive API, ontology and semantics [4]. Third, the objective of a given HLA federation is usually a predetermined simulation and all the preparation is made for one time simulation. This is not in accordance with the principle of reusability. Then, the description of management functions is relatively simple. Some important functions were not included, such as fault tolerance, intelligent update and consistent sustain. Last, but not the least important, is that there are so many agreements outside the federation system. What is worse is that they are not guaranteed by any workflow or software. This does real harm to applicability and the robustness of federated applications. To address these problems, a hierarchical federated integration architecture has been proposed in [5]. In hierarchical federated integration architecture, there may be several active application federations at a time, and the integration software cannot stop to recompile interface codes for satisfying dynamic collaboration requirements.

On the other hand, in a HLA-based CPD environment, a FOM (Federation Object Model) file describes the data and information exchange standard of a given simulation, and they are keys to mutual understanding during collaborative operations. But the construction and modification of a FOM needs multidisciplinary professional knowledge and technologies [6]. An ontology-based method has been successfully explored to use collaboration ontologies as an alternative to the use of a FOM file in HLA-based systems [7].

In this paper, an ontology maintenance method is introduced to support the dynamic adjustment of the collaboration ontology when an existing federate resigns from its federation or a new federate joins a federation. Section 2 introduces related efforts towards ontology-based CPD and reviews the state of art about ontology maintenance, analyses the requirements of this application problem, identifies some limitations and discusses the outline of this research. In section 3, the algorithms supporting the proposed method are described in depth. The definitions of basic concept, functions, graphs and relations used in CPD ontology maintenance are given in a

formalized matter. Then two main algorithms are introduced here: ontology maintenance (+) and ontology maintenance (-), which are corresponding to federate joining and resigning, respectively. Section 4 is about some implementation issues about ontology-basic RTI (Run Time Infrastructure). The conclusions and some discussions are reported in section 5.

II. RELATED WORK

The objective of this research is to establish a semantics-based environment that supports CPD. Some effort has already been made and reported in several papers, such as HLA-based semantic environment, modeling and consistency checking for domain ontologies, and domain ontologies fusion to a collaborative ontology. A hierarchical federated integration has been proposed to support the HLA-based semantic environment. A FCA-like modeling method and an automata for transformation from SOM files to domain ontologies also has been submitted for publication. Ontology fusion has been reported in the IEEE International Conference of SMC 2010.

This paper is closely related to hierarchical federated integration and ontology fusion, so first, a brief introduction to them is given in this next section.

A. Hierarchical Federated Integration

Since CPD systems accelerate the need for dynamic cooperation of existing computing resources in order to work together harmoniously, these requirements brought much more complexity to contemporary computing technology and operational problems. They lead to technical difficulties as well as financial crises. Autonomy, integration, scalability and mobility are necessary features of integration software in CPD for harmonious collaboration in physically distributed and technology varied application systems.

H. Sun, T. Xiao and S. Tang offered a system independent, loosely coupled, and flexible integration method of heterogeneous information systems, which is named hierarchical federated integration [5].

In that method, *System federation* defines the environment of the physical aspects, and *application federation*s define that of logical ones. The *cooperative individual*s of a system federation are projected from real systems which are intended to collaborate together, and the Meta model of cooperative individuals that will participate in the application federation is also defined in system application. The collaboration in system federation is relatively simple and monotonous, only publishing sharable resources and *candidate application federate*s. After the application context is defined, these candidates can sponsor or join in an *application federation* to be a real *application federate*.

Before collaboration, a domain of interest related to resources sharing must first be established. When cooperative individuals project to a domain of interest, the projections from the same physical node form a *System Federate*. That is to say, one system federate can contain more than one projection of cooperative individuals. When system federates publish their resources, the static resources go into a sharable resource pool and the reactive resources become candidate application federates.

When a subsystem wants to establish a collaborative task, first, it will look up the candidate application federate pool After selection of the appropriate candidate application federates, a message of invitation will be sent to the projection owner of the involved candidate application federates. If these owners agree to their responsibilities during the coming collaboration, they will join this established *application federation* one by one and perform the collaboration as the agreement says. Or, the collaboration sponsor will seek other alternative participants in the candidate application federate pool.

B. Ontology fusion

In a CPD environment, there are always several subsystems in the same environment with independent design goals. And these subsystems may follow different design or management rules according to their professional fields [8].In HLA-based CPD, construction of a FOM needs multidisciplinary professional knowledge and technologies [6]. It is always time consuming and expensive.

H. Sun, W. Fan, W. Shen and T. Xiao presented an ontology fusion approach aiming to establish a mutual understanding in HLA-based distributed heterogeneous CPD systems [7]. The proposed approach has three steps: ontology mapping, ontology alignment, and ontology merging. Ontology mapping employs a top-down mechanism to explore all bridge relations between two terms from different ontologies on the basis of bridge axioms and deduction rules. Ontology alignment adopts a bottom-up mechanism to discover implicit bridge relations between two terms from different domain ontologies on the basis of equivalent inference. Ontology merging generates a new collaboration ontology from the discovered equivalent bridge relations.

C. RTI

RTI is a service program that realizes all service procedures in an interface specification of HLA and provides a series of interoperation functions among federates. In the implementation part of ontology maintenance, modification of RTI is of great importance. In this paper, the implementation part takes full advantage of TH_RTI, a RTI version developed by National CIMS ERC, Tsinghua University.

As Figure 1 shows, TH_RTI includes three main parts: LibRti is an interface library; RTIServer is a global process and it is server-end software of RTI system; RTIAmb is a local process, which performs as the RTI Ambassador.

LibRTI: A C++ library for developers, which provides a series of services mentioned in a HLA interface specification.

A federate invokes RTI services to communicate with RTIAmb according to libRTI via Windows TCP Socket. The

HLA interface specifies libRTI services for federates and the collaboration responsibilities of federates. Within libRTI, class RTI::RTIAmbassador encapsulates the services provided by RTI. And the service requirements from federate to RTI are all realized by RTI::RTIAmbassador invocations. Class RTI::FederateAmbassador is an abstract class, and it defines the necessary RTI callback functions of federates.

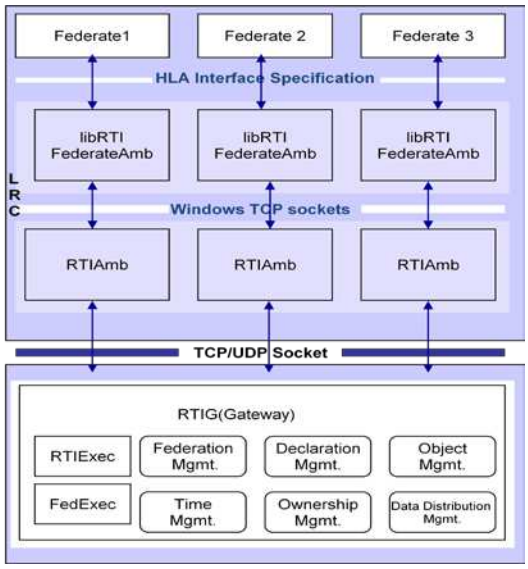

Figure 1. Software architecture of TH_RTI

RTIAmb Process (RTIA): a local process, which is in charge of Socket communication between federates and the RTIServer.

When a new federate joins a federation, the RTIAmb process will automatically start in the background and monitor the requirements from federates or RTIServer.

RTISevrer(RTIG)

RTIServer has two functions. One is communication management for RTIAmb processes, and its importance lies in that the communication among federates is fulfilled by RTIAmb processes communication. Another one is gateway management. When looking at RTIServer as a gateway, the communication routes can be deemed as a star topology, which can greatly reduce network workload.

Two processes run at RTIServer: RTI executive process RtiExec and federation executive process FedExec. Process RtiExec is a global process for all federations, which controls the creation and destruction of federation executions. Process FedExec performs federate joining to or resigning from a given federation. Within a given federation execution, FedExec is a global process, but there may exist several FedExec processes to arrange a federate changing in different federation executions, respectively.

D. Ontology maintenance

Ontology once confined to the world of philosophy has recently been moved to the regime of computer science. One of the most challenging aspects of using ontology is to keep it consistent, up-to-date and synchronized [9] with other such ontologies developed by similar developers and designers. This research area is called ontology maintenance.

However, different researchers define this term differently. Rafi, et al. [10] believe ontology maintenance is a broader term that encompasses in it a large number of complex activities like: combining, merging, integrating, aligning, mapping, articulation, translating, transforming, version and versioning. They investigated a Multi-Agent Based approach towards support for fully automatic ontology maintenance. Luczak-Rosch [11] deems ontology maintenance is an ontology engineering problem related to ontology evolution. He defined an agile ontology maintenance methodology because it is focused on continuously evolving ontologies in an application-dependent context. In this paper, ontology maintenance is a term which is related to the dynamic adjustment of collaboration ontology in the hierarchical federated integration environment when federates join in or resign from a given federation. When new federates join an existing federation, it may bring new collaboration requirements in the form of new collaboration concepts. In this circumstance, ontology maintenance can be deemed as an ontology merging problem, which is also part of ontology integration. However, when existing federates resign from a given federation, some collaboration concepts may not be useful. There is a need for separating these kinds of concepts from collaboration ontology. If these redundant concepts are ignored, the efficiency of a given federation execution cannot be guaranteed after running a long time. .

Because HLA-based CPD always involves multiple disciplinary domains, the terms which they use are often very different. It is very difficult to find equivalent relations by literature similarities. And before collaboration starts, there is no instance of collaboration concepts in the whole system. Instances-based merging is not applicable here. Most of the well known ontology integration tools cannot satisfy these requirements, as some of them are based on literal-based similarity computing methods (OntoMerge[12], PROMPT[13], ONION, Anchor-PROMPT). Some of them are also too simple, and weak in their description abilities (OntoMap[14]) and some are instances-based merging (GLUE[15]).

III. ONTOLOGY MAINTENANCE

The objective of this paper is to develop a novel ontology maintenance algorithm which can be used in ontology-based dynamic CPD processes. To address this issue, related formal definitions are given first.

A. Definitions

Because the algebraic system defined on the concept set of CPD and the partial order relations of these concepts have the same upper bound and lower bound, it can be deemed as a concept lattice [16]. This paper formally defines related concepts as follows:

Definition 1: CPD ontology

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

CPD ontology O is defined as a seven tuple. C denotes a collaboration concept set of CPD. H_C defines a set of partial orders on concept set C, which gives 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 the collaborative product and whose sink is binary fragments. R_C denotes a set of non-inherit partial order relations on concept set C, corresponding 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, which give a series inheritable instances of R_C. R_M denotes a set of partial order relations under M, which describe the relations among elements in a meta ontology set, and are also the basis for collaborative product ontology reasoning. A defines a set of axioms among an ontology concept set and meta ontology relation set, which provide the major premises of CPD ontology reasoning.

Definition 2: Ontology-based CPD ontology maintenance (+)

$$\text{maintain}_+ ::= O \times O_{\text{fuse}(\text{SET}_O)} \rightharpoonup O_{\text{fuse}(\text{SET}_O+O)},$$
$$(\forall e \in O_{\text{fuse}(\text{SET}_O+O)} \wedge e \in O \rightarrow \exists f.\ e \Leftrightarrow f, f \in O_i, O_i \subset \text{SET}_O:$$
$$\text{maintain}_+(O, O_{\text{fuse}(\text{SET}_O)}) = O_{\text{fuse}(\text{SET}_O+O)})$$

It is a partial order mapping from a Cartesian product to an updated collaboration ontology. The Cartesian product is composed of a new ontology and the fusion result of a given ontology set SET_O. The term $e \in E$ may be concept, or relation. To any term e in the output ontology $O_{\text{fuse}(\text{SET}_O+O)}$, if e also belongs to the new Ontology O it can find at least one corresponding equivalent term in an ontology of a prepared ontology set SET_O.

Ontology maintenance (+) ($\text{maintain}_+(O, O_{\text{fuse}(\text{SET}_O)})$) and ontology fusion ($\text{fuse}(\text{SET}_O + O)$) can reach the same result collaboration ontology, but they have a basic difference. Ontology maintenance (+) adds new equivalent bridge relations to an existing collaboration ontology, but ontology fusion creates a new collaboration ontology from scratch.

Definition 3: Ontology-based CPD ontology maintenance (-)

$$\text{maintain}_- ::= O \times O_{\text{fuse}(\text{SET}_O)} \rightharpoonup O_{\text{fuse}(\text{SET}_O-O)},$$
$$((\forall e \in O \rightarrow \nexists f.\ e \Leftrightarrow f, f \in O_{\text{fuse}(\text{SET}_O-O)}) \vee$$
$$(\forall e \in O, \exists f.\ e \Leftrightarrow f, f \in O_{\text{fuse}(\text{SET}_O-O)} \rightarrow \exists f', f \Leftrightarrow f', f' \in O_{\text{fuse}(\text{SET}_O-O)}):$$
$$\text{maintain}_-(O, O_{\text{fuse}(\text{SET}_O)}) = O_{\text{fuse}(\text{SET}_O-O)})$$

It is a partial order mapping from a Cartesian product to an updated collaboration ontology. The Cartesian product is composed of the collaboration ontology and one ontology in a given ontology set SET_O. The term $e \in E$ may be concept or relation. To any term e in the selected ontology from ontology set SET_O, if it can find an equivalent term f in the result collaboration ontology $O_{\text{fuse}(\text{SET}_O-O)}$, then there must exist one term f' in the resulting collaboration ontology which equals to term f. The output collaboration ontology of ontology maintenance ($\text{maintain}_-(O, O_{\text{fuse}(\text{SET}_O)})$) is the same as the one of ontology fusion ($\text{fuse}(\text{SET}_O - O)$).

The equivalent and mutual exclusive graph is an enhanced graph G' based on equivalent graph G with the exclusive relations added (no longer a DAG). The mutually exclusive relation between concepts (C_i, C_j) in CPD ontology is a symmetrical relation, and any instance of C_i and its sub concepts will not be the instance of C_j and its sub concepts. The equivalent and mutual exclusive graph denotes these relations by $\leftrightarrow$ between C_i and C_j. One mutual exclusive relation may contain another one. In that case, two ancestor concepts mutual exclusion implies descendant concepts mutual exclusion. This mutual exclusive relation is named as a trivial mutual exclusive relation.

B. Algorithms

When adopting ontology-based CPD, new collaboration partners may emerge from time to time. At the same time, existing partners also have the possibility to secede from the collaboration ally. Ontology maintenance technology is most useful under this circumstance. It can find the relations of given concepts and those in existing ontology so as to reuse data and interoperate among various applications [17]. Ontology maintenance technology enhances flexibility and adaptability of ontology-based product development systems.

Ontology maintenance (+) algorithm

The input of an ontology maintenance(+) algorithm is domain ontology set $\{O^m\}$ (domain ontologies of existing participants), ontology O^* (domain ontology of adding participant), bridge equivalent concepts pair list EC, bridge mutual exclusive concepts pair list IC, domain axiom set DA and collaboration ontology FON. The output is FON', the updated collaboration ontology.

The ontology maintenance (+) algorithm can be divided into three stages: mapping, alignment and merging. In the mapping stage, the equivalent bridge relations and the mutual exclusive relations between concept in O^* and the ones in $\{O^m\}$ are found. In the alignment stage, all potential equivalent concept pairs between concept in O^* and the ones in $\{O^m\}$ are compared. In merging stage, O^* is added into the collaboration ontology FON according to its equivalent structure graph G^*.

Algorithm 1. Ontology_maintenance + (O^*, $\{O^m\}$, EC, IC, DA, FON)

Input: O^* ontology of the new participant
$\{O^m\}$ ontology set of collaborative participants
EC bridge equivalent concept pair list
IC bridge mutual exclusive concept pair list
DA domain axiom set
O_{FON} collaboration ontology

Output: O_{FON}

1 foreach (O_i) **in** $\{O^m\}$ **do**
2 EC ← $\{(C_k^{(O_i)}, C_l^{(O^*)})\}$
// find domain equivalent bridge axiom from DA
3 IC ← ∅

/*Extract equivalent (mutual exclusive) graphs */
4 $\widetilde{G}_i$ ← Equivalent(Mutual_Exclusive)_Relation_Travel(O_i)
5 $\widetilde{G}_*$ ← Equivalent(Mutual_Exclusive)_Relation_Travel(O^*)

/*Simplify equivalent (mutual exclusive) graphs by deleting trivial equivalent (mutual exclusive) relations (Thing, data type equivalent, trivial mutual exclusive relations and independent concept nodes)*/

6 $G_i \leftarrow \text{Simplify}(\widetilde{G}_i)$

7 $G_* \leftarrow \text{Simplify}(\widetilde{G}_*)$

/*According to $\{(C_k^{(O_i)}, C_l^{(O^*)})\}$, mark $\{C_k^{(O_i)}\}$ of G_i, and mark $\{C_l^{(O^*)}\}$ in G_*, iteratively delete un-marked concepts of zero in-degree and their m-out-arc */

8 $G'_i \leftarrow \text{Bridge\_simplify}(G_i)$

9 $G'_* \leftarrow \text{Bridge\_simplify}(G_*)$

/*Inferring bridge equivalent relations, only equivalent graphs of G'_i and G'_*, $G'^{=}_i$ and $G'^{=}_*$, are used here and all the discussions below are all based upon structure equivalent relations*/

10 **foreach** unmarked concept C_i **in** $G'^{=}_i$ **do**

11 **if**(∃ one-one bridge equivalent relation between two ancestor concept sets of C_i and C_j, any concept of G'_* in structure equivalent relations) **then**

12 EC← EC + (C_i, C_j)
// duplicate elements eliminated

13 **elseif**(∃ one-one bridge equivalent relation between two ancestor concept sets of C_i and C_j, any concept of G'_*. The attributes, constraints, partial order relations between concept and its attributes are also equal, and the concepts in constraint paths also have corresponding equivalent bridge concepts in mixed equivalent relations.) **then**

14 EC← EC + (C_i, C_j)

15 **end if**

16 **end**

/* Extract structure graphs.*/

17 $G_i^S \leftarrow \text{Travel}(O_i)$

18 $G_*^S \leftarrow \text{Travel}(O^*)$

19 $\{(SC^{(O_i)}, SC^{(O^*)})\} \leftarrow {O^C}_i \times {O^C}_*$
// Cartesian product of concept set in O_i and O^*

/* According to mutual exclusive bridge relations simplify $\{(SC^{(O_i)}, SC^{(O^*)})\}$ */

20 **while**(∃$\{(IC_m^{(O_I)}, IC_m^{(O_i)})\}$ in IC and $IC_m^{(O_I)} \equiv SC_k^{(O_I)}$) **do**
//$\{IC_m^{(O_I)}|G_{O_I}\}^{\top}$ is the concept set which includes concept $IC_m^{(O_I)}$ and all its ancestors //according to the structure graph G_{O_I} of O_I

21 $\{(SC^{(O_i)}, SC^{(O^*)})\} \leftarrow \{(SC^{(O_i)}, SC^{(O^*)})\} - \{(IC_m^{(O_I)} \times \{IC_m^{(O_i)}\}_{\perp})\}$

22 $\{(SC^{(O_i)}, SC^{(O^*)})\} \leftarrow \{(SC^{(O_i)}, SC^{(O^*)})\} - \{(IC_m^{(O_I)} \times \{IC_m^{(O_i)}\}^{\top})\}$

23 **end while**

24 $\{(R_I C^{(O_i)}, R_I C^{(O^*)})\} \leftarrow \{(SC^{(O_i)}, SC^{(O^*)})\}$

/*According to equivalent bridge relations simplify $\{(R_I C^{(O_i)}, R_I C^{(O^*)})\}$ */

25 **while** (∃$\{(EC_m^{(O_I)}, EC_m^{(O_i)})\}$ in EC and $EC_m^{(O_I)} \equiv R_I C_k^{(O_I)}$) **do**

26 $\{(R_I C^{(O_i)}, R_I C^{(O^*)})\} \leftarrow \{(R_I C^{(O_i)}, R_I C^{(O^*)})\} - \{\{(EC_m^{(O_I)})\}^{\top} \times \{EC_m^{(O_i)}\}_{\perp}\}$

27 **end while**

28 $\{(R_E C^{(O_i)}, R_E C^{(O^*)})\} \leftarrow \{(R_I C^{(O_i)}, R_I C^{(O^*)})\}$

/*Inferring equivalent bridge relations.*/

29 $\{(R_M C^{(O_i)}, R_M C^{(O_j)})\} \leftarrow \{(R_E C^{(O_i)}, R_E C^{(O^*)})\}$

30 **foreach** $(R_E C_m^{(O_I)}, R_E C_n^{(O_i)})$ **in** $\{(R_E C^{(O_i)}, R_E C^{(O^*)})\}$ **do**

31 **if**(data type construction is different according to data type meta class definition of meta ontology) **then**
//the difference of data type construction include data type unit number inconsistency and //data type inheritable

32 $\{(R_M C^{(O_i)}, R_M C^{(O^*)})\} = \{(R_M C^{(O_i)}, R_M C^{(O^*)})\} - (R_E C_m^{(O_I)}, R_E C_n^{(O_i)})$

33 **end if**

34 **end**

35 $\{(R_U C^{(O_i)}, R_U C^{(O^*)})\} \leftarrow \text{Confirmed}(\{(R_M C^{(O_i)}, R_M C^{(O^*)})\})$
//confirmed by domain experts

36 **end**

37 $EC \leftarrow EC \cup \{(R_U C^{(O_i)}, R_U C^{(O^*)}), O_i \subseteq \{O^m\}\}$

/* simplify structure graph of O^*according to bridge equivalent concept pair list EC. */

38 $G_E^* \leftarrow \text{Equivalent\_travel}(EC, O^*)$

/*Extract structure graph of O_{FON}. */

39 $G_{FON}^S \leftarrow \text{Travel}(O_{FON}.)$

40 **let** $\bar{C}_t^{G_E^*}$= top node of G_E^*

41 **let** $(C^{(O_{FON})}, \bar{C}_t^{G_E^*}) \in EC$

42 **if**$(C^{(O_{FON})} \in G_{FON}^S)$ **then**
Add $\bar{C}_t^{G_E^*}$ into bridge equivalent concept chain of $C^{(O_{FON})}$

43 **else**

44 Add $\bar{C}_t^{G_E^*}$ into O_{FON} as a direct child of *its root*

45 Add $C^{(O_{FON})}$ into bridge equivalent concept chain of $\bar{C}_t^{G_E^*}$

46 **end if**

47 $\{C^{(O^*)}\} \leftarrow \text{child}(\bar{C}_t^{G_E^*}, G_E^*)$

48 **if**($\{C^{(O^*)}\}$! =**null**) **then**

49 **foreach**($C^{(O^*)}$ **in** $\{C^{(O^*)}\}$)

50 **let** $(C^{(O'_{FON})}, C^{(O^*)}) \in EC$

51 **if**$(C^{(O'_{FON})} \in G_{FON}^S)$ **then**

52 Add $C^{(O^*)}$ into bridge equivalent concept chain of $C^{(O'_{FON})}$

53 **else**

54 Add $C^{(O^*)}$ into O_{FON} as a direct child of *its direct parent* according to G_E^*

55 Add $C^{(O'_{FON})}$ into bridge equivalent concept chain of $C^{(O^*)}$

56 **end if**

57 $\{C^{(O^*)}\} = \{C^{(O^*)}\} + \text{child}(C^{(O^*)}, G_E^*)$

58 **end**

59 **end if**

60 **return** O_{FON}

In ontology maintenance (+) algorithm, line 1 to line 37 is a loop of every ontology in the domain ontology set $\{O^m\}$. This loop performs ontology mapping and alignment between new adding domain ontology O^* and existing domain ontologies. Line 2 to line 16 is the ontology mapping part. Line 2 searches domain equivalent bridge axioms from domain axiom set DA. Line 4 to line 9 gets the simplified equivalent (mutual exclusive) graphs of two ontologies. Note that the former equivalent (mutual exclusive) graphs of existing federates in the ontology fusion process are different from these, because new bridge equivalent relations may add to this collaboration. Line 10 to line 16 is another loop to infer new bridge equivalent relations. Line 17 to line 37 is the

ontology alignment part. Line 17 to line 19 gets all possible equivalent concept pairs SC. The loop from line 20 to line 23 simplifies SC by mutual exclusive bridge relations. The loop from line 25 to line 27 simplifies SC by equivalent bridge relations. The loop from line 30 to line 34 simplifies SC by datatype of concept attributes. Line 35 confirms simplified SC by domain experts. Line 38 to line 59 adds new concepts into collaboration ontology O_{FON}. The top concept of O^* (adding domain ontology) equivalent tree is first added to the proper position of collaboration ontology in line 42 to line 46. Line 49 to line 58 performs a breadth-first search of equivalent structure tree G_E^*, and adds every node to the proper position of collaboration ontology O_{FON}.

Ontology maintenance (-) algorithm

The input of the ontology maintenance (-) algorithm is ontology O^*(domain ontology of resigned participant), bridge equivalent concepts pair list EC and collaboration ontology FON. The output is FON', the updated collaboration ontology. The ontology maintenance (-) algorithm removes concepts $\{C\}$ in ontology O^* from collaboration ontology FON.

Algorithm 2. Ontology_maintenance - (O^*, EC, FON)

Input: O^* ontology of the new participant
EC bridge equivalent concept pair list
O_{FON} collaboration ontology
Output: O_{FON}

/*simplify structure graph of O^* according to bridge equivalent concept pair list EC*/

1 $G^* \leftarrow$ Equivalent_travel(EC, O^*)
2 $\{C^{(O^*)}\} \leftarrow$breadth_first_travel(G^*)
3 $\{C^{(O_{FON})}\} \leftarrow$breadth_first_travel($O_{FON}$)
4 **foreach**($C^{(O^*)}$ **in** $\{C^{(O^*)}\}$)
5 **do**
6 $C^{(O_{FON})}$=first($\{C^{(O_{FON})}\}$)
7 $\{C^{(O_{FON})}\} \leftarrow$remove_first($\{C^{(O_{FON})}\}$)
8 **while**($C^{(O_{FON})} != C^{(O^*)}$)

// more than 2 equivalent concepts
9 **if**($C^{(O_{FON})}$. bridge_equivalent_concept_chain.length > 2) **then**
10 Remove $C^{(O^*)}$ from bridge equivalent concept chain of $C^{(O_{FON})}$
11 **else**
12 Remove concept $C^{(O_{FON})}$ and its bridge equivalent concept chain from O_{FON}
13 Add children of $C^{(O_{FON})}$ to its direct parent in O_{FON}
14 **end if**
15 **end**
16 **return** O_{FON}

This algorithm just removes concepts of resigning federate ontology from their bridge equivalent chain in collaboration ontology. If only one concept left, the node in collaboration ontology also would also need to be removed, or else just removing the concept would be enough. Line 1 gets the equivalent structure graph G^* of resigning ontology according to the bridge equivalent concept pair EC. Line 2 and line 3 output the concepts in G^* and O_{FON} in a breadth-first travel manner. Line 4 to line 15 performs the remove. The loop from Line 5 to line 8 searches for a corresponding equivalent concept. Line 9 to line 14 performs the removal according to different situations.

These algorithms are defined on the equivalent structure graph-based knowledge representation and an attribute group comparison-based merging mechanism. Compared with the lightweight ontology integration method t is only based on structure and terms, the main advantage is that, when maintaining ontologies, the heuristic information, such as the equivalent structure graph and semantic equivalence of the attribute, is also taken into consideration. The efficiency and accuracy of ontology maintenance have been greatly improved.

IV. IMPLEMENTATION ISSUES

RTI is key software supporting HLA-based product development. Supporting from RTI is very important to semantics-based federated CPD environment. All the work mentioned below is on the bases of TH_RTI, a RTI version developed by National CIMS ERC, Tsinghua University.

A. The framwork of ontology-enabled RTI(ORTI)

As Figure 2 shows, the main framework of ORTI is similar with that of TH_RTI. The main parts are federate end and server end (RTI gateway). The federate part includes the ontology enabled RTI ambassador (ORTIA) and federates ambassador (ORTIFA). The server end is the ontology enabled RTI gateway (ORTIG). The communications on the network are performed by TH_RTI. Protégé enabled container is introduced to parse, operate, store and generate ontology instances.

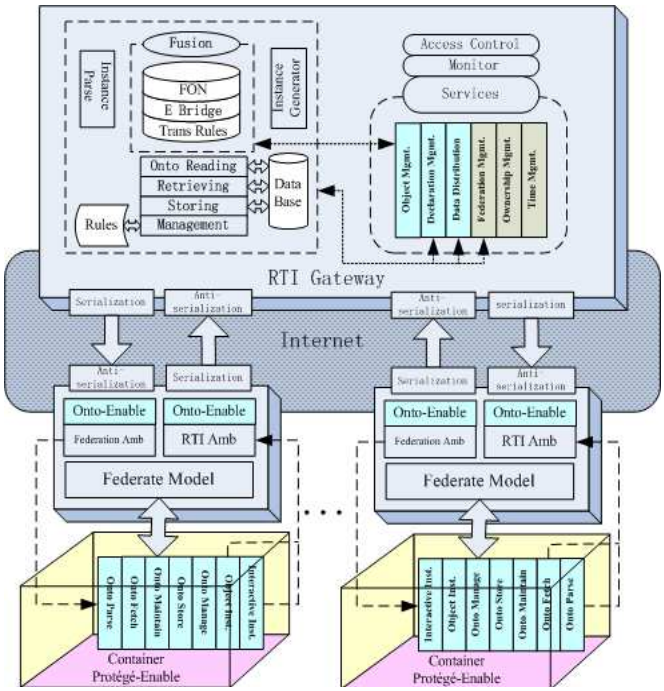

Figure 2 framework of ORTI

The federate end of ORTI is composed of 4 parts: federate model, ORTIA, ORTIFA and the ontology enabled container. The difference between ORTIA, ORTIFA and RTIA, RTIFA is that ontology enabled *Ambassador* classes do not pass

instances of object classes or interaction classes as invocation parameters, while they use instances of the corresponding concepts in the collaboration ontology. Serialization and anti-serialization functions are used for transmitting concept instance on the network.

Federate End

Theoretically, RTI is independent with federates. In implementation of ontology-enabled RTI the parameters of functions for communication between federates and ORTI involves instances of collaboration ontology concepts. If there is no transforming middleware, the collaboration interface of federates needs to make some change to comply with ontology related interactions. In ORTI, the Protégé enabled container is this kind of middleware, so federates can use ORTI in a TH_RTI compatible manner.

The container performs most ontology management functions within ontology-enabled federated collaborations. In the preparation stage, domain ontologies are well established and stored in the container. The main functions of the Protégé enabled container include ontology management, instance generation and instance parsing. Ontology management provides domain ontology manipulating functions as inquiry, read and modification to federates. Instance generation generates concept instances of objects and interactions according to the collaboration requirement based on the collaboration ontology. Instance parsing translates a received concept instance and output to a given federate in a TH_RTI compatible way.

The concept instance substitutes object class or interaction class instance in the ORTI system. After being generated by the ontology container, ORIA sends the concept instance to ORTIG by network socket. When ORTIG receives a new concept instance notification, it looks up the subscription list of this concept instance and distributes the new instance or reflects the attribute values to federates which have subscribed to the given concept or attribute. The concept instances from ORTIG to ORTIFA are first parsed by the ontology container and then reflected to federates in callback functions with TH_RTI compatible formats.

Because interactions of some management services, such as time management and ownership management, do not involve instance of ontology concepts, there is no change for this kind of management services.

RTI Server

The RTIServer end is the main operational part of RTI related functions. ORTIG inherits 6 management services of TH_RTI which are specified in the HLA standard. Ontology related management includes declaration management, object management, federation management and data distribution management. Besides these, ORTIG provides some other functions as a federation monitor (displays real-time interaction states of federates), access control, result analysis, and collaboration evaluation.

There is a central Protégé enabled container in ORTIG. Before collaboration starts, collaboration ontology O_{FON} and related information are stored in this container. The information includes equivalent structure graph and transformation rules between semantic equivalent concepts. This container also provides ontology inquiry, store and read functions of O_{FON}. The modification of O_{FON} can only be done by ontology maintenance algorithms after the collaboration starts.

Compared with an FOM file, ontology enabled RTI improves the dynamic performance of the HLA-based collaborations. The domain ontology represents collaboration requirements of federates. When a given federate is going to join in another federation, it does not necessarily need to rewrite the code of collaboration interfaces; a few changes in the domain ontology may already be enough. The transformation rules for semantic equivalent concepts also reduce the workload of the collaboration interface's modification. At the same time, because the container is introduced, there can be several virtual federates corresponding to one physic federate. The ontology enabled RTI can support intersecting federation executions.

B. ORTI services

ORTI's main changes to services of TH_RTI are declaration management, object management and data distribution management. This section focuses on the changes of these management services and the implementation of ORTIG.

Declaration management. Declaration management provides a data filter at the class level and declares inter-federate interactions at the logical level. It includes publication, subscription and supporting control functions.

To every subscribing or publishing object classes:

- *getObjectClassHandle* service is used to get its unique handle, which is corresponding to a given concept in collaboration ontology O_{FON}.
- Static function create of ORTI class AttributeHandleSetFactory is used to create the attribute handle set AttributeHandleSet(AHS).
- Function getAttributeHandle is used to get the attribute handle from collaboration ontology O_{FON}, and adds this handle to AttributeHandleSet.
- publishObjectClass services is used to publish object classes and their attributes.
- Federate invokes subscribeObjectCalssAttributes to subscribe its attributes. After ORTI invokes the callback function startRegistrationForObjectClass, the publisher can register and update instances of the given object class.
- When there is no valid subscriber of a given object class in the collaboration, ORTI informs the publisher to invoke the callback function of stopRegistrationForObjectClass to stop the publisher

form registering and updating instances of given object class.
- The publisher federate uses unpublishObjectClass to declare the end of publishing of a given object class.
- Empty and delete attribute handle set.

Publication/subscription routines of interaction classes are similar with that mentioned above, and the instances of interaction classes are also used in management objects in MOM (Management Object Model).

The class inheritance relations are implied in the concept inheritance relations of collaboration ontology O_{FON}.

Object management. Object management realizes information transmitting among federates when the collaboration runs, which includes register and discovery of object class instances, updating and reflecting of attribute values, as well as receiving and sending of interaction class instances.

After federate uses the function *publishObjectClass* publish object classes, *registerObjectInstance* is used to generate an object instance of a given object class in federation execution. The instance is named by the publisher. The instance generated in ORTI federation execution is a concept instance of collaboration ontology O_{FON}, and only the instance owner can change its attribute value in the federation execution. When the collaboration does not need the instance any more, the federate invokes *deleteObjectInstance* to delete the instance from a given federation execution.

The workflows of object class instance updating/ reflecting are almost the same as those in TH_RTI. But the parameters of service *updateAttributeValues* and *reflectAttributeValues* need to change. The attribute handle set *AttributeHandleValuePairSet* is changed to concept instances *ObjectClassInstance,* corresponding to object classes. Modification of the concept instance can be sent to ORTI by updating/reflecting functions, and federates get their attribute values by instance parsing.

The processes of sending/receiving interaction class instances are very similar to those of object classes. The creation process uses *ORTI::InstanceValueSetFactory:: create* to create concept instances *InteractionClassInstance*. Please note that the concept instance corresponding to interaction classes has a transient existing; after the subscriber receives it, ORTI will automatically delete it. However, the handle for this concept instance is still unique in a federation execution.

Data distribution management

Ontology fusion is an important preparation for ontology-based federated CPD. Because the objective of ontology fusion is to find a union of bridge equivalent intersections in a domain ontology set, only bridge equivalent relations can be found in resulting collaboration ontology. Thus, most of the data distribution functions have been satisfied in the ontology fusion process. Most of TH_RTI data distribution functions are enough for ontology-based federated CPD, except ontology concepts need an additional attribute of routing space information, which is useful for intersection determination of a concept instance to the subscriber's requirements.

ORTIG service realization. Compared with RTIG, ORTIG uses an ontology container to store and manipulate collaboration ontology O_{FON}. Its equivalent structure graph and equivalent concept chain implies publication/subscription relations between ontology concepts from different federates and indicates a series of equivalent transformation entity sets. There is great potential to use artificial intelligent tools to automatically transform one concept to another concept in the same equivalent transformation entity set.

Besides functions that TH_RTI already has, some new/updated functions add to ORTIG in order to support ontology-based federated CPD (Table 1).

Table 1. Functions of Protégé enabled container

Service Type	Function
Basic communication	Instance generation Instance parsing
Declaration Management	Acquire/return concept handle Acquire/return instance handle Acquire/return attribute handle Activate object concept Inactivate object concept Activate attribute concept Inactivate attribute concept
Object Management	Object/interactive concept instance generation Update concept instance attribute Reflect concept instance attribute Destroy concept instance
Data Distribution Management	Equivalent concepts transformation

V. DISCUSSION AND FUTURE PLANS

Ontology maintenance is very similar to the process of ontology fusion. They all work at collaboration ontology. They have similar processes.

However, there are still some remarkable differences. Although the collaboration ontology definition forms abelian monoids which make ontology fusion a linear complexity problem, the objective of ontology fusion is $n \times n$ relations among an ontology set with length n. Ontology maintenance just occurs between one ontology and an ontology set with length n, and its computing complexity is inherent $O(n)$. Ontology fusion is an important preparation for ontology-based federated CPD, while ontology maintenance happens in the collaboration execution stage. In other words, ontology fusion is sort of "static" and ontology maintenance is more "dynamic". Ontology fusion makes collaboration bigger and bigger,

and ontology maintenance sometimes removes the concept from collaboration ontology.

Let n denote the average concept number in one ontology; m is the length of the ontology set, and l represents the length of the DA (Domain Axiom set). The complexities of ontology fusion are $O(m \times \max\{n \cdot l, n^2\})$, the complexity of ontology maintenance (+) is $O(m \cdot n^2)$ which is almost as much as that of ontology fusion, and the complexity of ontology maintenance (-) is $O(n \cdot \log_2 n)$. Hence, although collaboration ontology can get by using ontology maintenance (+) $n-1$ times, it is not recommended.

In collaboration ontology, all concepts in one equivalent concept chain are supposed to be distinguishable. That is to say, all concepts in collaboration ontology have their federate identification. Two concepts from different federates, even use the same name, but are different in the collaboration ontology. This helps to find corresponding domain ontologies and furthermore is easy to track and maintain the collaboration ontology.

In HLA-based CPD, the most difficult issue is to not establish a collaborative system, but to adaptively adjust interface codes of existing systems and to negotiate among multidisciplinary domains. This paper proposes a novel method to dynamically maintain the collaboration ontology when federates dynamically join in or resign from a given federation. The main part of this method includes two algorithms: ontology maintenance (+) and ontology maintenance (-).

Although, from the view of complexity analysis, this approach may not be the best choice, it still enjoys several sound advantages which are more suitable for ontology-based federated CPD:

- This method is built on firm theoretical foundations and formalization definitions.
- It enables a dynamic adjustment of collaboration ontology, which can keep the collaboration going smoothly and correctly.
- It avoids federates from rewriting interface codes, stopping the collaboration execution to recompile these codes.
- Different from most other ontology integration tools using literature distance, this method employs heavy-weighted ontology to perform ontology maintenance functions. Axioms, bridge axioms, equality rules and attribute set equality conditions are all taken into consideration.
- Since ontology is used in this method, the reuse of resources, flexibility and expandability of existing systems are greatly enhanced.

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. 59 (2008), 855–862.

[2] W. Fan, W. Wang and T. Xiao, Multidisciplinary Collaboration Simulation Optimization Platform for complex product design, Proceedings of the 2nd International Conference on Pervasive Computing and Applications, 2007 (ICPCA 2007). Birmingham, UK, 2007,174 -178.

[3] H. Zhang, H. Wang and D. Chen, Integrating web services technology to HLA-based multidisciplinary collaborative simulation system for complex product development, 12th International Conference on Computer Supported Cooperative Work in Design, 2008, Xi'an, China, 2008, 420-426.

[4] K. L. Morse, M. Lightner, R. Little, B. Lutz and R. Scrudder, Enabling Simulation Interoperability, Computer, The Institute of Electrical and Engineers, Inc. New York, 2006, 115-117.

[5] H. Sun, T. Xiao and 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,vol. 7, 535–539.

[6] 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.

[7] H. Sun, W. Fan, W. Shen and T. Xiao, Ontology Fusion in HLA-based Collaborative Product Development, Proceedings of 2010 IEEE International Conference on Systems, Man, and Cybernetics: SMC 2010. Istanbul, Turkey, 2010, 2526-2532.

[8] 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. 18(8) (2010), 1048-1068.

[9] S.C. Flavio and J. Agusti-Cullell, Knowledge Coordination, WILEY, USA, July 2003.

[10] M. Rafi, H. Qureshi and H. Khatoon, Ontology Maintenance via Multi-Agents, 2009 Fifth International Joint Conference on INC, IMS and IDC, Seoul, Korea, 2009, 955-959.

[11] M. Luczak-Rosch, Towards Agile Ontology Maintenance, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, 2009, 965-972.

[12] D. Dou, D. McDermott, and P. Qi, Ontology Translation on the Semantic Web, Journal on data semantics II, 3360 (2005) , 35-57.

[13] N. F. Noy and M. A. Musen, The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping, International Journal of Human-Computer Studies, 59 (6)(2003), 983-1024.

[14] H. Schnurr and J. Angele, Do not use this gear with a switching lever!Automotive industry experience with semantic guides, 4th International semantic web conference, Galway, IRLANDE, 2005, 3729 (2005) 1029-1040.

[15] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, Learning to map between ontologies on the semantic web, Proceedings of the 11th international conference on World Wide Web, Honolulu, Hawaii, USA, 2002, 662-673.

[16] K. Qu, J. Liang, J. Wang and Z. Shi, The algebraic properties of Concept Lattice, Journal of Systems Science and Information, 2(2) (2004), 271-277.

[17] J. Yu, and Y. Dang, Review on Ontology Integration, Computer Science, China, 35 (7)(2008), 9-14.