N° d'ordre: 4673

THÈSE

PRÉSENTÉE A

L'UNIVERSITÉ BORDEAUX 1

ÉCOLE DOCTORALE DES SCIENCES PHYSIQUES ET DE L'INGENIEUR

Par Zhiying, TU

POUR OBTENIR LE GRADE DE
DOCTEUR

SPÉCIALITÉ: PRODUCTIQUE

Federated Approach for Enterprise Interoperability: A Reversible Model driven and HLA based methodology

Soutenue le 20 décembre 2012

Devant la commission d'examen formée de MM.:

David CHEN	Professeur, Université Bordeaux 1	Directeur
Gregory ZACHAREWICZ	Maître de conférences, Université Bordeaux 1	Co-directeur
Agostino BRUZZONE	Professeur, University of Genoa, Italy	Rapporteur
Ricardo JARDIM-GONCALVES	Professeur, New University of Lisbon, Portugal	Rapporteur
Dechen ZHAN	Professeur, Harbin Insititute of Technology, China	Examinateur
Yves DUCQ	Professeur, Université Bordeaux 1	Examinateur

Acknowledgements

The past three years are the unforgettable time in my life. I started my doctorate research, gained lots of invaluable advices from professors and senior researchers, met many wonderful friends and colleagues, and moreover, I had my first child during this time. Meanwhile, these three years are also a difficult time for me. However, I was fortunate enough to met lots of warm hearted professors and friends, and had a supportive family. They always helped me out, when I got confused and felt helpless. Accordingly, I would like to take a brief moment to specifically thank those people who have been a consistent presence through my time as a Ph.D. student.

Firstly, I have to thank the academic and support staff at the IMS/LAPS of University of Bordeaux 1. In particular, I must sincerely thank my supervisors, Professor David Chen and Gregory Zacharewicz. Thank you for introducing me to the “enterprise interoperability” world and “modeling and simulation” world. Thank you for pointing out the right way for me every time when I got confused in the research. Thank you for accepting to my idea and complementing it. Thank you for recommending me some interesting academic activities that broaden my horizon. I would also like to mention my gratitude to Professor Guy Doumeingts, Jean-Paul Bourrieres, Yves Ducq, Bruno Vallespir, Marc zolghadri, Alix Thecle, and Julien Francois. Thank you for your advices about how to be a good researcher. In addition, I would also like to thank Madam Isabelle Zolghadri Grignon and Valerie ABEL for help me out from the complex administrative problems.

Secondly, I would like to thank my dear colleagues, Jia Zhenzhen, Zhang Xin, Song Fuqi, Mounir BADJA, Guillaume Vicien, and Wael TOUZI. You are not only my colleagues, but also my dear friends. Thank Jia Zhenzhen for helping me out every time when I was short of money. Thank Zhang Xin, and Song Fuqi for willing to listen to my often incoherent ramblings or provide a welcome distraction. Thank Mounir BADJA, Guillaume Vicien, and Wael TOUZI for teaching me French, and helping me out in the troubles in particular with French. Mounir, thank you for always be with me at the very first time when I was in trouble. Guillaume, thank you for the first congratulation when I received best paper awards, let’s always remember the story happened in Coventry.

Thirdly, I would like to thank my friends, Yi Guangpeng, Zhang Zhenchuan, Zhang Songtao, Zhao Kai, Ma Xiaotian, Zhao Qiang, Li Chong, Hou Jinying, Lv Hao, Taochy Mario, and Armelle Polette. Thank you for hanging out with me every special day, such as Chinese New Year, Christmas day, and so on. Because of you, my dear friends, I felt less homesick in the past three year.

Finally, I must reserve me most heartfelt thanks to my family. Without your support, I cannot finish my doctorate study. Firstly, I must sincerely thank my wife, Lai Xiaoying. We were married at the first year of my time as a PhD. Then, she quitted her job and flew to Bordeaux to accompany me. The love support I received from her during this time was altogether above and beyond anything I could ever expect. One year after, we had our first child, Tu Zihan. After that, besides taking care of me, she also needed to take care of the baby. She had not complaint, but kept offering unconditional love without reservation. And then, I must thank my cute little baby, Zihan. You brought me an enjoyable life. Furthermore, I must mention my deepest gratitude to my parents, Tu Liuzhang and Lai Dongshui. Honestly, it is very difficult to adequately articulate the love I received from them. In the past twenty eight years, they gave me all their love. When, I decided to go abroad to study, they fully supported me, even though they were so reluctant to send their son to a strange place which is far far away from home. In a word, they provided everything I needed and without their support I would never have finished my doctorate study.

Résumé étendu en français

Approche fédérée pour l'interopérabilité d'entreprise: une méthodologie réversible, modèle entraînée et HLA basée

1. Contexte et problème

Au début des années 2000, la Commission Européenne a proposé d'identifier la problématique relative au développement des applications logicielles d'entreprise. Plusieurs projets de recherche ont contribué au développement de l'interopérabilité d'entreprises, « Enterprise Interoperability » (EI) qui se concentre principalement sur les architectures, les modèles, les méthodologies et les solutions opérationnelles pour l'EI. Sur la base des résultats de ces projets de recherche, de nombreuses solutions d'interopérabilité d'entreprise ont été testées et mises en œuvre pour aider les entreprises à se connecter et à collaborer avec leurs partenaires d'affaires dans une entreprise étendue et en réseau.

Aujourd'hui, le contexte économique très dynamique pousse les entreprises à fonctionner de plus en plus en réseau. Pour obtenir plus d'opportunités commerciales, et survivre face à la concurrence, les entreprises ne doivent pas uniquement tenir compte de leurs partenaires commerciaux en lien direct, mais aussi identifier des partenaires commerciaux potentiels en relation indirecte. Ce contexte nécessite des recherches dans le domaine de l'EI pour étudier tous les éléments coopératifs et compétitifs dans un environnement très dynamique et complexe. Ainsi, les solutions historiques de l'EI, telles que l'approche intégrée et l'approche unifiée identifiées dans le Cadre d'Interopérabilité des Entreprises proposé par les membres du réseau d'excellence INTEROP NoE puis utilisé par le laboratoire virtuel INTEROP V-Lab, ne permettent plus de satisfaire au contexte économique actuel et futur très versatile. Ceci signifie que la recherche de l'EI doit porter davantage sur la nature dynamique des besoins de l'entreprise future, à la fois pour l'entreprise unique ainsi que pour les écosystèmes. Dans ce contexte, le Cadre d'Interopérabilité des Entreprises a défini ce que devrait être l'Interopérabilité, celle-ci devrait être plus dynamique, cette nouvelle forme est nommée « approche fédérée ». Cette approche exige que l'interopérabilité soit établie « à la volée ». Cela signifie que l'ajustement des systèmes et le partage des modèles des divers

partenaires doivent s'effectueren définissant une ontologie ou un méta-modèles qui ne soient pas prédéfinis, mais formé par une négociation dynamique. Théoriquement, le développement d'une EI conforme à cette approche fédérée doit fournir un environnement d'interopérabilité très flexible et agile qui peut aider les entreprises à s'adapter au contexte économique dynamique et évolutif.Cette nouvelle voie est identifié dans une feuille de route pour l'interopérabilité des entreprise publié par la Commission Européenne qui avait estimé l'approche fédérée comme l'un des défis de recherche pour les années à venir (Charalabidis et al., 2008). Cependant, actuellement, mettre complètement en œuvre l'approche fédérée reste difficile compte tenu de l'avancée des travaux par approches sémantiques en informatique. Par rapport à l'ensemble des points évoqués, cette recherche de doctorat a identifié les défis suivants :

- Le marché dynamique et un contexte économique obligent l'entreprise à être capable d'interagir simultanément avec de multiples partenaires hétérogènes. Cela signifie que l'entreprise doit être en mesure d'ajuster et d'adapter son système en permanence sur différents canaux de communication.
- Pour s'adapter et répondre de façon dynamique aux partenaires potentiels d'interopérabilité, il est nécessaire d'effectuer « à la volée », les changements nécessaires pour se connecter aux systèmes des partenaires. Par conséquent, la capacité à restructurer rapidement les systèmes d'entreprise est un enjeu important pour développer l'approche fédérée de l'EI.
- Avant toute tentative de réingénierie, un autre défi est d'être capable de modéliser et collecter automatiquement des informations et des données pertinentes sur les systèmes et les applications existants déjà mis en œuvre dans l'entreprise et concernés par l'interopérabilité.
- Pour établir dynamiquement l'interopérabilité, il est nécessaire de réduire la complexité de l'IE. Comment utiliser les services d'interopérabilité comme des mécanismes «plug-and-play »permettant de translaterles principes d'interopérabilité du niveau de l'IEauxquels ils sont conçus vers leurs opérationnalisations (depuis les niveaux supérieurs tels que le business, vers les inférieurs tels que les applications techniques) est un autre défi à prendre en compte dans cette recherche.

2. Contribution de la thèse

Afin de vaincre les défis mentionnés plus haut, cette thèse a contribué à développer un cadre de modélisation réversible dirigé par les modèles et le standard de simulation distribuée HLA (High Level Architecture) et une méthodologie basée sur la mise en œuvre de l'approche

fédérée au titre du Cadre d'Interopérabilité des Entreprises. La contribution globale est résumée dans la figure 1.

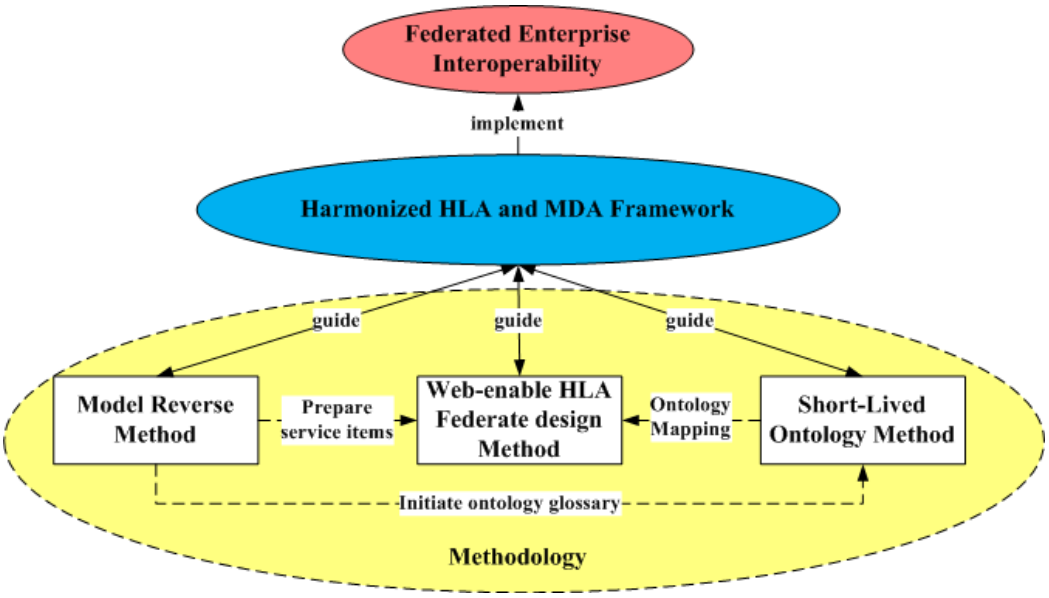

Figure 1.Contribution globale de cetterecherche

Firstly, a Harmonized and Reversible HLA based framework(as shown in figure2) has been elaborated. This framework has four primary concepts: (1)*Harmonized* means that this framework is synthetic, which consists of several techniques. As the framework in figure 2 shows, we propose a new five steps development life cycle which aligns MDA and HLA FEDEP. In addition, this framework uses web services to improve the flexibility and compatibilityof the HLA. (2) *Reversible* means that this framework uses model reverse engineering technique to discover part of the models from the legacy system. Model reverse engineering technique aims at avoiding rebuilding the complete legacy system for a new reuse. The objective is to accelerate the development and reduce the cost. (3) *HLA* means that this framework dedicates to the development of HLA based application. The RTI used in this approach is an open source RTI, poRTIco(poRTIco, 2009). In addition, as mentioned earlier in *Harmonized* part, Web Services will be used to improve the limitation of the traditional HLA. Thus, the HLA approach proposed in this thesis is based on the HLA evolved IEEE 1516^{TM}-2010 standard.

Tout d'abord, un cadre harmonisé et réversible basé sur HLA (comme le montre la figure 2) a été élaboré. Ce cadre comporte quatre concepts principaux : (1) *Harmonisée* signifie que ce cadre est synthétique, il se compose de plusieurs techniques. Comme le cadre de la figure 2 le

montre, nous proposons un nouveau cycle de vie (de développement) de cinq étapes qui aligne MDA et HLA FEDEP. En outre, ce cadre fait appel à des services Web afin d'améliorer la flexibilité et la compatibilité du système HLA. (2) R*éversibles* signifie que ce cadre utilise une technique de modélisationinverse (d'ingénierie inverse) pour découvrir une partie des modèles de l'ancien système. Cette technique d'ingénierie inverse de modélisation vise à éviter la reconstruction complète de l'ancien système pour une nouvelle réutilisation. L'objectif est d'accélérer le développement et réduire les coûts. (3) *HLA* signifie que ce cadre se consacre au développement d'applications basées sur HLA. Le RTI utilisé dans cette approche est un RTI en source ouverte, poRTIco (poRTIco, 2009). En outre, comme mentionné précédemment dans le cadre harmonisé, les services Web seront utilisés pour améliorer les limitations traditionnelles de HLA. Ainsi, l'approche HLA proposée dans cette thèse est basée sur la norme IEEE HLA évoluée1516TM-2010 standard.

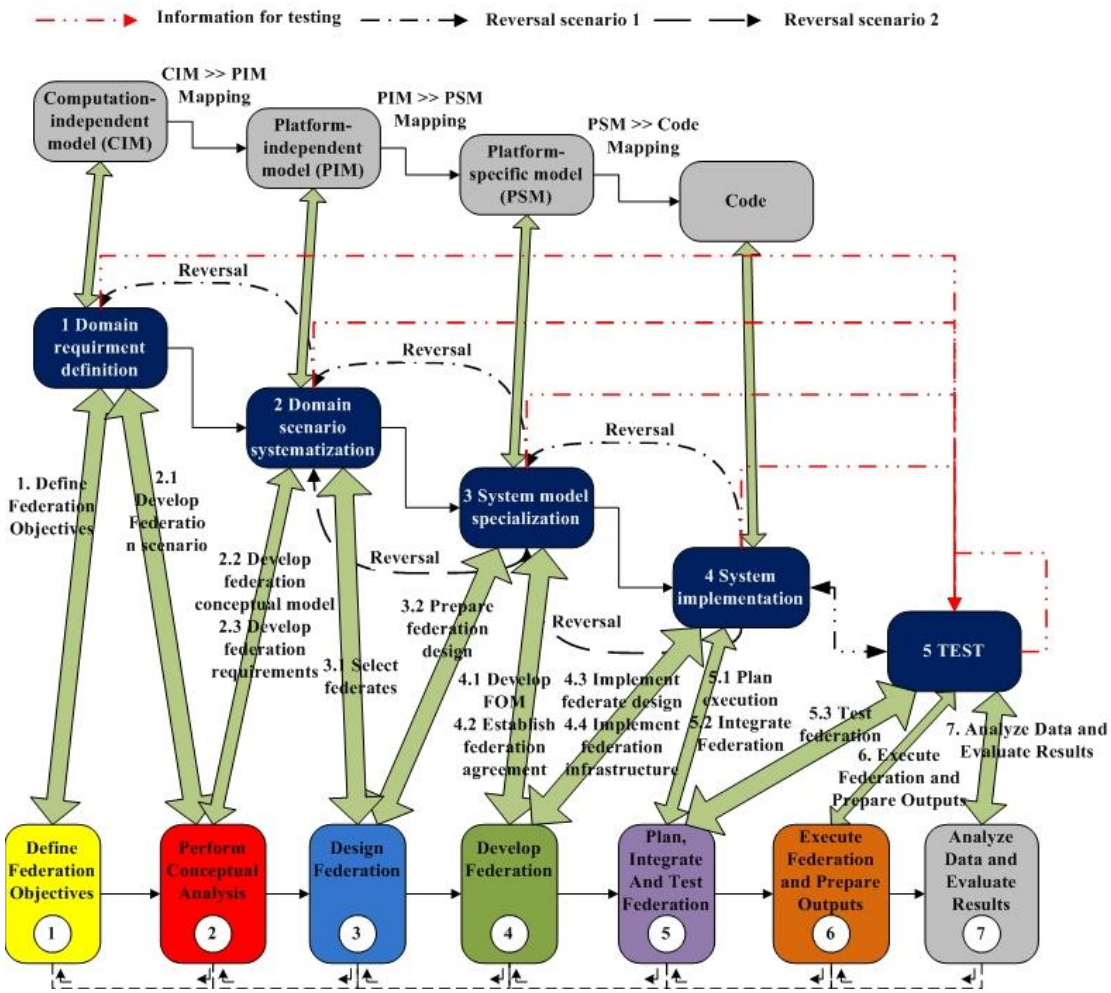

Figure 2.Cadre de développement harmonisé et réversible pour application basée sur HLA

Deuxièmement, pour étayer le cadre réversible et harmonisé basé sur HLA, une méthodologie

a été élaborée. Elle se compose de trois méthodes : la génération de modèles inverse par la découverte de modèles, une méthode de conception de fédérés HLA « web-enable », et une méthode basée sur l'utilisation d'ontologies éphémères. Cette méthodologie a proposé une nouvelle façon de soutenir le développement de l'approche fédérée de l'interopérabilité des entreprises en réutilisant certaines méthodes existantes, des architectures et des technologies, tels que MDA (Model Driven Architecture), le « Reverse Engineering » de modèles, HLA (High Level Architecture), les services Web, et les ontologies. Plus précisément, cette méthodologie (1) utilise MDA pour formaliser l'architecture du système et les relations entre les systèmes, (2) applique le reverse engineering de modèle pour réutiliser et harmoniser les différents systèmes/composants dans le nouveau système d'information de l'entreprise interopérable, (3) utilise HLA et les fonctionnalités des services Web comme assistance technique, et (4) utilise l'ontologie pour l'analyse de l'information. Après la définition de la méthodologie, architecture de Reverse Engineering dirigée par les modèles et HLA a été élaboré sur la base duquel un outil logiciel a été développé. L'utilisation de cet outil logiciel a été illustrée par une étude de cas illustratifs.

The Harmonized and Reversible HLA based framework defines the general guideline for the implementation of the three methods mentioned above. These three methods also complement each other in order to achieve the expected result of the federated approach of enterprise interoperability.

This framework and methodology have been implemented into a software tool called Model driven and HLA based Reverse Engineering Tool. The objective and functionality of this tool is identified by breaking down the name "Model driven and HLA based Reverse Engineering Tool":

Ce cadre et la méthodologie ont été mis en œuvre dans un outil logiciel appelé outil de Reverse Engineering dirigée par les modèles et HLA. L'objectif et la fonctionnalité de cet outil sont identifiés en décomposant le nom de « dirigée par les modèles et HLA outil de base Reverse Engineering »:

- *Reverse Engineering* signifie que cet outil peut acquérir des modèles de systèmes d'information d'entreprise en « rembobinant » les cycles de développement des systèmes existants.

- *Basé sur HLA* signifie que la plate-forme cible de cet outil est HLA. L'utilisateur final se connecte à la plate-forme grâce à un fédéré HLA de fédération.
- *Dirigé par les modèles (Model Driven)* signifie que cet outil doit résoudre les problèmes d'interopérabilité basées sur des modèles de systèmes existants, puis réformer les modèles du système interopérables, ce qui peut être converti au final en code exécutable en fonction de la plate-forme cible.

Ainsi, l'objectif (ou la sortie) de cet outil est une plate-forme interopérable de communication basée sur HLA. Les modules fonctionnels de cet outil sont (1) un module de construction, contenant une fonctionnalité de découverte de modèles et d'inversion de modèles, d'ajustement de modèle, et de définition de modèle cible et enfin de génération de code, et (2) un module d'exécution, contenant l'envoi/réception de message et leur gestion. L'architecture de cet outil est illustrée à la figure 3.

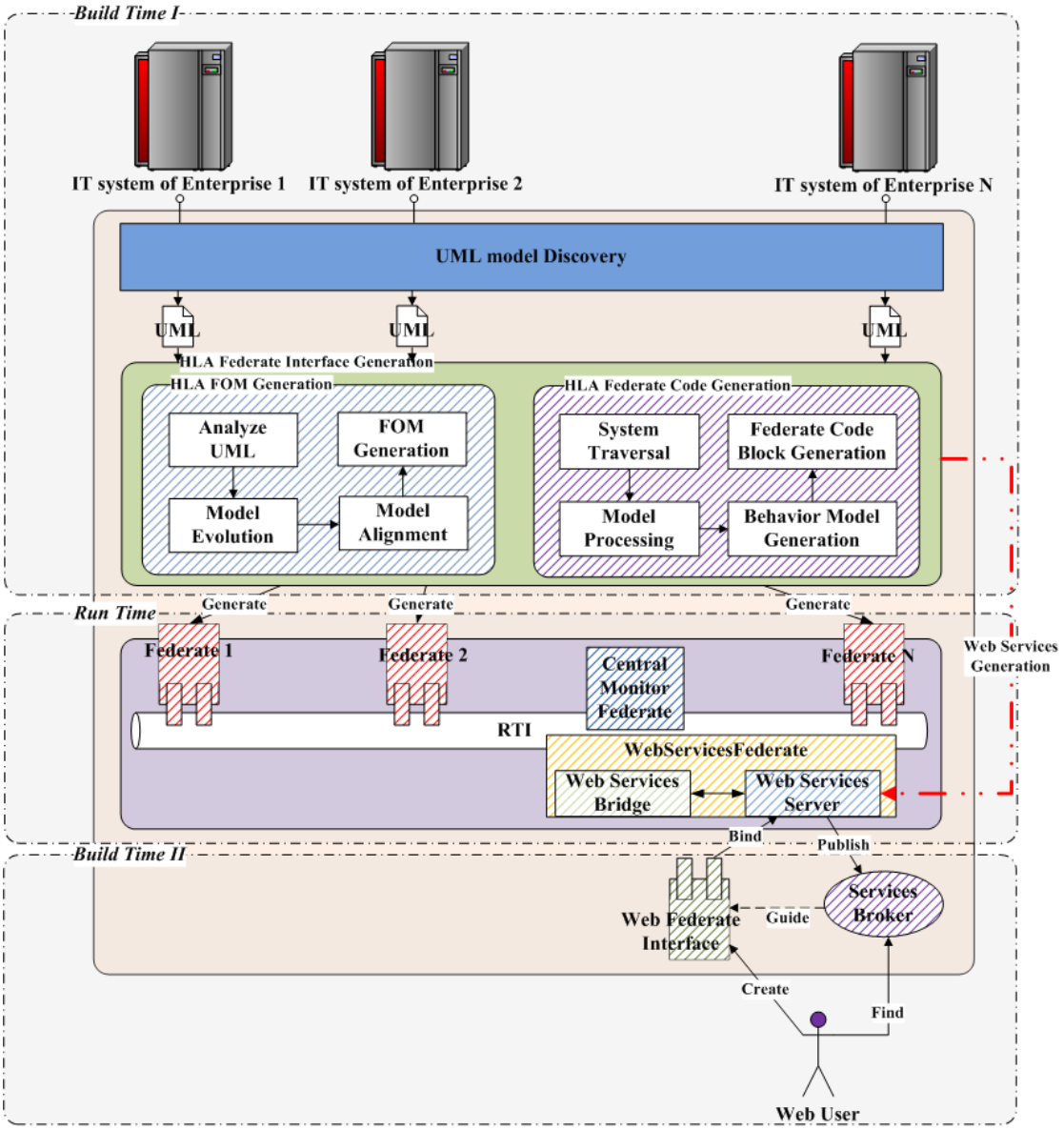

Figure 3. Architecture de l'outilde Reverse Engineering bas ésur les mod èles et HLA

- La partie «temps de construction » I (Build Time I) est la phase primaire. Il doit impl émenter la m éthode inverse de mod élisation et de d éveloppement de la F éd ération HLA basé sur le RTIpoRTIco. La méthode demodélisation inverse comprend l'inversion du modèle, l'ajustement du modèle, la définition d'un modèle cible et la génération de code. Il est chargé de préparer l'environnement de simulation pour l'interopérabilité de l'entreprise, qui concerne l'établissement d'une interopérabilité rapide et dynamique. Il est également responsable de la pr éparation des composants pour les services Web qui permettent le développement de fédérés et d'initier la contribution au glossaire d'ontologie Web des participants, qui visent à mettre en œuvre environnement aveccompatibilit é agile, et la gestion de l'environnement de collaboration.

- La partie « temps de construction » II (Build Time II) est une phase à la demande. Elle n'est réalisée que si un nouveau participant veut se joindre à partir du Web. La tâche de cette partie est de mettre en œuvre la compatibilité de l'environnement agile qui permet aux participants Web de rejoindre la collaboration comme un « plug-and-play ». Cette partie se compose d'une méthode de conception d'un fédéré HLA web-enable et d'une méthode de utilisant l'ontologie éphémère pour établir la communication. La méthode reposant sur les ontologies éphémères est partiellement mis en œuvre dans cette phase afin d'aider les participants à initier leur glossaire web à partir d'une ontologie locale.
- La partie « temps d'exécution » (Run Time) est la simulation, cette phase gère l'échange d'information en dynamique y compris l'envoi et la réception de message et leur gestion. Cela concerne l'échange d'informations transitoireet leurs analyses. Pendant ce temps, la production et la connexion d'un nouveaufédéré« Web-enable »peut arriver à tout moment en cours d'exécution.

Table of Content

List of Figures

List of Tables

General Introduction

1. Background and Problem

Since the beginning of 2000s, the European Commission has proposed to identify the problematic/approach relating to the development of enterprise software applications. Many research projects have contributed to Enterprise Interoperability (EI) development that mainly concentrates on EI architectures, models, methodologies, and operational solutions. Based on the results of these research projects, numerous enterprise interoperability solutions have been tested and implemented to help enterprises to connect and to collaborate with their business partners in an extended and networked enterprise.

Nowadays, the economic context is becoming increasingly networked and dynamic. To get more business opportunities, and survive in the competition, the enterprises must not only consider about the apparent business partners with direct relationship, but also the potential business partners with indirect relationship. This context requires EI research to consider all the cooperative and competitive elements in a very dynamic and complex environment. Thus, the traditional EI solution, such as integrated approach and unified approach as identified in the INTEROP Enterprise Interoperability Framework, is becoming less efficient to satisfy with such current and future economic context. It means that the EI research must concern more about the dynamic nature of future business requirement, both for the single enterprise and for ecosystems. The INTEROP Enterprise Interoperability Framework has also proposed a dynamic solution called federated approach. This approach requires that the interoperability must be established "on-the-fly". It means that the adjustment and accommodation of the models and systems from diverse partners must use a shared ontology or meta-models that are not pre-defined, but formed through dynamic negotiation. Theoretically, the EI development conformed to this federated approach can provide a very flexible and agile interoperability environment that can help enterprises to adapt to the dynamic and evolutionary economic context. However, currently, to completely implement the federated approach seems to be difficult. The Enterprise Interoperability roadmap published by the European Commission had considered the federated approach as one of the research challenges for the years to come (Charalabidis et al., 2008). This doctorate research has identified the challenges as followings:

- Dynamic market and economic context require an enterprise capable of interoperating simultaneously with multiple heterogeneous partners. This means that an enterprise must

be able to adjust and adapt their systems constantly and without delay.

- To adapt and accommodate dynamically to potential interoperability partners, it is necessary to perform 'on-the-fly' needed changes and mapping of systems connected to partners. Consequently the ability to quick reengineer enterprise systems is an important challenge to develop federated approach of EI.
- Prior to any reengineering attempt, another challenge is to be able to model and automatically collect relevant information and data on the legacy systems and software applications already implemented in the enterprise and concerned by the interoperation.
- To dynamically establish interoperability, it is necessary to reduce complexity in EI. How to use interoperability services as "plug-and-play" mechanisms independently of the EI level for which they are designed (higher levels such as business, or lower ones such as technical applications) is another challenge to consider in this research.

2. Contribution of the thesis

The contribution of this research is a Reversible model driven and HLA based methodology for implementing federated approach under the INTEROP Enterprise Interoperability Framework. The priorities of the development of this methodology are:

(1) To develop a semantic interoperability solution through an agile EI analysis process and engineering.
(2) To create a methodology for model use and reuse, that can enhance the rapid and dynamic enterprise interoperability establishment and cooperation environment control.
(3) To elaborate a technical architecture to support the implementation of the "plug-and-play" mechanism.

In order to define this federated methodological approach, many existing methods, architectures, and techniques have been referred to, such as:

- Model Driven Architecture, which can support modularization of development process and enhance the reusability (OMG, 2003).
- Model Driven Interoperability, which can provide guidance on how model driven development (MDD) should be applied to address interoperability (Bourey et al., 2007).
- Architecture Driven Modernization, which can discover models from the coding level of

legacy information system (OMG, 2010).

- Simulation and application distribution frameworks, which can support the information exchange among distributed enterprise systems, such as CORBA (Common Object Request Broker Architecture) (Mowbray et al., 1995), RMI (Remote Method Invocation) (Buss et al., 1998), DIS (Distributed Interactive Simulation) (IEEE, 1995), ALSP (Aggregate Level Simulation Protocol) (Weatherly, 1993), HLA (High Level Architecture) (IEEE, 2000) and SOA (Service-Oriented Architecture) (Gustavson et al., 2005).
- Ontology, which can aid the business community to agree on a common "vision" of the domain (Veltman, 2001).

Moreover, In order to learn how existing interoperability methods or models implement or define interoperability, LISI (Level of Information System Interoperability) reference model (C4ISR, 1998), Database interoperability & Inverted-V model (Tolk, 2001), LCIM (Levels of Conceptual Interoperability Model) (Tolk et al., 2003), and SOSI (System of Systems Interoperability) Model (Morris et al, 2004) have been reviewed as well.

The objectives of these research efforts are as following:

- To develop a federated approach to support establishing enterprise interoperability dynamically in a heterogeneous and multi-partners environment.
- To elaborate a model driven architecture to facilitate re-use of models and re-engineering sub-systems based on models.
- To implement a reverse engineering approach that allows extracting relevant information from legacy systems and software applications for EI engineering or re-engineering.

To achieve these goals, the Reversible Model driven and HLA based methodology proposed by this research provides:

- Firstly, a model driven enterprise interoperability framework enhanced with needed technology to support federated approach of establishing interoperability;
- Secondly, an enterprise interoperability engineering methodology which is composed of a set of methods to support interoperability modelling, "on-the-fly" negotiation design, and model reversal;
- Finally, a computer aided tool to allow implementing the framework, interoperability engineering methodology.

3. Organization of the thesis

This document is organized as follows:

- Chapter 1 introduces the historical, current situations and future challenges of Enterprise Interoperability. This includes context and background of Enterprise Interoperability, concepts and definitions of Enterprise Interoperability, and Enterprise Interoperability research challenges and priorities. At the end, this chapter presents the objective and position of this thesis.
- Chapter 2 presents the state-of-the-art on relevant models of Systems Interoperability, and architectures, techniques and methodologies that are relevant to the development of federated enterprise interoperability, including model driven technologies, simulation and application distribution frameworks, and ontology.
- Chapter 3 presents a harmonized and reversible development framework and methodology for rapidly developing an interoperable and HLA based application from existing enterprise information systems. This framework includes: (1) a harmonized HLA&MDA engineering framework that forms a rapid and flexible development life cycle; (2) model reverse method that discovers the enterprises' knowledge from the legacy information systems; (3) web-enable HLA federate design method that provides a platform for interoperability negotiation; and (4) short-lived ontology method that supports "on-to-fly" negotiation semantically.
- Chapter 4 presents the architecture and the implementation of functional modules of Model driven and HLA based Reverse Engineering Tool based on the framework and methods presented in chapter 3.
- Chapter 5 demonstrates a case study of using this tool based on laboratory data. This case aims at showing the feasibility of the methodology proposed in chapter 3, and the efficiency of the architecture implementation elaborated in chapter 4.

Chapter 1. Towards a federated approach of Enterprise interoperability

1.1. Context and background

This section will introduce the research of enterprise interoperability, including the economic context, research background and industrial requirement.

1.1.1. Economic context

Nowadays, enterprise collaboration becomes more and more important because of globalised economic context. An enterprise often needs to interoperate at the same time with many different heterogeneous partners having different technologies, semantics, methods of work and organizations. In this context, it needs a proper solution to avoid the collaboration barriers caused by those differences. In the last decades, there are many solutions for this enterprise collaboration problem. Generally speaking, those solutions are either enterprise integration (tightly coupled systems) or enterprise interoperability (loosely coupled systems).

As the economic context becomes more and more networked and dynamic, enterprise collaboration are required to be more and more flexible and agile. There is a shift from traditional full enterprise integration paradigm to enterprise interoperability. In this case, enterprise interoperability seems more suitable for and adapted to this context. Besides that, in this new business context, value generation is increasingly knowledge-intensive and requires new and adaptable expertise in products, services, and markets. In this case, the traditional Enterprise Interoperability (EI) solution by connecting partners in an extended and networked enterprise to support business cannot fully satisfy the new economic requirements. EI needs to accommodate continuous and emergent change. Interoperability for enterprises, therefore, is no longer about basic interconnectivity at the level of technology, or basic information exchange between two entities, in static contexts of “universal” business models. Instead, interoperability is closely coupled with the changing nature of business needs, at the level of the enterprise and the community of enterprises, the individual, and the economy (EC, 2008). In other words, the sustainable and dynamic Enterprise Interoperability is calling by current, even future, economic context.

1.1.2. Research background

Since the beginning of 2000s, the European Commission has set up an expert group to

identify problematic/approach relating to the development of interoperability of enterprise software applications in Europe, and to make proposition to the Commission to launch projects in this domain. This group identified three main research themes or domains that address interoperability issues: (1) Enterprise Modelling dealing with the representation of the inter-networked organization to establish interoperability requirements; (2) Architecture & Platform (A&P) defining the implementation solution to achieve interoperability; (3) Ontologies (ONTO) addressing the semantics necessary to assure interoperability (IDEAS, 2003). Based on the recommendation of this expert group, a thematic network Interoperability Development of Enterprise Applications and Software (IDEAS) was launched (July 2002–June 2003). The objective was to elaborate a roadmap to develop interoperability (IST, 2001). This roadmap was used by the Commission to define orientation for future projects under the FP6 (Sixth framework programme) for the years to come.

Two main initiatives relating to interoperability development within FP6 were carried out: ATHENA Integrated Project (IP) and INTEROP Network of Excellence (NoE).

- Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications (ATHENA) is actually a programme. It consists of a set of projects dealing with gaps-closing activities considered as priorities in IDEAS roadmaps and will lead to prototypes, technical specifications, guidelines and best practices that form a common European repository of knowledge (ATHENA, 2003).
- Interoperability Research for Networked Enterprises Applications and Software (INTEROP) aims at integrating expertise in relevant domains for sustainable structure of European Research on Interoperability of Enterprise applications (INTEROP, 2003).

Both ATHENA and INTEROP initiatives have significantly contributed to enterprise interoperability development. They laid down foundations to build enterprise interoperability architectures, models, methodologies and operational solutions.

Besides ATHENA Integrated Project (IP) and INTEROP Network of Excellence (NoE), there are also many research projects contributed on interoperability frameworks in the past decade. For example, LISI (Levels of Information Systems Interoperability) reference model, IDEAS interoperability framework, European interoperability framework, etc. Generally speaking, the main purpose of these frameworks is to provide an organizing mechanism so that concepts,

problems and knowledge on enterprise interoperability can be represented in a more structured way. It is a structure expressed in terms of diagrams, text and formal rules that relates the components of a conceptual entity to each other (EN/ISO, 2003). Nowadays, these frameworks have guided many developments of the interoperability of companies' ICT systems and applications. Meanwhile, most international software, hardware and service vendors have created their own strategies for achieving the goal of open, collaborative, loosely coupled systems and components. However, as usual, the business context changing, human desire growing, and technology progressing will lead constant and dynamic change of market demand. Thus, the new concepts or solutions are needed to complement the traditional frameworks and methodologies, since the existing solutions cannot fully handle the new requirements. The ENSEMBLE FP7 project is providing the framework to validate such results and considerations, working within the Future Internet Enterprise Systems (FInES) community to develop and implement a systematic approach to the establishment of EI as a science (ENSEMBLE, 2011) (Gonçalves et al., 2012).

More recently, some additional projects continue the research and development in enterprise interoperability domain, focusing more specifically on dynamic enterprise interoperability approach. We can mention among others:

- ***ABILITIES (Application Bus for InteroperabiLITy In enlarged Europe SMEs) project*** is a FInES (Future Internet Enterprise Systems) Cluster in FP6. Its objective is to study, design and develop a federated architecture implemented by a set of intelligent and adaptive UBL active messages (an Application Bus for EAI - Enterprise application integration) and basic interoperability services, aiming at supporting SMEs EAI in e-commerce contexts, specifically in less developed Countries and less RTD intensive industrial sectors (ABILITIES, 2008).
- ***COIN (Enterprise Collaboration & Interoperability)*** is one of the FInES Cluster FP7 Projects (COIN, 2011). Its objective is to study, design and develop an open, self-adaptive, generic ICT integrated solution to support the 2020 vision, starting from notable existing research results in the field of Enterprise Interoperability and Enterprise Collaboration. COIN project believes that Enterprise Interoperability and Enterprise Collaboration (two different concepts) can be interdependently and simultaneously presented in every networked enterprise.
- ***NisB (The Network is the Business) project*** is one of the FInES Cluster FP7 Projects

(NisB, 2010). It aims at providing ICT support for value networks of SMEs, namely hierarchical supply chains or dynamic business ecosystems, thereby rendering them the primary facilitators of innovative networked businesses. NisB allows small and medium businesses to easily and affordably connect, align, exchange data and complete transactions with peers they have little common business language with.

Last but not least, it is important to also mention one research initiative developed at IPK Berlin (IST, 2005) and one research project developed at DIP of Genoa University (Bruzzone et al., 2007) (Bruzzone et al., 2009) (Bruzzone et al., 2011) Both of them carried out the distributed modelling and simulation in supply chain management. Both of them used HLA (High Level Architecture) for distributed modelling and simulation.

- ***Research initiative of IPK Berlin***: This project focuses on distributed, decentralised simulation. The concept is based on the results of the European MISSION project and an extension of the Enterprise Modelling Method IEM (Integrated Enterprise Modelling). The approach extends the High Level Architecture (HLA) approach to support the industrial use of distributed simulation.
- ***Research project of DIP of Genoa University:*** This project designed a new modelling methodology to simulate a complex logistics network and ensure interoperability among the co-operators. This project developed an application of intelligent HLA Agent for solving the problems like, distributed production planning and control, multisite production scheduling and optimization, dynamic negotiation, and distributed logistics network optimization.

1.1.3. Industrial requirements

Currently, enterprises face many difficulties related to the lack of interoperability which costs industry huge sums of money.

In 2003, the company budgets for integration projects added up to 30-40% of companies' total IT budgets. This figure led a new industrial goal of the reduction of enterprise application integration costs, and adopted standards to achieve compliant solutions/practices, reducing development and management costs. This means that there was a shift from the past integrated paradigm to federated one. This goal was to reduce Enterprise Application

Integration costs by as much as 40%, starting from early 2005 (IDEAS, 2003).

An investigation performed by the Forrester Group in the US shows that some 40% of ICT (Information and Communications Technology) project costs in most major manufacturing industries can be attributed to solve interoperability problems. The Enterprise Applications Integration (EAI) market is projected to grow to some 7 billion US dollars in 2006 making it the biggest IT market ahead of the Enterprise Architecture market. The investigation report briefly gives industrial evidence of the main causes of non-interoperability. This is done through an effort to understand the complexity of the overall interoperability issue and framing it in a convenient articulated framework that hopefully permits us to position the single contribution to the overall issue (ATHENA, 2005).

From those investigation results, it is clear that enterprises are pursuing the ability of interoperability, and desiring to save the cost of integration projects. Thanks to the technology and standards progress, the industrial goals are coming closer. Many enterprises are attempting to abandon the traditional enterprise application integration (EAI) approaches that have resulted in too monolithic systems. Instead, they are adopting more service-oriented, loosely coupled, messaged-based, and asynchronous techniques (Vernadat, 2007). More recently, with the deployment of new technologies, such as, Web 2.0, and enterprise cloud computing, enterprise needs more and more dynamic engineering capability to allow quickly reconfiguring their systems, in order to set up collaboration relationships with their business partners.

1.2. Concepts and definitions

1.2.1. Basic definitions on enterprise interoperability

As mentioned, interoperability is a key feature for enterprises in today's competitive environment. Generally, "Inter-operate" implies that one system performs an operation on behalf of (or for) another system. However, Interoperability means different things to different people, so there are various kinds of definitions as follows:

- Interoperability is the ability of a system to use the parts of another system – definition in Webster.

- From software engineering point of view, interoperability means that two co-operating software systems can easily work together without a particular interfacing effort. It also means establishing communication and sharing information and services between software applications regardless of hardware platform(s). In other words, it describes whether or not two pieces of software from different vendors, developed with different tools, can work together.
- The definition of Interoperability in IEEE is "the ability of two or more systems or components to exchange information and to use the information that has been exchanged" (IEEE, 1990).
- Ability of interaction between enterprises. The enterprise interoperability is achieved if the interaction can, at least, take place at the three levels: data, application and business process (IDEAS, 2003).

These definitions describe interoperability from different aspects. Some definitions describe the interoperability behaviour. Some others emphasize the information interoperability. Some definitions consider software application interoperability. While, the definition from IDEAS focuses enterprise interoperability on business processes interoperability, not only information interoperability. To summarize those definitions, Enterprise Interoperability is the ability to (1) communicate and exchange information; (2) use the information exchanged; (3) access to functionality of a third system (Chen, 2009).

However, in the last few years, some researches considered that those definitions need to be extended to cover the additional interoperability issues in the enterprises, and a broader, more comprehensive definition is needed. As a result, some new definitions of Enterprise Interoperability were given in different projects.

- Enterprise Interoperability Research Roadmap (EIRR) define Enterprise Interoperability as "a field of activity with the aim to improve the manner in which enterprises, by means of Information and Communications Technologies (ICT), interoperate with other enterprises, organizations, or with other business units of the same enterprise, in order to conduct their business. This enables enterprises to, for instance, build partnerships, deliver new products and services, and/or become more cost efficient" (Charalabidis et al., 2008).
- European Interoperability Framework defines interoperability as "the ability of

information and communication technology (ICT) systems and of the business processes they support to exchange data and to enable the sharing of information and knowledge" (IDABC, 2008). It also indicates "Interoperability is the ability of disparate and diverse organizations to interact towards mutually beneficial and agreed common goals, involving the sharing of information and knowledge between the organizations via the business processes they support, by means of the exchange of data between their respective information and communication technology (ICT) systems" (IDABC, 2008).

These definitions involve interoperability between organizational units and business processes and units either within distributed enterprises or within an enterprise network. In a word, Enterprise Interoperability is perceived as a capacity of two or more enterprises, including all the systems within their boundaries and the external systems that they utilize or are affected by, in order to cooperate seamlessly, in an automated manner, in depth of time for a common objective (ENSEMBLE, 2011) (Gonçalves et al., 2012).

1.2.2. Main concepts of enterprise interoperability

To analyse and summarize those definitions mentioned in the previous section, the enterprise interoperability is an ability that can support the communication and transactions between heterogeneous and networked enterprises / organizations based on shared business references. Those communication and transactions are not only happen on ICT level but also on business level and knowledge level as illustrated in simplified interoperability framework shown in figure 1-1 (ATHENA, 2003). The business level includes the business environment and business processes. The knowledge level includes the organizational roles, skills and competencies of employees and knowledge assets. The ICT level includes the applications, data and communication components. Besides that, semantics description, which can be used to get the necessary mutual understanding between enterprises, exists throughout these three levels. In order to bring this framework into effect, some relevant knowledge (mentioned in section 1.1.2, Enterprise Modelling, Architecture & Platform (A&P), and Ontologies (ONTO)) are needed to model target systems, implement interoperability solutions and translate the semantic differences (ATHENA, 2003).

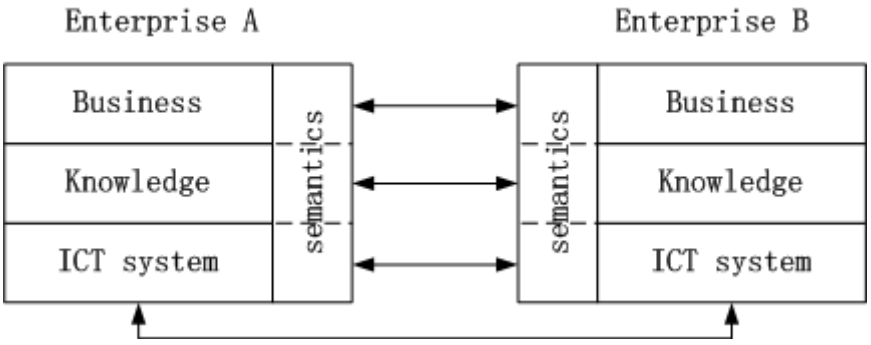

Figure 1-1. Simplified Interoperability Framework

Enterprise interoperability concept is to be distinguished to enterprise integration. Enterprise integration is the process of ensuring the interaction between enterprise entities necessary to achieve domain objectives. Enterprise interoperability refers to the ability of interactions (exchange of information and services) between enterprises.

From the above discussion, it seems that there are not many differences between integration and interoperability, but they are actually different. It is very important to clarify their differences. Intra ERP and EAI implementations are most concerned with 'integration', which can be achieved by using a single integration tool or vendor/integrator. Interoperability has the advantage of using local or company vocabularies rather than conforming to vendor-specific requirements and provides a loosely-coupled architecture, allowing changes to be made on one system without seriously hampering other systems. Some characteristics about interoperability and integration are shown in the table 1-1.

Table 1-1. Characteristics about interoperability and integration (IST, 2005)

Interoperability	**Integration**
Autonomy	Assimilation
Loosely-coupled	Brittle
Sharing	Conforming
Local Vocabulary	Standard Vocabulary
Model-based Maps	Scripts, Functions, Code
Concrete and Conceptual	Concrete

As summary, interoperability has the meaning of coexistence, autonomy and federated environment, whereas integration refers more to the concepts of coordination, coherence and uniformization. From the point of view of degree of coupling, a fully integrated system is

‘tightly coupled’, which indicates that the components are interdependent and cannot be separated. Interoperability is “loosely coupled”, which means that the components are connected by a communication network and they can exchange services while continuing locally their own logic of operation. Thus two integrated systems are inevitably interoperable, meanwhile two interoperable systems are not necessarily integrated.

1.2.3. Main dimensions of enterprise interoperability

To better understand the Enterprise interoperability concept, to define and position our research theme, it is necessary to study various dimensions of enterprise interoperability. Those dimensions representing problems, issues and concerns of EI research and development are usually structured and represented in enterprise interoperability frameworks.

Figure 1-2 shows the INTEROP Enterprise interoperability Framework (now CEN/ISO 11354 standard) (Chen et al., 2008) with its three main dimensions.

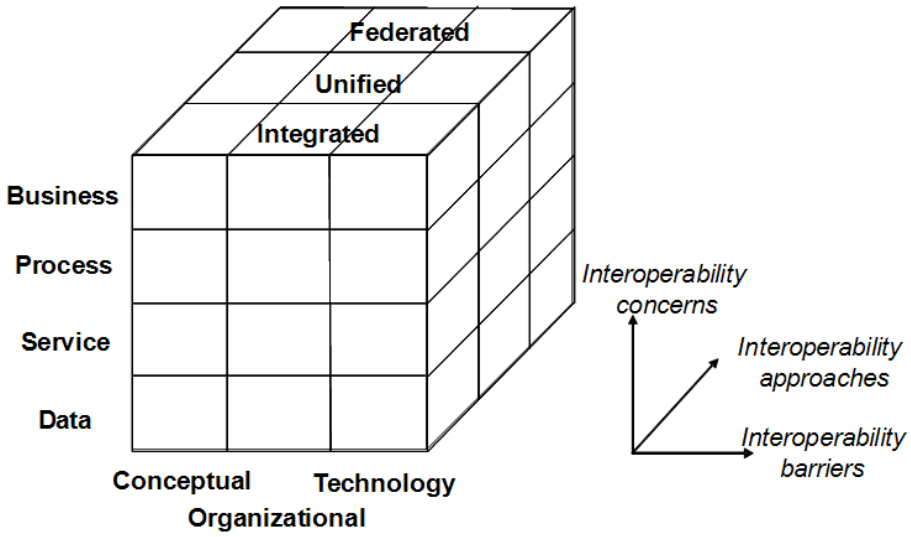

Figure 1-2. Enterprise Interoperability Framework

This framework consists of three basic dimensions: interoperability concerns, interoperability barriers and interoperability approaches. Three categories of barriers are defined: conceptual barriers (syntactic and semantic incompatibilities), technological barriers (additional incompatibility due to the use of technology), and organizational barriers (related to the incompatibilities of method of work, organization structure, etc.). These barriers can exist at four different levels of concerns: data, service, process and business. The interoperability concerns and interoperability barriers can constitute the interoperability problem space. The

intersection of an interoperability barrier and an interoperability concern is the set of interoperability problems having the same barrier and concern. In order to constitute the solution for the interoperability problem, the interoperability approaches are imperative. This framework defines the approaches into three types: integrated, unified, and federated. The following sub-sections will describe the three dimensions of this framework.

1.2.3.1. Interoperability concerns

Interoperability concern is a dimension representing various interoperability aspects (or levels) at which enterprise interoperation takes place. This framework shows that the Enterprise Interoperability can take place at different levels depending on various interoperation aspects. Different research organizations specify the aspects in different ways. For example, (1) ATHENA adopts and reforms the IDEAS simplified interoperability framework, and then proposes the interoperability reference architecture (ATHENA, 2007) as figure 1-3 A shows. This architecture illustrates the interoperations can take place at enterprise/business level, process level, service level and information/data level between provided and required enterprises. (2) Adapted from the ATHENA interoperability reference architecture, INTEROP NoE proposes another interoperability concern categorisation (Chen, 2009) as figure 1-3 B shows. Besides the same concerned levels, this architecture also emphases the interoperability can take place not only between enterprises (inter enterprise interoperability), but also inside one enterprise between different departments (intra enterprise interoperability).

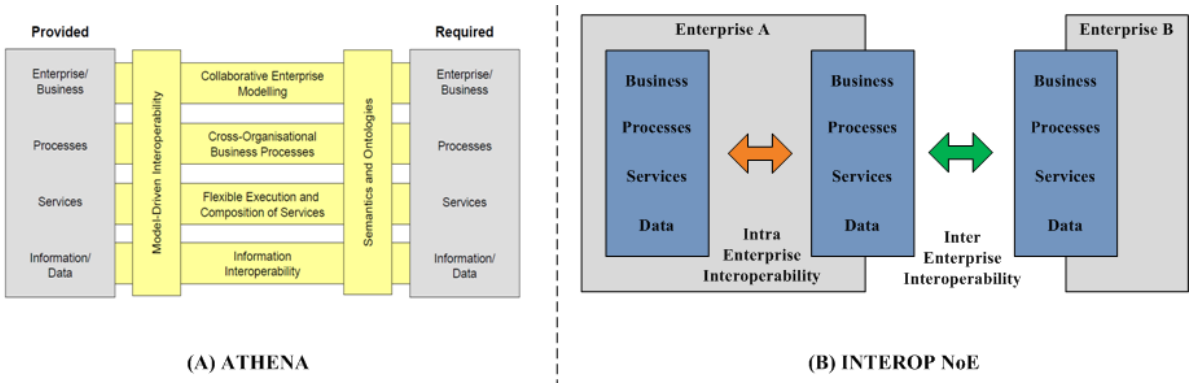

Figure 1-3. Interoperability concerns from ATHENA and INTEROP NoE

In other words, during the enterprise cooperation, depending on the participants' viewpoints and needs, enterprise interoperability will be presented in different ways. For example, according to the INTEROP NoE framework, enterprise needs processes to realise the business,

needs services/functions to materialize the processes, and needs data to perform and simulate services/functions. Therefore, cooperative enterprises can establish enterprise interoperability in the following expectations:

- ***Data interoperability***: it is concerned with finding and sharing information coming from heterogeneous data bases and which can moreover reside on different machines with different operating systems and data bases management systems.
- ***Services interoperability***: it deals with the capability of exchanging services among partners. It has two main problems, service exchange between a service demander and a service provider, and interconnection between different services to form a complex service.
- ***Process interoperability***: it aims at linking different process description to form collaborative processes and perform verification, simulation and execution.
- ***Business interoperability***: it is concerned with how business are understood and shared without ambiguity among interoperation partner. It explores interoperability from a business perspective and identifies the fundamental artefacts related to business issues.

1.2.3.2. Interoperability approaches

Interoperability approach dimension represents various ways or principles according to which an interoperability solution is elaborated. Semantics description part in figure 1-1 shows that we need a proper solution to overcome the gaps at each level, and then to satisfy the interoperability expectation. In other words, establishing interoperability requires relating entities together in some ways. According to ISO 14258 (concepts and rules for enterprise models) (ISO, 1999), there are three basic ways to relate entities together: integrated, unified and federated as shown in figure 1-4.

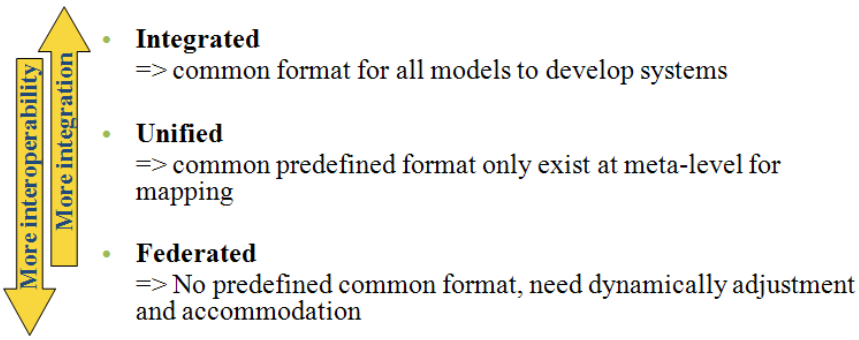

Figure 1-4. Basic approaches to develop interoperability

- ***Integrated approach***: it requires a common format for all constituent systems. Divers models are interpreted in the common format. This format must be as rich as the constituent system models.
- ***Unified approach***: it requires a common predefined format in meta-level. This format covers across the constituent models, providing a means for establishing semantic equivalence.
- ***Federated approach***: it requires that the models must be dynamically accommodated rather than having a predetermined meta-model. This assumes that concept mapping is done at an ontology level, i.e. semantic level.

Therefore, it is considered that the federated approach is the most promising scenario for full interoperability wherein most models will not be in a standardised or common form because it is not economically feasible to put them in such a form (ISO, 1999).

1.2.3.3. Interoperability barriers

Besides the interoperability concerns and approaches mentioned in previous sections, another dimension - interoperability barriers needs to be defined to precisely identify the interoperability problems. "Barrier" means an "incompatibility" or "mismatch" which obstructs the sharing and exchanging of information. The "incompatibility" or "mismatch" can take place in all the concerns. Thus the interoperability concerns and interoperability can constitute the problem space of the enterprise interoperability. Different research organizations specify the barriers in different ways. For example, (1) The European Interoperability Framework in the eGovernment domain (EIF, 2004a) defines three types of interoperability: semantic, technical and organizational. (2) A similar approach was also proposed in e-Health interoperability framework (NEHTA, 2005) which identified three types: organizational, informational and technical interpretabilities. (3) The ATHENA Interoperability Framework (AIF) proposes to structure interoperability issues and solutions at the three levels: conceptual, technical and applicative (ATHENA, 2003). (4) The INTEROP NoE Enterprise Interoperability Framework defined three categories of barriers: conceptual, technological, and organizational (ISO, 2011).

- ***Conceptual Barriers*** are concerned with the syntactic and semantic incompatibilities of information to be exchanged. These problems concern the modelling at the high level of

abstraction (such as the enterprise models of a company) as well as the level of the programming (such as low capacity of semantic representation of XML). Conceptual barriers are the main barriers to interoperability.

- ***Technological Barriers*** are concerned with the use of computer or ICT to communicate and exchange information. The typical technological barriers include incompatibility of IT architecture & platforms, infrastructure, operating system etc. In other words technological barriers occur because of the lack of compatible standards to allow using heterogeneous computing techniques for sharing and exchanging information among systems.
- ***Organizational Barriers*** are concerned with the incompatibilities of organization structure and management techniques implemented in different enterprises. For example, the way of assigning responsibility and authority. These barriers are concerned with human and organization behaviours which can create obstacles to interoperability.

1.3. Research challenges and priorities

1.3.1. Tendency of Enterprise Interoperability – Federated

Nowadays, under the globalised economic context, the markets are becoming more and more competitive and complex. The complex markets require the enterprises to adapt in the dynamic and changing environment. That means Enterprise Interoperability should become more and more related with the dynamic nature of future business requirements, both for the single enterprise and ecosystems.

As mentioned in the Enterprise Interoperability definition of EISB (Enterprise Interoperability Science Base), in order to achieve the common goal and realize the enterprise interoperability, the enterprises need to cooperate seamlessly, in an automated manner, in depth of time. This means that they need a very efficient, dynamic, sustainable and seamless approach/solution, as assumed in Enterprise Interoperability Dynamics. The Enterprise Interoperability Dynamics is the aspiration that the enterprises can be networked fluently, efficiently, dynamically, intelligently and with the lowest cost. In that case, all the collaboration operations can be established “on-the-fly”. Such as, enterprises do not need to think about reconstructing their legacy systems or building up an integrated platform to support cooperation. A potential participant can detect this collaborative sphere and access it easily,

and the existing participants can detect the new partner dynamically and evolve themselves to adapt the new environment. The federated approach aims at achieving this enterprise interoperability dynamics. As mentioned in previous section, in order to establish the interoperability 'on-the-fly', the partners must share ontology or agree on meta-models for mapping between diverse models/systems. However, all these ontology or meta-models are not pre-defined, and they are all formed through dynamic negotiation[1].

Nowadays, most of the approaches developed are unified ones, for example in the domain of enterprise modelling, we can mention UEML (Unified Enterprise Modelling Language) and PSL (Process Specification Language) which aim at supporting the interoperability between enterprise models and tools. However, using the federated approach to develop enterprise interoperability is a challenge and few researches have been performed in this direction. The federated approach aims at developing full interoperability and is particularly suitable for an inter-organizational environment (such as networked enterprises, virtual enterprises, etc.). In the enterprise interoperability roadmap published by the European Commission (Charalabidis et al., 2008), developing federated approach for interoperability is considered as one of the research challenges for the years to come.

1.3.2. Main research challenges

As mentioned previously, because the complex markets require the enterprises to adapt to the dynamic and evolutionary environment, the Enterprise Interoperability is forced to be more and more efficient, dynamic, and sustainable. Thus, how to achieve this goal is becoming the challenge of the Enterprise Interoperability research.

Some of main challenges towards dynamic enterprise interoperability through a federated approach are considered as follows:

- Dynamic market and economic context require an enterprise capable of interoperating simultaneously with multiple heterogeneous partners. This means that an enterprise must be able to adjust and adapt their systems constantly and without delay.

[1] Taking the data interoperability as example, one transnational corporation wants to do the sampling survey from branch companies who are using individual databases with different data structures. During this survey, if the semantic/syntactic annotation and mapping are done by using pre-defined reference ontology, i.e. using pre-defined reference ontology to model a category of products, then it is not a federated approach. While, if the annotation and mapping are performed thought negotiation on the fly, then it is a federated approach. For example, adding the descriptive ontology to data, the similarity of those descriptions will be considered during the dynamic mapping.

- To adapt and accommodate dynamically to potential interoperability partners, it is necessary to perform ‘on-the-fly’ needed changes and mapping of systems connected to partners. Consequently the ability to quickly reengineering enterprise systems is an important challenge to develop federated approach of EI.
- Prior to any reengineering attempt, another challenge is to be able to model and automatically collect relevant information and data on the legacy systems and software applications already implemented in the enterprise and concerned by the interoperation.
- To dynamically establish interoperability, it is necessary to reduce complexity in EI (Enterprise Interoperability). How to use interoperability services as “plug-and-play” mechanisms independently of the EI level for which they are designed (higher levels such as business, or lower ones such as technical applications) is another challenge to consider in this research.

The considerations above point out the bottle neck of the Enterprise Interoperability research, including the huge sum of cost, lack of sustainability (self-adapting and self-learning), lack of succession of knowledge and experience (reusability and repeatability), and complicated preparation and establishment of EI. In other words, these challenges give a novel requirement to EI research. EI is required to be a sustainable interoperability with low cost, excellent discovery ability, learning ability, adaptability, and reusability.

1.3.3. Priorities of development

As mentioned in the previous section, there are many challenges throughout this dynamic EI development research. The priorities of this research are:

(1) To develop a semantic interoperability solution through an agile EI analysis process and engineering.
(2) To create a methodology for model use and reuse, that can enhance the rapid and dynamic enterprise interoperability establishment.
(3) To elaborate a technical architecture to support the implementation of the “plug-and-play” mechanism.

1.4. Objective and position of the thesis

1.4.1. Problem and objectives

The objective of this thesis is to propose a federated approach for developing enterprise interoperability, which allows quick interoperability establishment, easy-pass, and dynamic environment update. In this approach, cooperating parties must accommodate and adjust "on-the-fly" to establish interoperability. "On-the-fly" means that all the models and systems mapping need to be done dynamically through "negotiation". In other words, federated approach has no common predefined format for all models/systems and needs dynamic adjustment and accommodation.

Dynamic adjustment and accommodation is an ideal and prefect idea, but the process of achieving this goal copes with challenges and difficulties. It needs systematic semantic specification and adequate information technique support. This thesis will present a solution, which builds a semantic specification on a harmonized IT environment, to achieve federated approach.

As mentioned before, federated approach requires runtime information analysis without preparative script, thus, the semantic specification here needs to be self-adaptive and easily adaptive. This thesis proposes an ontology based specification called "short-lived ontology". As the name shows, this specification has a non-persistent ontology based parsing script. This script only exists when it is needed and it is easy to understand.

In addition, federated approach requires a flexible and reconfigurable IT environment to support dynamic adjustment and accommodation. Meanwhile, because of using computer or information and communication technology (ICT) to communicate and exchange information, it causes technique barriers in enterprise interoperability. As a result, how to conquer the barriers in technique level is also the concern of this research.

As a summary, the concrete objectives of this thesis are to:

- develop a federated approach to support establishing enterprise interoperability dynamically in a heterogeneous and multi-partners environment;
- elaborate a model driven architecture to facilitate re-use of models and re-engineering of

sub-systems based on models;

- implement a reverse engineering approach that allows extracting relevant information from legacy systems and software applications for EI engineering or re-engineering.

1.4.2. Position of the thesis

This doctorate research can be positioned in the Enterprise Interoperability Framework mentioned in section 1.3.1. The approach of this research is a federated approach, which aims at establishing interoperability by overcoming the conceptual, organizational and technological barriers on data and services concerns (as figure 1-5 shows).

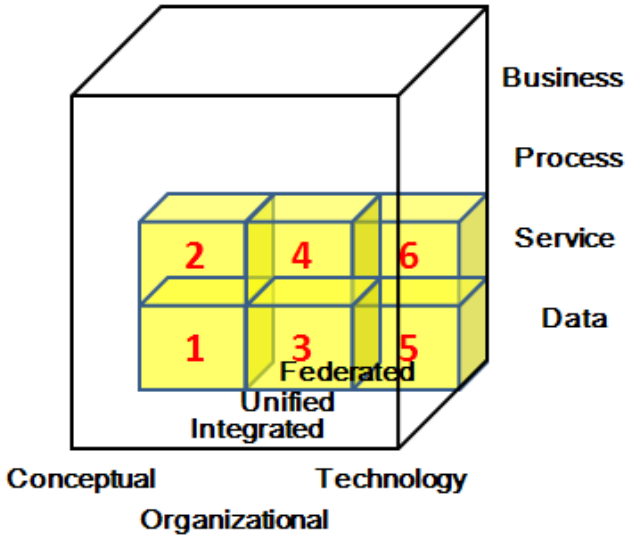

Figure 1-5. Position of the thesis

- The federated approach for problem space of conceptual barrier and data concern (cube 1 in figure 1-5): it has to address the on-the-fly mapping issue of information with different format (syntactic) and meaning (semantic).
- The federated approach for problem space of conceptual barrier and service concern (cube 2 in figure 1-5): it has to provide a platform-independent, technology-independent, and language-independent service for the participants.
- The federated approach for problem space of organizational barrier and data concern (cube 3 in figure 1-5): it has to provide a mechanism to manage the ownership of the data and the authority of obtaining information.
- The federated approach for problem space of organizational barrier and service concern (cube 4 in figure 1-5): it has to provide a mechanism to control the authority of accessing the services.

- The federated approach for problem space of technological barrier and data concern (cube 5 in figure 1-5): it has to provide a data distribution service for implementing on-the-fly information mapping.
- The federated approach for problem space of technological barrier and service concern (cube 6 in figure 1-5): it has to provide a method to accurately define the service interface, so that user can obtain the service correctly and the service authority control can be realized.

1.4.3. Expected results

The expected results of this research are:

(1) A model driven enterprise interoperability framework enhanced with needed technology to support federated approach of establishing interoperability.

(2) An enterprise interoperability engineering methodology which is composed of a set of methods to support interoperability modelling, "on-the-fly" negotiation design, and model reversal.

(3) A computer aided tool to allow implementing the architecture, interoperability engineering methodology.

1.5. Conclusion

This chapter has given an overview of the enterprise interoperability, its context and background, basic definitions, concepts and dimensions. Based on this overview, this chapter elaborates the research challenges and tendency of the current enterprise interoperability research. It has been considered that federated approach represents the most promising solution for today's enterprise to gain competitiveness in the markets. Consequently the objective of this doctoral research aims at taking this challenge contributing to developing federated enterprise interoperability focusing on the data and service levels. In order to achieve the expected goals, this research will propose an innovative methodology based on the state-of-the-art learned from the existing relevant interoperability methodologies and architectures. The chapter 2 will study those methodologies and architectures.

Chapter 2. Methods and architectures relevant to federated enterprise interoperability

2.1. Introduction

This chapter will present the state-of-the-art on the development of Systems Interoperability, and models, architectures, techniques and methodologies which are helpful for the development of federated enterprise interoperability.

Section 2.2 presents some existing models that are considered relevant to federated enterprise interoperability. Useful concepts will be reviewed and abstracted to develop our proposed methodology.

Section 2.3 reviews model driven technologies, in particular Model Driven Architecture (MDA), related approaches using MDA to develop interoperability, such as Model Driven Interoperability (MDI), and model driven reverse engineering technology, e.g. Architecture Driven Modernization (ADM).

Section 2.4 investigates some existing software application architectures / infrastructures that make interoperability happen. In particular HLA, SOA and some other similar approaches will be studied in detail.

Section 2.5 is concerned with ontology techniques to support semantic interoperability development.

Based on this state-of-the-art review, some existing architectures, techniques and methodologies will be adapted if necessary to develop our proposed methodology presented in chapter 3.

2.2. Relevant models for Systems Interoperability

The current situation, tendency and challenges of Enterprises and Enterprise Collaborations show that the Enterprise Interoperability is not the outmoded isolated interoperability and even not just a simple connected interoperability in a peer-to-peer environment, or simple functional interoperability in a distributed environment. It becomes an exhaustive interoperability in a complex distributed enterprises network, in which the individual enterprise plays as a sub-system of a hug system. Thus, the concepts of Systems

interoperability can be borrowed. This section will introduce some maturity models and architectures for systems interoperability. In (Joint, 2000), the perspective of systems interoperability can be described as follow.

"Although technical interoperability is essential, it is not sufficient to ensure effective operations. There must be a suitable focus on procedural and organizational elements, and decision makers at all levels must understand each other's capabilities and constraints. Training and education, experience and exercises, cooperative planning, and skilled liaison at all levels of the joint force will not only overcome the barriers of organizational culture and differing priorities, but will teach members of the joint team to appreciate the full range of Service capabilities available to them."

2.2.1. LISI reference model

LISI (levels of information systems interoperability) approach which is not a framework, but the first significant initiative of Enterprise Interoperability. It is developed by C4ISR (Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance) Architecture Working Group (AWG) during 1997. Its objective is to provide the US Department of Defense (DoD) with a maturity model and a process for determining joint interoperability needs, assessing the ability of the information systems to meet those needs, and selecting pragmatic solutions and a transition path for achieving higher states of capability and interoperability (C4ISR, 1998). It defines the following five layers for technical interoperability:

- Isolated interoperability in a manual environment, where no physical connection exists.
- Connected interoperability in a peer-to-peer environment, where homogeneous product exchange is possible.
- Functional interoperability in a distributed environment, where heterogeneous product exchange is possible.
- Domain based interoperability in an integrated environment, where systems are connected via wide area networks sharing domain-based data models.
- Enterprise-based interoperability in a universal environment, where systems are capable of using a global information space across multiple domains.

LISI model also defines four interoperability attributes as PAID, namely: Procedures, Applications, Infrastructure (hardware, communications, security, and system services) and Data. This is a basic and plain definition which is very useful and heuristic for the following EI research.

The LISI is a widely recognized model for system of systems interoperability, but mainly focuses on technical interoperability, does not address organizational issues. In order to make up for this deficiency, (Clark et al., 1999) proposed the Organizational Interoperability Maturity Model (OIM), which extends the LISI model into the more abstract layers of command and control support. OIM defines five levels of organizational maturity, which describe the ability to interoperate as follows:

- Level 0 - independent: it describes the interaction between independent organizations, which would normally work without any interaction and sharing of common goals. Even if interoperation is required, the arrangements are unplanned and unanticipated. This level can be aligned with isolated level of LISI in manual environment.
- Level 1 - ad hoc: it contains very limited organizational frameworks which could support ad hoc arrangement. The specific arrangements are still unplanned, and the organizations remain entirely distinct. This level can be aligned with connected level of LISI in peer-to-peer environment.
- Level 2 - collaborative: it will use recognised frameworks to support interoperability. Shared goals are recognised, and roles and responsibilities are allocated as part of on-going responsibilities. However, the organizations are still district. This level can be aligned with functional level of LISI in distributed environment.
- Level 3 - integrated (also called combined): it has shared value systems and shared goals, a common understanding and preparedness for interoperation. But, it still has residual attachments to a home organization. This level can be aligned with domain level of LISI in integrated environment.
- Level 4 - unified: it allows the organizational goals, value systems, command structure/style, and knowledge bases to be shared across the systems. There is no impediment in the organizational frameworks to full and complete interoperation. While, it is likely to occur only in very homogeneous organizations. This level can be aligned with the enterprise level of LISI in universal environment.

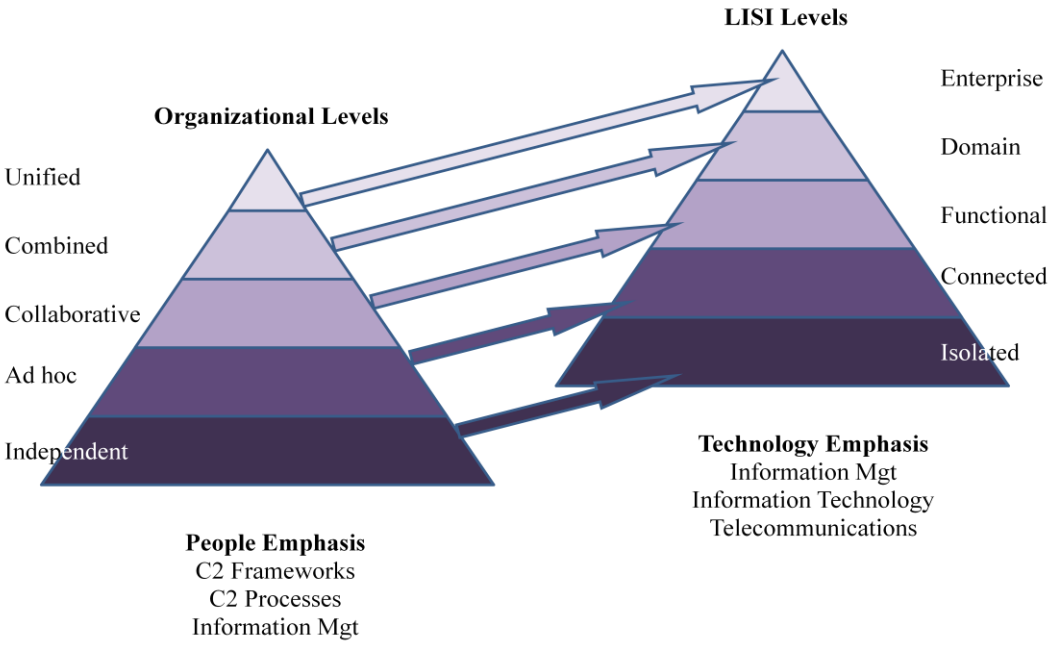

Figure 2-1. Alignment between Organizational Model and LISI

In addition, the four enabling attributes of organizational interoperability have been identified as following:

- Preparedness: it describes the preparedness of the organization to interoperate. It is made up of doctrine, experience and training.
- Understanding: it measures the amount of communication and sharing of knowledge and information within the organization and how the information is used.
- Command Style: it describes the management and command style of the organization – how decisions are made and how roles and responsibilities are allocated or delegated.
- Ethos: it is concerned with the culture and value systems of the organization and the goals and aspiration of the organization. The level of trust within the organization is also included.

2.2.2. Database interoperability & Inverted-V model

(Tolk, 2001) introduces database interoperability by summarizing the study of Sheth's book (Sheth et al., 1990) and (Özsu et al., 1991). Figure 2-2 shows that they have defined the database interoperability into three categorizations based on the degree of database coupling, Homogeneous Non-Distributed Database (figure 2-2 a), Homogeneous Distributed Database (figure 2-2 b), and Federated Database (figure 2-2 c).

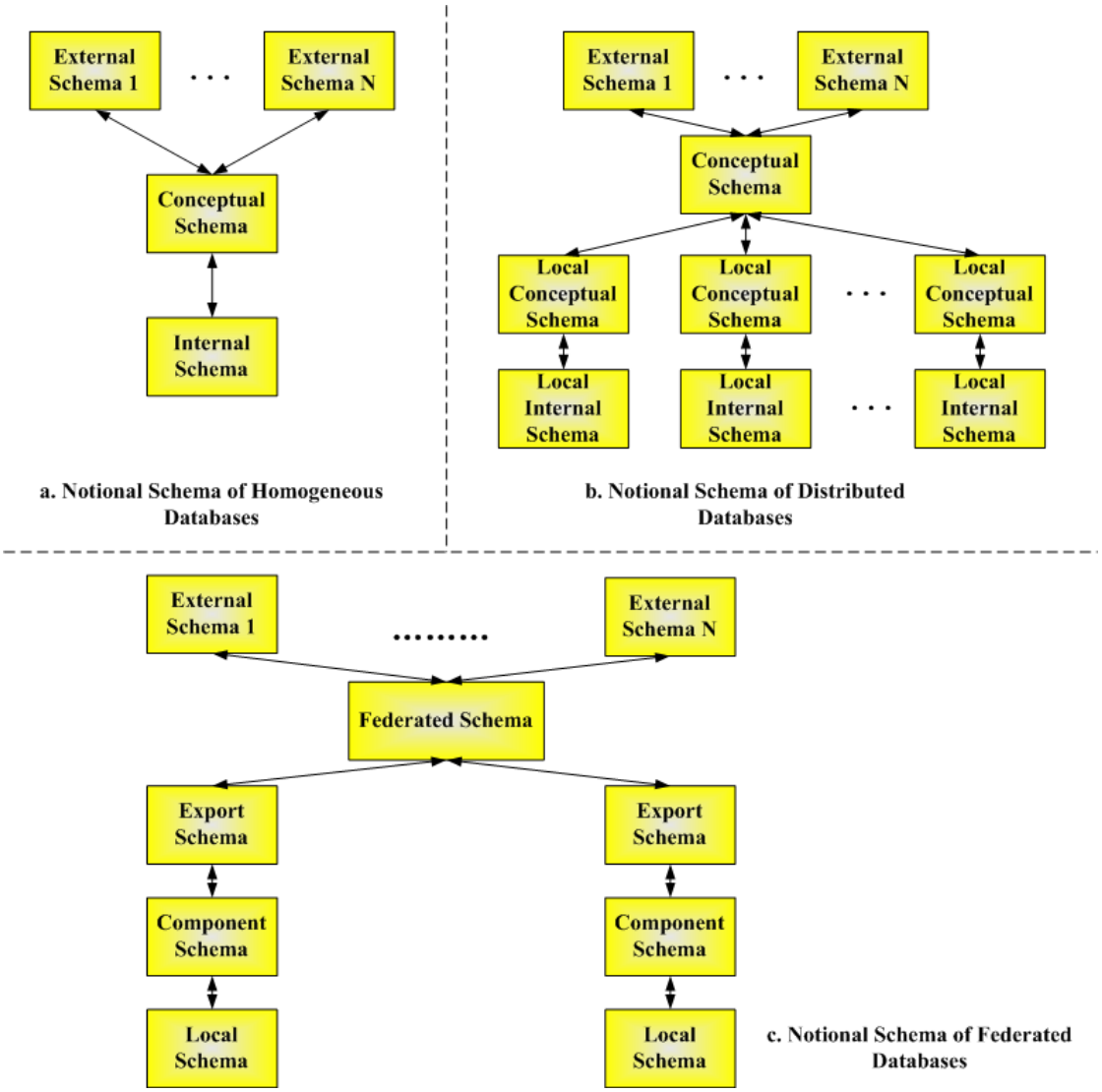

Figure 2-2. Notional Schema of Database Interoperability

- Homogeneous Non-Distributed Database has three standardized levels: (1) internal schema which describes how the data will be physically stored and accessed, using the facilities provided by a particular DBMS; (2) conceptual schema describes the complete stored data in terms of the data model of the DBMS; (3) external schema, for every application, describes the data subset with the respective rights to read, write, and add new data needed for the functionality provided by the application. This notional schema defines the data mapping of the respective information exchange requirement in external schema in local application, not within the architecture.
- Homogeneous Distributed Database has an additional schema, local conceptual schema, compared with Homogeneous Non-Distributed Database. This schema has to be implemented using the respective local internal schema. Besides that, conceptual schema, which is implemented upon the local conceptual schema, is the common conceptual

schema for all the distributed participants/databases. This notional schema is the right architecture and technique for a homogeneous system, where all participants of the database federation are using the same data model, data replication.

- Federated Database is implemented because the scenario of Homogeneous Distributed Database becomes so unlikely within the current joint and combined market context. It is impossible to require all participating systems to use the same common data model. Thus, the objective of federate database is to merge different data sources, which will remain distributed, heterogeneous, and autonomous. In this notional schema, federated schema takes place of conceptual schema to comprise the shared data elements, but not deal with all details of the local autonomous data bases. Component schema is used as the common presentation of the data elements being comprised in the local system dependent schema. Upon it, export schema is used to comprise the data to be shared by the local database with others. This notional schema enables the evolutionary growing of the common data exchange model based on the actual information exchange request being formulated between the global applications and the local databases.

After introducing different schemata for databases interoperability, (Tolk, 2001) also introduces the principles of the Inverted-V model within the use of Standardized Data Elements (SDE) for system coupling as shown in figure 2-3.

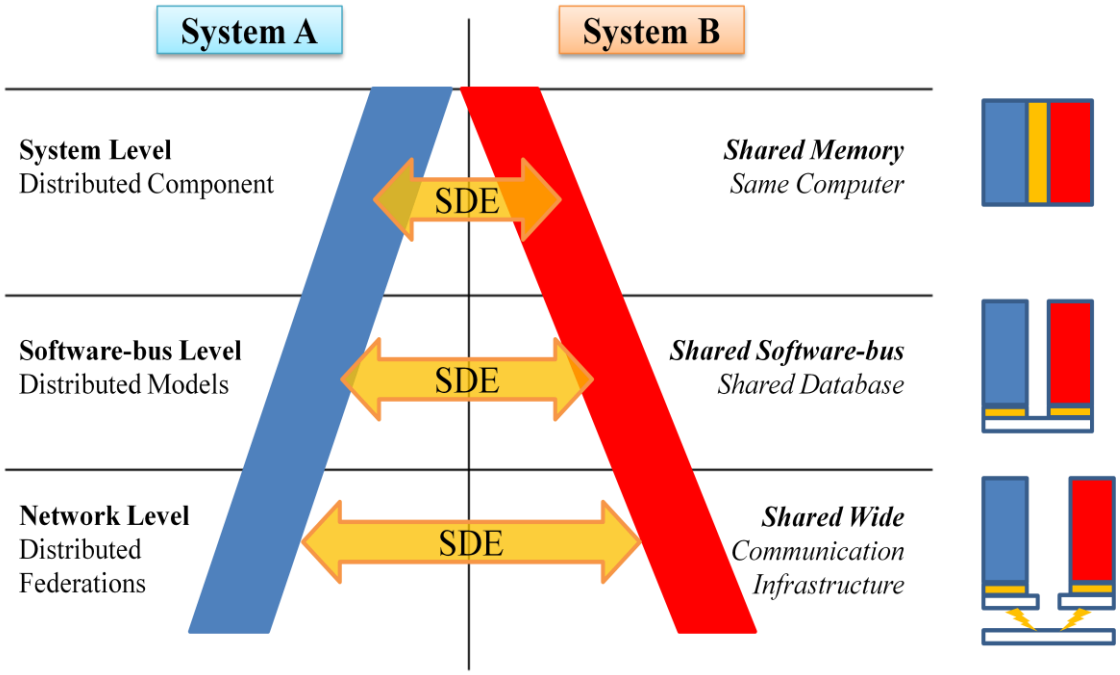

Figure 2-3. The principles of the Inverted-V model

- System level: Systems share the same memory on the same computer, which can be considered distributed components. In this case, it is high coupling, and each component only contributes to the functionality of the system and cares nothing about the way of information interchange among the systems.
- Software-bus Level: Systems share database via software platform. In this case, systems can be considered as sub-system or component of the entire collaborative system, then SDE would be helpful to define the interface between sub-systems/component and shared software-bus. For example, if we start to consider about reuse a legacy system, then the definition of the interface of this legacy system becomes vital, and SDE can be used to describe the data elements of the interface.
- Network Level: Systems exchange information via the communication infrastructure. It is the real use of implemented SDEs to exchange data enables the "plug and play" use of the component in other systems using the same common information exchange data model.

A summary of three aspects of Interoperability in system of system has been given in (Tolk, 2001).

- Information Exchange Aspect: How do systems interchange information? What are the semantics used? How does one describe the objects/concepts used to do this?
- Functional Aspect: What states can the system, which has to be integrated into the federation, be in? What functions are defined, starting at what state, with which respective end state, knowing the used parameters and constraints? What interdependencies can be defined between the state changes?
- Dynamical Aspect: What processing time is needed to perform the transition (1) in "real time" and/or (2) in "simulated time"? How can the dynamic interdependencies be described?

2.2.3. Levels of Conceptual Interoperability Model

(Tolk et al., 2003) introduces a general model called Levels of Conceptual Interoperability Model (LCIM) addressing various levels of conceptual interoperability that goes beyond the technical reference models for interoperable solutions like LISI. The model is intended to become a bridge between the conceptual design and technical design. The scope of this model

goes beyond the implementation level of actual standard, and focus on the data to be interchanged and the available interface documentation. The layers of the LCIM (as shown in figure 2-4) include:

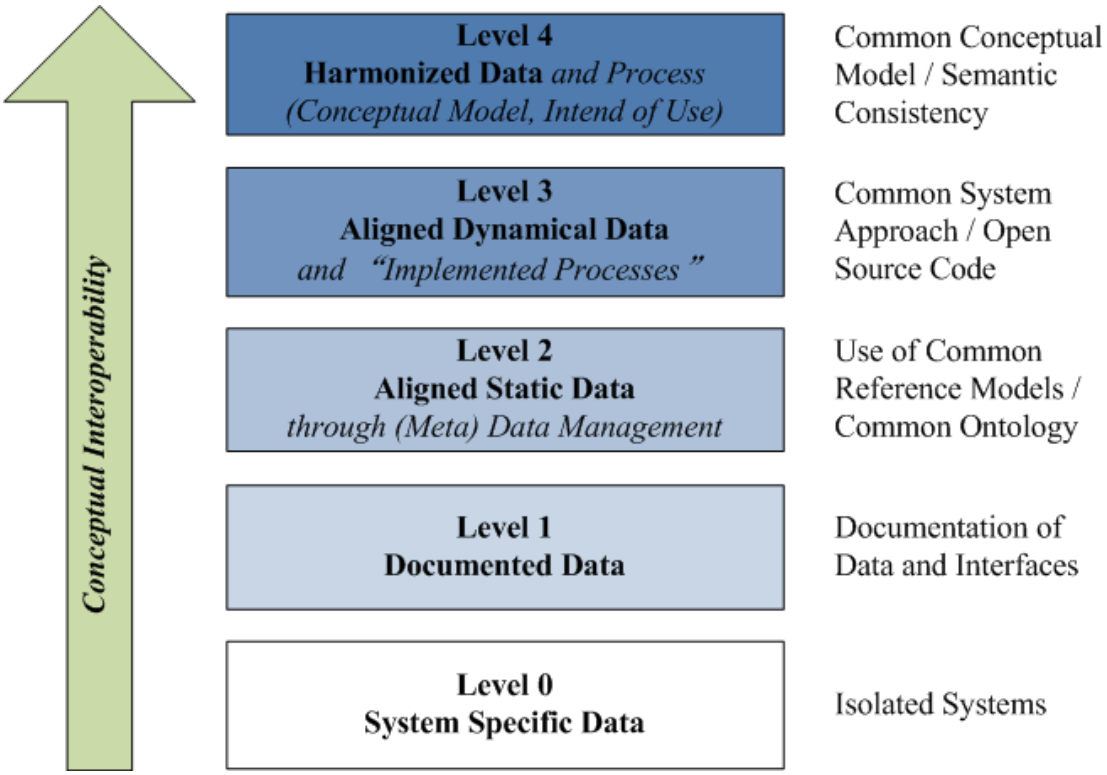

Figure 2-4. Levels of Conceptual Interoperability Model

- Level 0 - System specific data: systems are black box components (or applications), which are interoperable, and use the data in a proprietary way without sharing, for example, data are hard-coded in the source code of the system, and poorly documented data like comma separated lists, and meaningless column name, etc.
- Level 1 - Documented data: systems are black boxes, which have common protocols for data documentation and interface for data access. Based on this, systems can establish mapping layers to interconnect the data with external sources.
- Level 2 - Aligned Static data: systems are black boxes with standard interfaces, and use common reference model based on common ontology for data documentation. The common reference model will take care of the following three kinds of conflicts, semantic conflicts, descriptive conflicts, and heterogeneous conflicts. However, the common reference model is not sufficient for conceptual interoperability, because, even with a common reference model, the same data can be interpreted differently in different systems. Thus, the next dynamic level is required to cope with this.

- Level 3 - Aligned dynamic data: systems are white boxes with well defined data by using standard software engineering methods such as UML (Unified Modeling Language). This allows visibility into how data is managed in the system. This level focus on making the behaviour of the components visible to the integrator, because, even systems with the same interfaces and data can have different assumptions and expectations about the data.
- Level 4 - Harmonized data: systems are white boxes. Non-obvious semantic connections are made apparent via a documented conceptual model underlying components. But not only that, beyond the implemented parts of the concept the important relations that are not captured in the implementation are captured. When doing the modelling, parts of the real world and its relations are left out, which lead to interoperability problems.

2.2.4. The System of Systems Interoperability Model

(Morris et al, 2004) introduces the System of Systems Interoperability (SOSI) Model. This model addresses both technical interoperability (also covered by LISI, and LCI) and operational interoperability (also covered by OIM and LCI). In addition, this model also addresses programmatic concerns between organizations building and maintaining interoperable systems.

(Morris et al, 2004) points out that most of the existing approaches for interoperability only achieve partial interoperability, only specific to the targeted systems but cannot facilitate extension to other systems. Thus, achieving large-scale and consistent interoperation requires a consistently applied set of management, constructive, and operational practices that support that addition of new and upgraded systems to a growing interoperability web. The System Activities Model of SOSI model (as shown in figure 2-5 a) defines necessary activities for achieving interoperability. This model represents the activities within a single acquisition organization. The description of the activities is specified into following aspects:

- Program Management: this aspect defines the activities that manage the acquisition of a system. This aspect specifically concerns the contracts, incentives and practices.
- System Construction: this aspect defines the activities that develop or evolve a system, such as use of standards and COTS (commercial off-the-shelf) products, architecture.
- Operational System: this aspect defines the activities within the executing system and between the executing system and its environment, including the interactions with other

systems and also with end users.

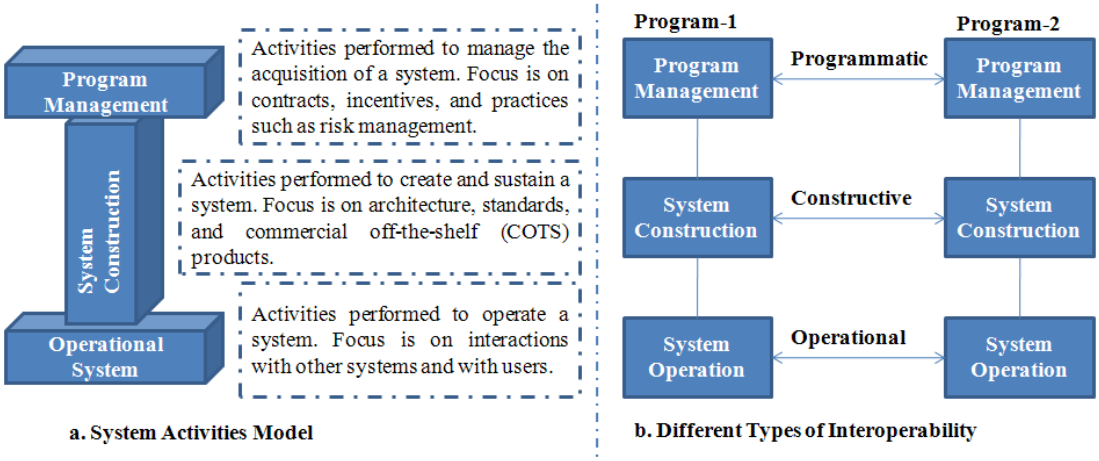

Figure 2-5. System of Systems Interoperability (SOSI) Model

When the interactions occur between two programs, the following types of interoperability (as shown in figure 2-5 b), which is the key premise of the SOSI work, need to be premeditated.

- Programmatic: interoperability between different program offices.
- Constructive: interoperability between the organizations that are responsible for the construction (and maintenance) of a system.
- Operational: interoperability between the systems.

These types of interoperability show that the precondition of SOSI to achieve interoperability between operational systems is to introduce and address the full scope of interoperability between those organizations that participate in the acquisition of systems.

2.2.5. Summary

All models mentioned in this section have achieved some success in developing Systems Interoperability. However, none of them proposes the complete solution for all the interoperability issues.

- LISI focuses on technical interoperability and the complexity of interoperations between systems. But LISI model does not address the environmental and organizational issues that contribute to the construction and maintenance of interoperable systems. OIM can be seen as the evolved LISI model in the context of the layers developed in the command

and control support (C2S) Study by extending LISI into the organizational layer.

- Database interoperability & Inverted-V model is an overall architecture to merge information comprised in heterogeneous data sources into one technically consistent and semantically coherent information space. However, it is only for data but not procedure or architecture.
- The LCIM model has been carried out successfully in simulation domain, but the basic premises apply to many complex sets of interoperating systems.
- The SOSI model extends the existing models by adding a focus on programmatic, constructive and operational issues which must be managed across the life cycle.

Even these models only propose a partial representation of some aspects of interoperability, but they still provide some very useful concepts for identifying and solving enterprise interoperability from the views of conceptual, organizational, and technological barriers.

2.3. Model Driven technologies

The model driven technology aims at supporting the standardization & modularization of system design and development, which enhances the systems/components reusability and interoperability. This section will review some well known popular model driven technologies or evolved model driven technologies.

2.3.1. Model Driven Architecture (MDA)

2.3.1.1. Overview

Model Driven Architecture (MDA) has been defined and adopted by the Object Management Group (OMG) in 2001, and updated in 2003 (OMG, 2003). It is designed to promote the use of models and their transformations to consider and implement different systems as figure 2-6 shows. The MDA has three major goals, which are portability, interoperability and reusability. The MDA starts with the well-known and long established idea of separating the specification of the operation of the system from the details of the way the system uses the capabilities of its software execution platform (e.g. J2EE, CORBA, Microsoft .NET and Web services).

The MDA builds on six basic concepts -- *System, Model, Architecture, Viewpoint, View* and *Platform*. *System* means existing or planed system, which may include a program, a single computer system or some combination of parts of different systems. *Model* is a description or specification of the system modelled and its environment for some certain purpose. *Architecture* is a specification of the parts and connectors of the system and the rules for the interactions of the parts using the connectors. *Viewpoint* is a technique for abstraction using a selected set of architectural concepts and structuring rules. *View* is a representation of the system from the perspective of a chosen viewpoint. *Platform* is a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented.

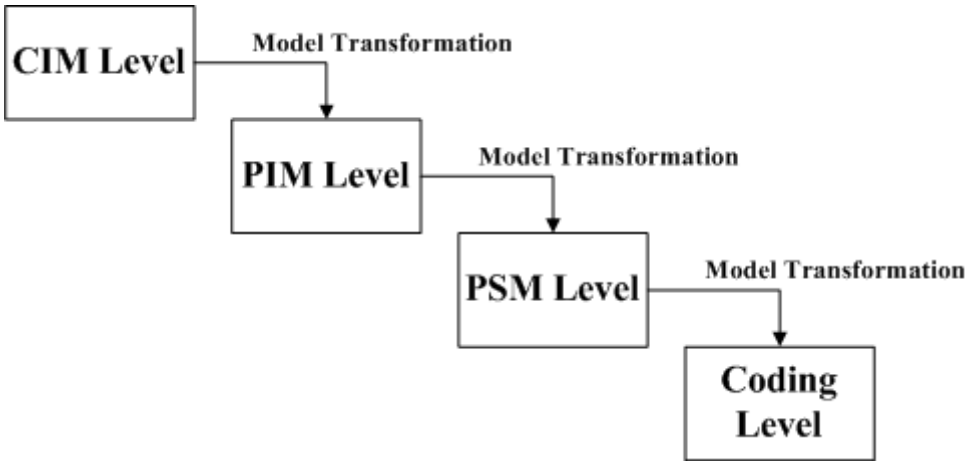

Figure 2-6. OMG's Model Driven Architecture

The MDA defines four levels according to different viewpoints, which go from general considerations (conceptual level) to specific ones (implementation level).

- CIM Level (Computation Independent Model) is a view of a system from the computation independent viewpoint. It focuses on the whole system and its environment. It is also named "domain model". It describes all work field models (functional, organizational, decisional, process, etc.) of the system with a vision independent from implementation.
- PIM Level (Platform Independent Model) is a view of a system from the platform independent viewpoint. It models the sub-set of the system that will be implemented, but does not show the details of its use of its platform. It might consist of enterprise, information and computational viewpoint specifications.

- PSM Level (Platform Specific Model) is a view of a system from the platform specific viewpoint. It takes into account the specificities related to the development platform. It combines the specifications in the PIM with the details that specify how that system uses a particular type of platform.
- Coding Level (Implementation) is last level, consisting in coding enterprises applications (ESA: Enterprise Software Application). It is also a specification, which provides all the information needed to construct a system and to put it into operation.

As the name shows, "Model-driven" means using models to direct the course of understanding, design, construction, deployment, operation, maintenance and modification. Thus, the models of these four levels can be transferred to others under certain order and rules. Model transformation is the process of converting one model to another model of the same system. For example, model transformation from PIM to PSM, the input to the transformation is the marked PIM (a certain mapping assigned) and the mapping (specification for transformation under a particular platform). The result is the PSM and the record of transformation.

2.3.1.2. MDA for Reuse and Interoperability

As mentioned in the overview, MDA provides a systematic architecture to model a system, which can bring amount of advantages including reduction of development cost and complexity and increase of interoperability and reuse. As the enhancement of interoperability and reuse is the most promoted advantages of the MDA (OMG, 2003), and also major concern of this research, so this section will describe how MDA supports interoperability and reuse.

Concerning the MDA for reuse, most of the time, it takes place at these levels or between these levels. For example, reuse of the work field models from a existing CIM to other CIMs; reuse of entities and data types from a PIM to other PIMs; Use of UML profile entities and data types in many PIMs; Reuse of a given PIM as the model for many differing PSMs and implementations; reuse functional module in one PSM to other functional module within this PSM or to other PSMs; and etc. The examples show that the models being reused are general, flexible. They are only focus on one specific problem, and they remove the distraction and complexity. In a word, to reuse the model entities and types defined in an existing MDA

model as the basement for other different business environments, technologies or platforms implementation can reduces development time and effort.

Concerning MDA for interoperability, from intra-system MDA model point of view, the interoperability ability of MDA is not so obvious. However, from inter-system point of view, it will be very clear. As the MDA model transformation shows that, the model transformation starts from PIM to PSM, than to implementation depending on different techniques and platforms. Because PIM model is an abstract model contains enterprise, information and computational viewpoint specifications and includes the mappings to the implementation technology, if two system implementations are derived from the same PIM, then a bridge between these two implementations can be generated based on those known and standardized clues. In this way, the bridge enables the interoperability between these two system implementations. This example shows that to reuse the existing entities, types with a given PIM to guide a new implement across different technologies or platforms, a mapping or relationship among those implementations is concealed. Then, because the MDA around open, supported standards allows all models, data types and entities to be represented in a single, consistent manner, the interoperability of those implementations can be achieved.

Actually, to reuse or to map the model in PIM model showed in the example is just one way to achieve the interoperability. The interoperability can be achieved in even more abstract level, such as remove the business duplicate issues in CIM level, or in more detail level, such as adjust the function module in PSM level. The agile MDA model allows developer to realize the interoperability in different levels. This must be the original idea of Model Driven Interoperability, which will be introduced in next section.

2.3.2. Model Driven Interoperability (MDI) architecture

As previous section mentioned, the MDA provides a way for developing modern enterprise applications and software systems, meanwhile, it also provides a better way of addressing and solving interoperability issues compared to earlier non-modeling approaches. In addition, from an interoperability point of view, most of the enterprises build their information system by using MDA, so it seems that MDA is a good solution for overcoming the interoperability barriers (Ullberg et al., 2007). As a result, the researchers believe that an interoperability framework based on MDA can provide guidance on how model driven development (MDD)

should be applied to address interoperability. Thus, Model Driven Interoperability (MDI) framework is created for how to apply Model Driven Development (MDD) in software engineering disciplines in order to support the business interoperability needs of an enterprise (Elvesæter et al., 2007). It is a model driven method that considers interoperability problems at the enterprise model level instead of only at the coding level. It provides a foundation, consisting of a set of reference models. Figure 2-7 shows the reference model of MDI approach which performs different abstraction in each MDA levels. Between each level of models, the successive model transformations are carried out to reduce the gap existing between enterprise models and code level. The models at the various levels may be semantically annotated (such as reference ontology) which helps to achieve mutual understanding on all levels. The mutual understanding also helps to achieve model interoperability horizontally between different enterprises' model in homologous level.

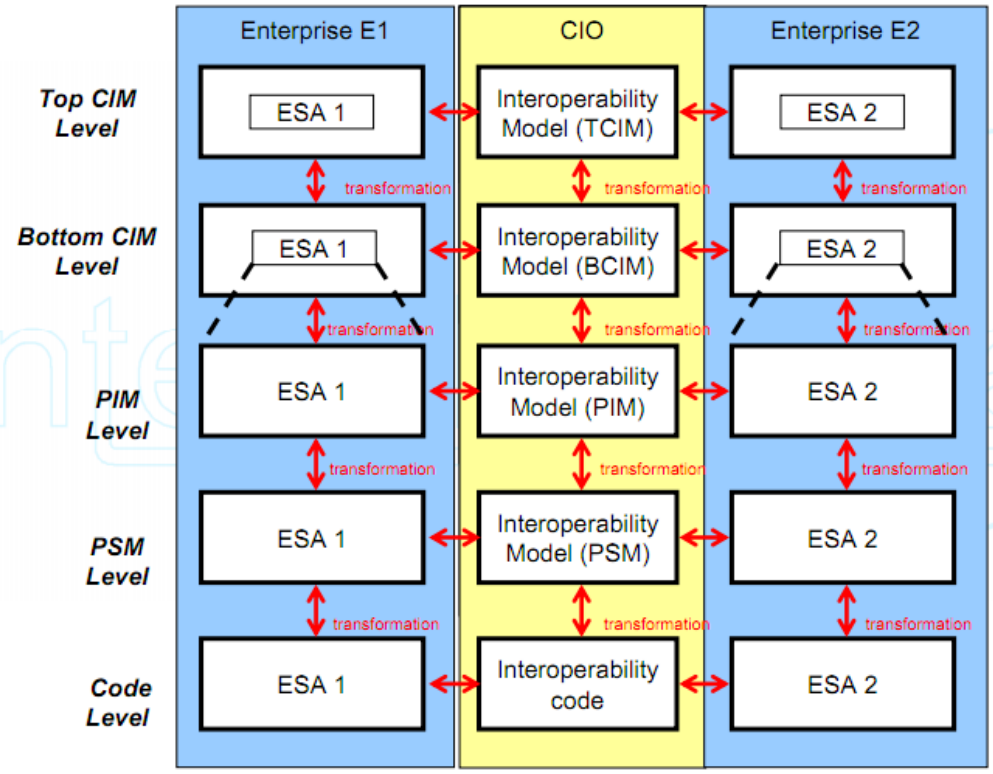

Figure 2-7. Reference model for MDI

The concepts of this method were realized in the Task Group 2 (TG2) of INTEROP-NoE project by defining an approach inspired by the OMG MDA concepts (Bourey et al., 2007). The goal of MDI is to tackle the interoperability problems at each abstraction level defined in MDA and to use model transformation technique to link both vertically the different levels of the MDA abstraction and horizontally the corresponding models of the systems to interoperate.

The main goal of MDI, based on model transformation, is to allow a complete follow-up from the expression of requirements to the coding of solutions and also to provide a greater flexibility thanks to the automation of these transformations.

In the context of TG2, experimentations have been realized and in particular the feasibility study to transform GRAI Methodology (Chen et al., 1997) (Doumeingts et al., 2001) Models to UML models between CIM and PIM levels (Bourey et al., 2007). These works are complemented by additional works realized in the context of ATHENA to define UML profiles to take into account also the Service Oriented Architectures (SOA) at the PIM level (Gorka et al., 2007). These results have been complemented by results presented by (Touzi, 2007) who has proposed an interoperability transformations method from BPMN to UML in the context of SOA.

2.3.3. Architecture Driven Modernization (ADM)

MDA is well-known for promoting the use of models and their transformations to design and implement different information systems. After MDA became an important change in software development, OMG launched another research activity leading to what was later called Architecture Driven Modernization (ADM) (OMG, 2010).

The basic idea proposed in the MDA approach is to translate from an abstract platform-independent model (PIM) expressed in UML into a more concrete platform-specific model (PSM) from which the code still needs to be generated (OMG, 2003). Reversing the MDA lifecycle, ADM is discovering models from the coding level of legacy information system, such as UML models, Knowledge Discovery Meta-model (KDM) and Abstract Syntax Tree Meta-model (ASTM). KDM and ASTM are aimed to satisfy someone interested in discover more specific models from a legacy system (OMG, 2010).

2.3.3.1. KDM - Knowledge Discovery Meta-model

KDM is a meta-model for representing existing software assets and their associations, as well as relationships among the function models in the system (OMG, 2010). It also describes the operation environments. It can insure the interoperability among the existing systems, make the data exchange among different vendor tools easier. As shown in figure 2-8, KDM contains

4 layers and 12 packages. The four layers are Infrastructure layer, Program Elements layer, Runtime Resource layer and Abstractions layer. The twelve packages are located in the different four layers. The Infrastructure layer consists of core package, kdm package, and source package. In the Program Elements layer, there are code package and action package. The data package, UI package, Event package, and platform package are located in the Runtime Resource layer. The conceptual package, structure package, and build package are located in the Abstraction layer.

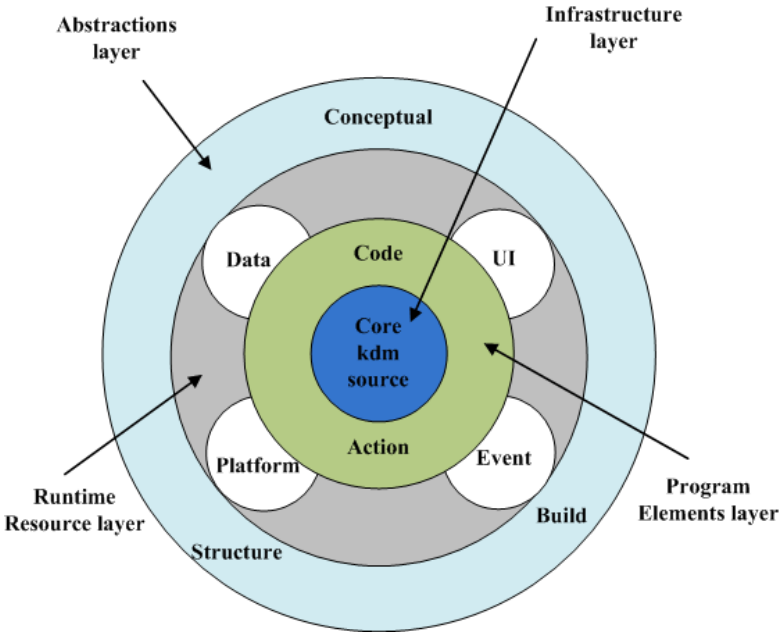

Figure 2-8. Layers, packages, and specification of concerns in KDM

2.3.3.2. ASTM - Abstract Syntax Tree Meta-model

ASTM aims at enabling easy interchange of detailed software metadata between software development and software modernization tools, platforms, and metadata repositories in distributed heterogeneous environments (OMG, 2011a). It defines a specification for modeling elements to express abstract syntax trees (AST) in a representation that is sharable among multiple tools from different vendors.

The Abstract Syntax Tree Metamodeling specification mainly consists of definitions of metamodels software application artifacts in the following domains:

- Generic Abstract Syntax Tree Metamodel (GASTM): A generic set of language modeling elements common across numerous languages establishes a common core for language

modeling, called the Generic Abstract Syntax Trees. In this specification the GASTM model elements are expressed as UML class diagrams.

- Language Specific Abstract Syntax Tree Metamodels (SASTM) for particular languages such as Ada[2], C, Fortran, Java, etc. are modeled in Meta Object Facility (MOF) or MOF compatible forms and expressed as the GASTM along with modeling element extensions sufficient to capture the language.
- Proprietary Abstract Syntax Tree Metamodels (PASTM) express ASTs for languages such as Ada, C, COBOL[3], etc. modeled in formats that are not consistent with MOF, the GSATM, or SASTM. For such proprietary AST this specification defines the minimum conformance specifications needed to support model interchange.

In a word, the KDM establishes a specification for abstract semantic graph models, while the ASTM establishes a specification for abstract syntax tree models. The relationships between these two are detailed in (OMG, 2011a).

2.3.3.3. Model Reverse Tool

Nowadays, there are many software tools developed based on model reversal theories. We choose MoDisco (for Model Discovery) tool which is an Eclipse GMT (Generative Modeling Technologies) component for model-driven reverse engineering. The reason of choosing MoDisco is that it is an open source plug-in of the Eclipse that is our research development IDE (Integrated Development Environment) and its result is a readable UML file in XML format, so it is very convenient to import the MoDisco and its result into our application (Bézivin et al., 2006).

The objective of MoDisco is to allow practical extractions of models from legacy systems. MoDisco proposes a generic and extensible metamodel-driven approach to model discovery and use a basic framework and a set of guidelines to discover models in various kinds of legacy systems.

As a GMT component, MoDisco will make good use of other GMT components or solutions available in the Eclipse Modeling Project (Eclipse Modeling Framework - EMF, Model To

[2] Ada is a structured, statically typed, imperative, wide-spectrum, and object-oriented high-level computer programming language, extended from Pascal and other languages (Gehani, 1983).

[3] COmmon Business-Oriented Language is one of the oldest programming languages. Its primary domain is in business, finance, and administrative systems for companies and governments (Sammet, 1978).

Model - M2M, GMF, Textual Modeling Framework - TMF, etc), and more generally of any plug-in available in the Eclipse environment.

MoDisco can extract XML model, KDM model, KDM code model, JAVA code model, UML model and etc. Our research will only use the intuitive and intelligible UML model to analyse interoperability issues which will be discussed in chapter 3. The installation and usage of MoDisco Tool can be found in the (MoDisco, 2012a) (MoDisco, 2012b).

2.3.4. Summary

This section has presented a survey on MDA, MDI and ADM. All of them have the highlights in standardization & modularization of system design and development, but also have the drawbacks that need to be improved. The summary of these technologies are as following:

- The MDA approach contributes on building an interoperable ICT model, from enterprise models to technology models. Those models are able to be aligned by using common meta-model. MDA also provides flexibility and adaptability to accommodate changes at a higher abstraction level. Furthermore, Model transformation ensures the interoperability achievement and/or agreement from higher level to infrastructure (lower level). Besides that, it allows document transformations on the fly, and can contribute to new approaches for semantic interpretations on information exchanges. However, no matter how many advantages MDA has, there are still many people doubt on its performance in practice. For example, (Ambler, 2003) doubted that MDA will follow the old way of Integrated Computer-Aided Software Engineering to ruin, to spend 10 percent effort to generate incomplete and useless code (80 to 90 percent), but spend 90 percent effort on struggling in tracing down the rest part to achieve perfection. In addition, the information is losing during the model transformation, such as details of system behaviours. Therefore, how to use MDA in helping achieve federated interoperability becomes a big concern of this thesis. The section 3.2 will introduce a harmonized HLA&MDA engineering framework that can improve the model transformation.
- Nevertheless, the soundness of the MDI methodology has been demonstrated in the current researches, but no full industrial scale validation has been yet achieved. Only some projects have been especially carried to demonstrate these concepts in an industrial real world significant application. The different methodological propositions are tested

and refined by focusing on models and their interoperability. They consist in particular of ways to improve the flexibility of the MDI transformation process and in obtaining dynamic interoperability in the context of the federated approach.

- ADM shows its strong power in obtaining information from the legacy systems. But, many people doubt on the validity of this information for achieving federated enterprise interoperability. ADM met the same model transformation problems as MDA. In addition, most of the current researches are focus on obtaining static models from the existing systems which cannot fully describe the systems. Most of the time, the reversed models can only be a guideline for the system reconstruction. Thus, the model reverse engineering did not achieve its real intention. The method introduced in section 3.3 will specify the usage of reversed static model for achieving interoperability and propose a way to obtain dynamic models that can describe the business behaviour of the enterprise. The static models and dynamic models will be used to generate an intelligent agent for establishing enterprise interoperability without reconstructing the system of each participant.

2.4. Simulation and application distribution frameworks

Since 1970s, people started to use computer to help manufacturing and named this activity as "Informatization", human civilization had moved into information age. The information technology (IT) is never-ending changes and improvement. Nowadays, IT has permeated through almost all the human activities, and of course, enterprise management is not an exception. Enterprise informatization and networked enterprise become inevitable trend. Thus, federated approach requires a flexible and advanced IT environment to support dynamic adjustment and accommodation. Thus, this section will give a brief survey of some typical and popular IT technologies that can promote distributed systems interoperability.

2.4.1. CORBA and RMI

CORBA (Common Object Request Broker Architecture) was developed and standardised by the Object Management Group (OMG). CORBA can link disparate applications together, which means that distributed, heterogeneous application can communicate with each other in a location and language independent manner (McCarty et al., 1998).

As shown in part (a) of the figure 2-9, the remote client application can request the public

interface in the remote server by using the Interface Definition Language (IDL). There is an IDL stub at the client side and an IDL skeleton at the server side. The IDL provides a programming language neutral method for specifying the specifics of an interface. It can also be used by other frameworks to generate the necessary stub code that will facilitate distributed communication (Mowbray et al., 1995).

In addition, the communication can only be carried out within the Object Request Broker (ORB), which is achieved by defining a generalised communications protocol – the Inter-ORB Protocol (IIOP). This protocol standardises the format of communications that are passing between the distributed CORBA based applications. This protocol also allows clients written in any programming language and on any platform to communicate with one another.

RMI (Remote Method Invocation) was developed by Sun Microsystems. Originally, RMI only supported the Java programming language, but the recent versions have added the IIOP protocol used by CORBA. RMI is similar to CORBA. It allows the programmers to write object-oriented programming in which objects on different computers can interact in a distributed network (Buss et al., 1998).

As shown in part (b) of the figure 2-9, the RMI system consists of three layers:

- The stub/skeleton layer: client-side stubs (proxies) and corresponding server-side skeletons. The stub appears to the calling program to be the program being called for a service.
- The remote reference layer: remote reference behaviour that can be different depending on the parameters passed by the calling program. (e.g. invocation to a single object or to a replicated object)
- The transport layer: connection set up and management and remote object tracking

The client uses the stub (proxy) to invoke a method on the remote server. The local stub is an implementation of the remote interfaces of the remote object. It holds a reference to the remote object and forwards the invocation requests to the server via the remote reference layer. The remote reference layer is responsible for carrying out the semantics of the invocation. The transport layer takes in charge of connection set-up and management. It also keeps track of remote objects (the targets of remote calls) and dispatches them to the transport's address space.

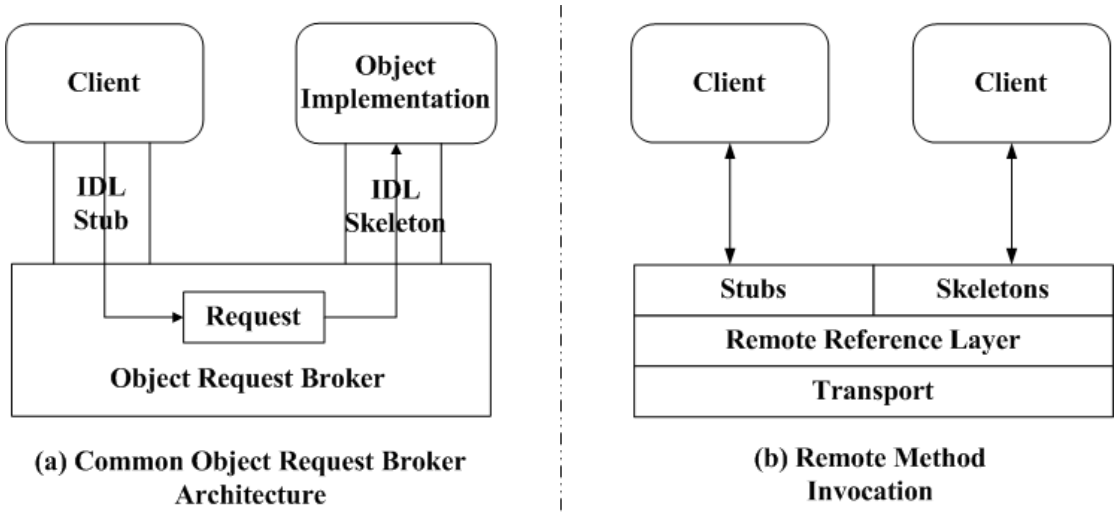

Figure 2-9. CORBA and RMI

Summary: This section has studied the CORBA and RMI. Both of them strongly support the interoperation of software application on in a distributed environment. But they cannot provide advance simulation services, such as integrated time management, interest specification, ownership management and data distribution services. Without these services, it is very hard to create a flexible and adaptable interoperability environment. The event control, time management can rarely be implemented. The organization barrier of EI will be a Chinese puzzle.

2.4.2. DIS and ALSP

DIS (Distributed Interactive Simulation) is a government/industry initiative to define an infrastructure for linking simulations of various types at multiple locations to create realistic, complex, virtual worlds for the simulation of highly interactive activities (IEEE, 1995). As the figure 2-10 shown, the DIS network can realize the communication among different systems built for separate purposes, with different technologies, and providing different products/services, so that they can interoperate. A standard set of Protocol Data Unit (PDU) has been defined for describing the format of messages exchanged between participating simulation hosts. The individual simulation host has a dis_mgr, which is PDUs dispatcher between the DIS network and application programs. The client-server protocol implemented between the dis_mgr and application programs use TCP/IP (Transmission Control Protocol/Internet Protocol) to exchange information. The connection between the DIS network and dis_mgr is based on UDP/IP (User Datagram Protocol/ Internet Protocol). Once

the simulation host changes its state, it will broadcast a message to all other participants.

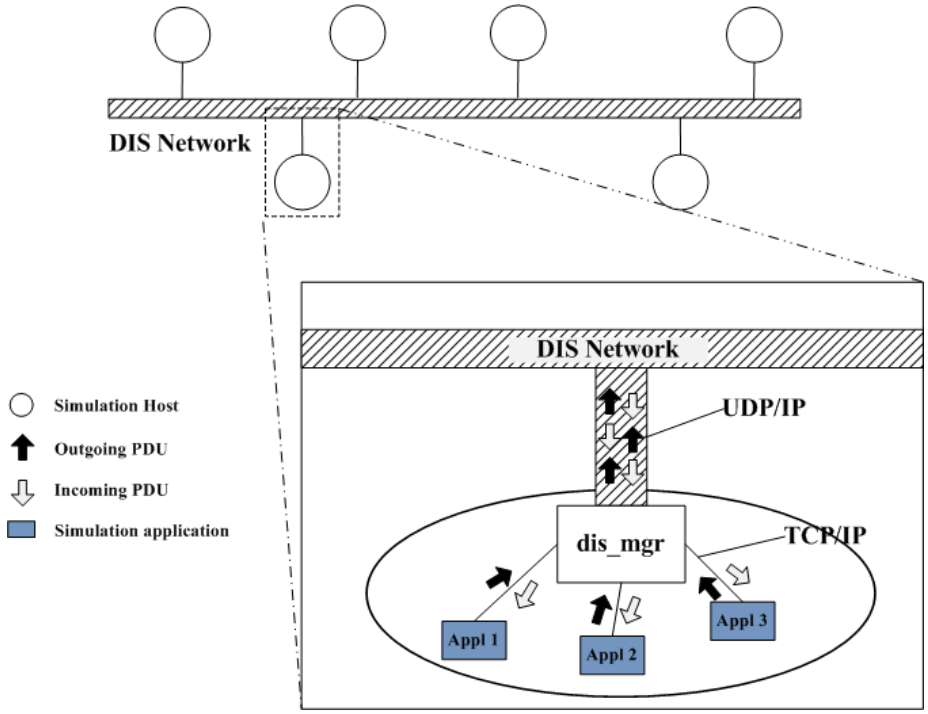

Figure 2-10. Distributed Interactive Simulation

The ALSP (Aggregate Level Simulation Protocol) is under the auspice of the Advanced Distributed Simulation, which is the nomenclature emanating from the U.S. Department of Defense. It provides a mechanism for the integration of the existing simulation models to support training via theater-level simulation exercises (Weatherly, 1993).

Similar to DIS, ALSP describes a collection of infrastructure software and protocols for passing the messages between the various participants of a distributed simulation. Different from DIS, ALSP has global time synchronization and use object-oriented approach to describe the shared object model of a distributed simulation.

Summary: This section has briefly introduced the DIS and ALSP. Both of them provide a protocol of the distributed systems communication. Both of them have proven successful in supporting the interoperation of disparate systems/platforms/services, and the ALSP even starts to take into account the time issue. However, they still cannot support time management and data distribution management. In this case, they still cannot fully satisfy the requirement of the federated approach proposed in this thesis.

2.4.3. High Level Architecture

2.4.3.1. Overview

The High Level Architecture (HLA) is a software architecture specification that defines how to create a global software execution composed of distributed simulations and software applications. This standard was originally introduced by the Defence Modelling and Simulation Office (DMSO) of the US Department Of Defence (DOD). The original goal was reuse and interoperability of military applications, simulations and sensors.

In HLA, every participating application is called "federate". A federate interacts with other federates within a HLA federation, which is in fact a group of federates. The HLA set of definitions brought about the creation of the standard 1.3 in 1996, which evolved to HLA 1516 in 2000 (IEEE, 2000). In order to benefit from the Web Services such as, the support for numerous newer and older languages and operating systems as well as the ease of deployment across wide area networks, HLA evolved IEEE 1516^{TM}-2010 was published in August, 2010 (IEEE, 2010).

Run Time Infrastructure (RTI): RTI is the supportive middleware for the distributed simulation. It is the fundamental component of HLA. It provides a set of software services for the dynamic information management and inheritance, in which federates coordinate their operations and exchange data during a runtime execution.

According to the HLA interface specification, RTI provides six management services: Federation management, Time management, Declaration management, Object management, Ownership management and Data distribution management.

Several commercial RTI software tools coexist such as Pitch portable RTI (pRTI), MAK Real-time RTI, BH RTI and etc. There is also open source RTI software, such as Portico RTI. Portico RTI is chosen for this doctorate research, because Portico is a fully supported, open source, cross-platform HLA RTI implementation. Designed with modularity and flexibility in mind, Portico is intended to provide a production grade RTI implementation and an environment that can support continued research and development.

HLA Federate: The Federate A and Federate B in figure 2-11 shows the structure of a single federate. A HLA federate has two parts, federate code and local RTI component code (LRC). The federate code is the user's code for a federate which is linked with Local RTI Component Code from the C++ library LibRTI to form a complete federate. The local RTI components provide the services for the federate through communication with the RTI executive component, the Federation executive component and other federates. Those services can be obtained by calling the member functions of Class RTI::RTIAmbassador, which is contained in the LibRTI. The federate code has to extend and implement RTI::Federate Ambassador, because when the RTI sends messages and responses to the federate code, it needs to call functions implemented in the federate which are known as callback functions and are implemented as a subclass of Class RTI::FederateAmbassador. Class RTI::FederateAmbassador is also contained in LibRTI, and contains pure virtual functions for each possible callback. These routines are simply "place holders" that cannot be called. The federate code must create a derived class from this class that contains the actual implementation for each of these callback functions.

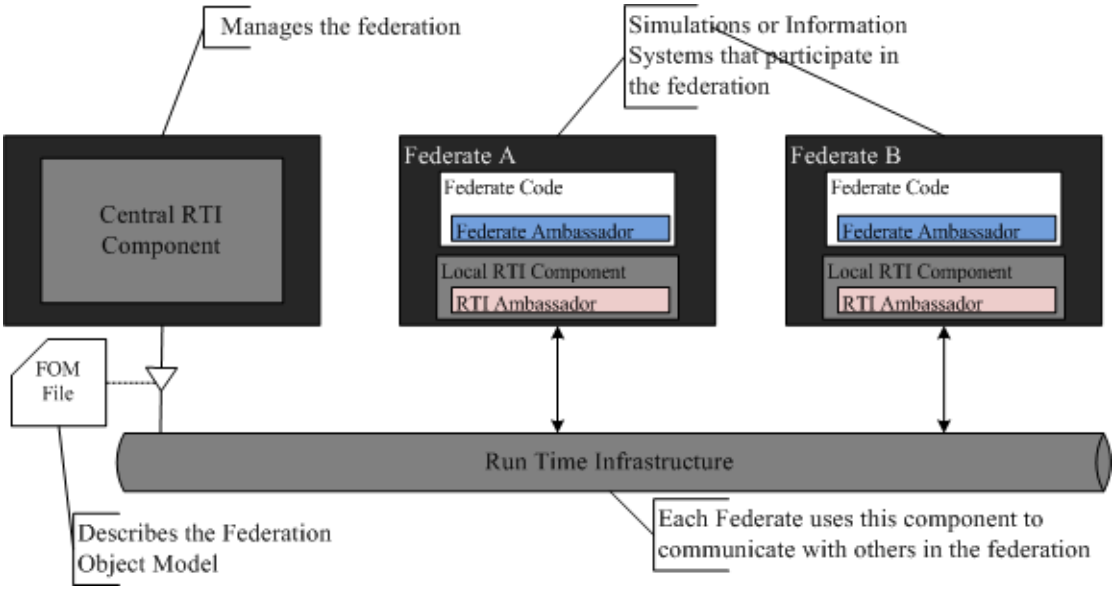

Figure 2-11. High Level Architecture

HLA Models: The interface specification of HLA describes how to communicate within the federation through the implementation of HLA specification: the Run Time Infrastructure. Federates interact using services proposed by the RTI. They can notably "Publish" to inform about an intention to send information to the federation and "Subscribe" to reflect some information created and updated by other federates. The information exchanged in HLA is represented in the form of classical object class oriented programming. The two kinds of

object exchanged in HLA are Object Class and Interaction Class. Object class contains object-oriented data shared in the federation that persists during the run time; Interaction class data are just sent and received information between federates. These objects are implemented within XML format. More details on RTI services and information distributed in HLA are presented in (IEEE, 2000).

In order to respect the temporal causality relations in the execution of distributed computerized applications, HLA proposes to use classical conservative or optimistic synchronization mechanisms (Fujimoto, 2000). In particular, the Lookahead is an important notion in conservative approach, it is the Delay given by an influencer federate to the RTI. Federates certify to the RTI not to emit message until their actual time plus their lookahead. Another important notion is the LITS (Least Incoming Time Stamp (IEEE, 2000)): Federate LITS is a lower bound until which the federate will receive no message, this value is calculated from its GALT and the messages in transit not received yet by the federate (i.e. messages stored in the LRC queue).

HLA FEDEP: The development and execution of HLA federation must follow the HLA FEDEP (Federation Development and Execution Process) which describes a high-level development and execution framework. The FEDEP uses the seven-step process to guide the development of the simulation system through phases of (1) requirements, (2) conceptual modelling, (3) design, (4) software development, (5) integration, (6) execution and (7) evaluation (IEEE, 2003). It has been recently integrated into the more general DSEEP (Distributed Simulation Engineering and Execution Process) framework (IEEE, 2011).

2.4.3.2. HLA for Interoperability

As mentioned, HLA has lot of outstanding features, such as generalized development process: Federation Development and Execution Process (FEDEP); synchronization standard: Runtime Infrastructure Specification; Data Standards; and etc. Because of these features, HLA provides excellent services: capability of achieving interoperability across disparate platform; reusability of simulation models; time management; secure simulation environment; and etc. These valuable services help us to realize the potential ability of HLA in supporting the achievement of enterprise interoperability.

In order to understand how HLA can help the enterprises to establish interoperability, it is necessary to know how to implement interoperability across heterogeneous platforms in HLA simulation. Various HLA components, functionalities and services support the interoperation of distributed simulation, but HLA FOM (Federation Object Model) is playing the primary role among them via following ways:

(1) Unify information exchange: As mentioned, FOM defines the shared vocabulary of a federation, which allows federates to work with one another in a defined manner. This means that the information exchange among the simulation units follows a unique manner, meanwhile, any simulation unit, who accept this manner, could join this communication if they want to. In order to achieve that, the notion of reference FOMs has been used within the defense domain for a long time. The creation of a central, standard FOM for a specific purpose allows components to be created with interoperability in mind (Shanks, 1997).
(2) Overcome platform differences: The FOM also helps overcome platform differences. While the data of different platforms have different representations, HLA concerns this situation very little. All the HLA communication is based on the transmission of an opaque series of bytes. In this case, it needs a mechanism to reconstitute any received information into the useful and understandable format. FOM provide this mechanism. In the HLA simulation, FOM plays as a recipe for the reconstitution of received information into its intended format.

Nowadays, many applications have been developed to implement HLA based interoperability solution in the last decade. (Zacharewicz et al., 2011) has reported several applications which establish interoperability between enterprises IS in various industrial domains. Most of those platforms were designed to exchange data inside the enterprise using distributed simulation for routing and synchronizing the information management using HLA. However, those applications emphasize more on integration. The structure of data exchanged is mostly static. Even HLA allows federation members (federates) to join or leave at run time, but this ability is not fully used. Also the flexibility and compatibility are based on HLA 1.3 or 1516, so they are not very satisfactory regarding at present web 2.0 technologies possibilities and requirements. Regarding these limitations, the methodology presented in this thesis wants to focus on not only reusability of components, but also the compatibility of the platform with web services to be interfaced through the web (i.e. being compliant with HLA 1516 Evolved). Consequently, it needs facilities for community joining and resigning from anywhere on the

web, and making software components interoperable with others thanks to the rapid development life cycle.

2.4.4. Service-oriented architecture (SOA)

2.4.4.1. SOA overview

SOA is an approach to build distributed systems that deliver application functionality as services to end-user applications or to build other services (Colan, 2004). It focuses on the loose coupling of integrated elements to minimize unnecessary dependencies among systems and software elements while maintaining functionality (Gustavson et al., 2005)

The service is the primary element of the SOA infrastructure. It is well defined business/application functionality, which can be reused for different purposes. Recently, the service is mostly represented in web services, but also can be described in other technologies, such as CORBA. (Wiedemann, 2007) introduced a typical SOA-Webservice-orchestration infrastructure as shown in figure 2-12. The major elements of this infrastructure are SOA Service Orchestration and Enterprise Service Bus.

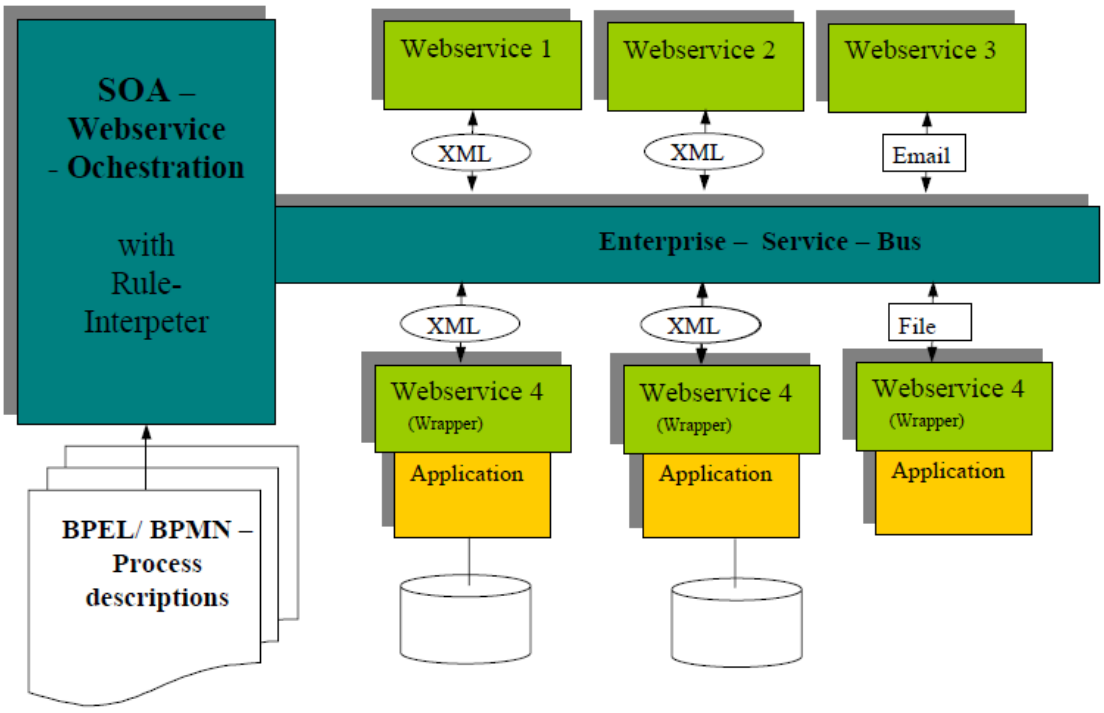

Figure 2-12. OA-Webservice-orchestration infrastructure

- *SOA Service Orchestration* is a control engine that dominates the sequence of service execution. This Service Orchestration Engine has the rule interpreter that can parse the rule defined in BPEL/BPMN (Business Process Execution Language/Business Process Modeling Notation).
- *Enterprise Service Bus* is an intermediator that brings the distributed loosely couple services together. The services can interact with others by sending XML based message, or file message, or email. The individual application can also generate a web service interface as wrapper to communicate with other participants.

As this infrastructure shows, the web services can fully support the service detail definition, so that the SOA solution can be well implemented. The next section will introduce the web services in detail.

2.4.4.2. Web Services

Web Services has achieved a great success in the business domain, which stems from the good characteristics of the technology itself, is widely recognized by enterprises and business organizations and provides effective support for the open source community (Richardson et al., 2007). Microsoft, IBM and Sun and other leading manufacturers as well as Apache and other open source organizations support it. In September 2000, Microsoft, IBM and Ariba published the specification for UDDI (Universal Description, Discovery and Integration). A month later, Microsoft, and IBM jointly published the specification for WSDL (Web Services Description Language) based on XML and SOAP (Simple Object Access Protocol). The SOAP and WSDL specifications have been submitted to the W3C (World Wide Web Consortium). The UDDI community drives the UDDI effort. The W3C and other standardization organizations with active participation in the Web services technology provide for great maturity and popularity of the organization advantage.

The web services technology offers a programming model for creating loosely coupled distributed applications that use open standards. It builds on Internet standards such as HTTP, XML and SOAP. These standards are not associated with any particular vendor, operating system and programming language, which makes Web services platforms with good vendor neutrality. Coarse-grained business functions can also be packaged for the Web service platform and can then be discovered by potential consumers. The figure 2-13 shows the

architecture of web services (Gisolfi, 2001).

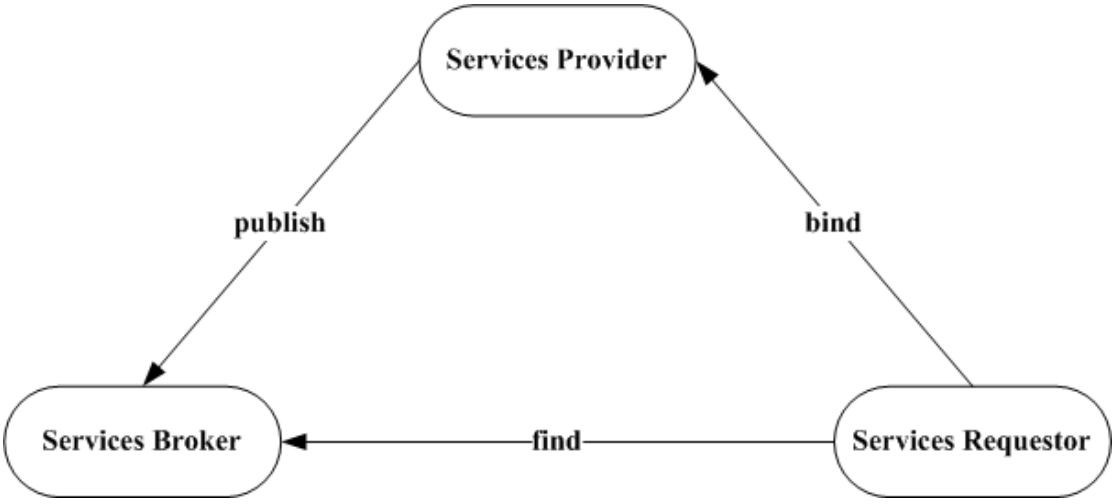

Figure 2-13. Web Services Architecture

Members of web services architecture:

- Service provider: an entity provides an interface for a system that manages a specific set of tasks. It can represent a business entity or a reusable subsystem.
- Service requestor: an entity discovers and invokes other software services in order to accomplish a task or provide a business solution
- Service broker: an entity acts as a repository for the software interfaces published by the service providers.

Process of web services:

- Step 1: The service provider implements the service and describes the service interface, and then publishes the service to the service broker. The service is described in WSDL.
- Step 2: the service requestor discovers the service by UDDI and WSIL, and then obtain the WSDL from the service broker.
- Step 3: the service requestor uses the information obtained from service broker to invoke the service from services provider. After application succeeds, requestor will be bound with provider.

Besides the WSDL, UDDI and WSIL, the cornerstones of this architecture are HTTP for transport, XML for data description, SOAP for invocation.

2.4.4.3. Web Services for Interoperability

As mentioned in the previous section, the primary elements of the web services are HTTP, XML and SOAP. These elements help the web services to overcome the barriers of different programming languages, operating systems, and vendor platforms, so that diverse and distributed applications can interoperate.

- The HTTP protocol provides a protocol and a paradigm for remote invocation across secure boundaries. The HTTP model cares anything about the operations environment in the involved systems.
- XML Web Services provides a common, platform-agnostic medium/technology-agnostic solution, which can support the integration, aggregation, and orchestration of the services across vendors, systems, and organizational boundaries. A set of standards for XML Web Services has been provided by companies such as Microsoft, IBM, BEA, and Sun. These standards contemplate all aspects of enterprise interoperability, such as security, reliability, and transactions. In addition, the Web Services Description Language is XML format. XML helps the WSDL to allow the description of services and their messages regardless of different message formats and different network protocol used for communicating (W3C, 2001a).
- SOAP is a specification for describing an exchange medium between peer systems (W3C, 2007). XML and SOAP fervently support each other. For example, the type of data in SOAP message is identified by XML Schema datatypes and structures, and SOAP helps XML to create Web Services that can provide both synchronous and asynchronous remote invocation. In addition, SOAP XML message over HTTP can travel through the boundaries caused by corporate firewalls.

In sum, through the support of Web Services, SOA can provide a flexible solution for distributed enterprise interoperability. SOA supports component coupling, synchronization, ownership management, and etc. However, some of the functionalities are not fully implemented or hard to implement. For example, the time management performs unstably; Synchronous and asynchronous logic for complicated task might be very complex; and data ownership is not as secure as HLA. In a word, SOA is not the perfect choice for the federated approach proposed in this thesis.

2.4.5. Summary

This section has introduced some simulation and application distribution frameworks, including CORBA, RMI, DIS, ALSP, HLA and SOA. All of them can support distributed system interoperability, but in varying degrees. None of them can fully satisfy the requirement of the federated approach proposed in this thesis as shown in the following table 2-1.

Table 2-1. Comparison CORBA, RMI, DIS, ALSP, HLA and SOA

	CORBA	**RMI**	**DIS**	**ALSP**	**HLA**	**SOA**
Component Coupling	Yes	yes	yes	yes	yes	yes
Time Management	No	no	no	partial	yes	partial
Ownership Management	No	no	no	no	yes	yes
Environment Management	No	no	no	no	yes	yes
Environment Flexibility	partial	partial	no	no	no	yes
data distribution services	No	no	no	no	yes	yes

The federated approach proposed in this thesis requires loose coupling, time control, information authority control, environment control, environment compatibility, and information distribution control. As the table 2-1 shows, none of the technologies reviewed in this section can fully cover these requirements. But, the combination of HLA and SOA seems to be a good choice to achieve the expected goal. Thus, the combination of HLA and SOA has been chosen for this federated approach that will be explained in detail in section 3.4.

2.5. Ontology

2.5.1. Ontology overview

From the philosophical view, ontology is the study of the nature of being, existence or reality in general, as well as of the basic categories of being and their relations. Ontology deals with questions concerning what entities exist or can be said to exist, and how such entities can be grouped, related within a hierarchy, and subdivided according to similarities and differences (Gomez-Perez et al., 2004). During the last decades, ontology has been used in many research domains, such as ontology engineering in computer science and information science. It is a new field, which studies the methods and methodologies for building ontologies: formal representations of a set of concepts within a domain and the relationships between those concepts. Ontologies are used in artificial intelligence, the Semantic Web, software engineering, biomedical informatics, library science, and information architecture as a form of knowledge representation about the world or some part of it (De Nicola et al., 2009).

As ontology is defined as a formal, explicit specification of a shared conceptualization, representation language is essential for description. During the last decades, several ontology representation languages have been developed, such as Ontolingua, RDF(S), and OWL (DAML+OIL).

- Ontolingua is originally an Interlingua for ontology representation and sharing developed by KSL (Knowledge Systems Lab) at Stanford University (Gruber, 1992). It is designed by adding frame-like representation and translation functionalities to KIF (Knowledge Interchange Format) (Genesereth, 1992) which is a logic-based Interlingua for knowledge representation.
- RDF(S) developed by W3C provides a common framework for expressing this information so it can be exchanged between applications without loss of meaning. Since it is a common framework, application designers can leverage the availability of common RDF parsers and processing tools. The ability to exchange information between different applications means that the information may be made available to applications other than those for which it was originally created (W3C, 2004a). RDF has an XML-based syntax (called serialization) which makes it resembles a common XML-based mark up language. But, RDF is different from such a language in that it is a data representation model rather

than a language and that the XML's data model is the nesting structure of information and the frame-like model with slots.

- OWL (DAML+OIL) is also a language developed by W3C (W3C, 2004b). OWL is designed to make it a common language for ontology representation and is based on DAML+OIL (W3C, 2001b). OWL is an extension of RDF Schema and also employs the triple model. Its design principle includes developing a standard language for ontology representation to enable semantic web, and hence extensibility, modifiability and interoperability are given the highest priority. At the same time, it tries to achieve a good trade-off between scalability and expressive power.

2.5.2. Ontology for Interoperability

As mentioned in chapter 1, the lack of a shared understanding among enterprises leads to a poor communication which impacts on levels of misunderstanding, effectiveness of people's cooperation and flaws in enterprise operations. In addition, because eBusiness is involved, the further problems arise on the identification of common, shared objectives, effective exchange of knowledge and services and interoperability among systems, to support value production. Thus, Ontology is increasingly seen as a key factor for enabling interoperability across heterogeneous systems. Ontology can help capturing meaning beyond technical solutions, specifying the representation of documents and modeling semantic content in unambiguous, formal way.

For enterprise interoperability, ontology can aid the business community to agree on a common "vision" of the domain. Ontology can allow business and enterprises to semantically enrich their own models of business documents and services. Ontology can also allow a preventive assessment of the inherent kinship of two enterprises, e.g., potential problems when starting cooperation, by Cross-analysis of Semantic Annotation. Semantic Annotation allows the identification and building of the reconciliation strategies (rules) to cope with divergences (Veltman, 2001).

2.5.3. Ontology mapping approaches

In most of the ontology approaches for enterprise interoperability, ontology mapping is core for combining distributed and heterogeneous ontologies. The existing ontology mapping approaches can be indentified into three kinds of approach: single ontology approach,

multiple ontology approach, and hybrid ontology approach (H.Wache et al., 2001).

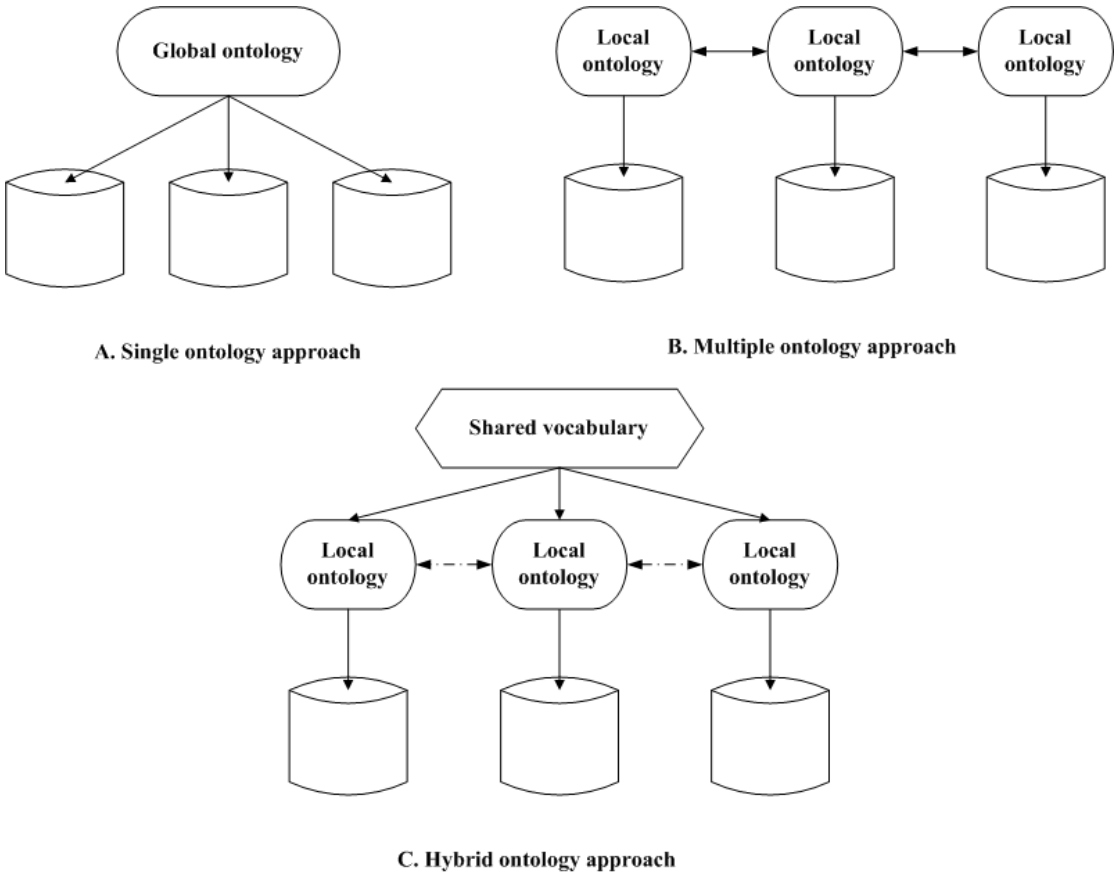

Figure 2-14. ontology mapping approaches

- Single ontology approach has a global ontology which provides a shared vocabulary for the specification of the semantics (as shown in figure 2-14 A). All the information sources have to relate to this global ontology. The global ontology can also be a combination of several specialized ontologies which can be the modularization of a potentially large monolithic ontology. This approach can be applied to integration problems where all information sources to be integrated provide nearly the same view on a domain. The domain differences may cause the difficulties on finding ontology commitment (Gruber, 1995). In addition, if one information source changes something, it will affect the global ontology and the mappings to the other information sources.
- Multiple ontology approach has no common and minimal ontology commitment about global ontology. Each information source is described by its own ontology (as shown in figure 2-14 B). The local ontology of each information source could be developed without respect to other sources or their ontologies — no common ontology with the agreement of all sources is needed. This approach can minimize the affection of change, such as modifications in one single information source or the adding and removing of

sources. However, in reality the lack of common vocabulary makes it extremely hard to compare different source ontologies.

- Hybrid ontology approach is kind of trade-off of the previous two approaches. It absorbs the essences of the previous approaches. As figure 2-14 C shows, this approach has a global ontology as single ontology approach does, but each information source has its own local ontology which is similar to multiple ontology approach. This idea aims at making the ontologies comparison among the coordinated information sources easier. The advantage of this approach is that new sources can easily be added without the need of modification in the mappings or in the shared vocabulary. It also supports the acquisition and evolution of ontologies.

2.5.4. Summary

The ontology can fully support the conceptual enterprise interoperability. As the section 2.5.3 mentioned, there are three ontology mapping approaches that can correspond to the three enterprise interoperability approaches introduced in section 1.2.3.2. Thus, the multiple ontology approach seems to be the approach that meets the demand. Based on the theory of multiple ontology approach, section 3.5 will propose a new ontology approach for enterprise interoperability called “short-lived ontology”.

2.6. Conclusion

This chapter reviewed relevant existing models, methodologies, technologies, and architectures for the development of federated enterprise interoperability. Due to the fact that enterprises require more and more dynamic, complex, and advanced interoperability, these methodologies, technologies, and architectures independently can hardly handle these requirements any more. However they are complementary rather than contradictory. On the basis of those existing approaches, chapter 3 will propose a harmonized and reversible HLA based methodology for developing model driven federated enterprise interoperability. This methodology will creatively combine the excellences of some of these existing methodologies, technologies, and architectures, and propose an innovative way to tackle enterprise interoperability at service and data levels through a federated approach.

Chapter 3. The Harmonized and Reversible HLA based framework and methodology

3.1. Introduction

This chapter presents a harmonized and reversible engineering framework and methodology for developing a HLA based application to set up interoperability rapidly among existing enterprise information systems. This framework and methodology contain the core information of the solution proposed in this doctoral thesis. As mentioned in chapter 2, many architectures, methodologies and technologies can support enterprise interoperability. Such as, MDA and Model reverse engineering can reduce the development cost and complexity, and optimize system/component reusability to enhance system interoperability. HLA and SOA have the potential abilities in supporting the achievement of federated enterprise interoperability in data concern by overcoming the technical barrier. This framework and methodology will draw their benefits to create a novel way to support the development of federated approach of enterprise interoperability. Thus, the methodology presented in this thesis will utilize MDA to formalize the system architecture and relationship among systems, and apply Model reverse engineering to reuse and align different systems/component to initiate enterprise IS interoperability environment, and use the HLA and SOA functionalities as technical support. This framework has three primary concepts that can be separately presented as follows.

Harmonized means that this framework is synthetic, which consists of several techniques. As the framework in figure 3-1 shows, we propose a new five steps development life cycle which aligns MDA and HLA FEDEP. MDA is easy to use and understand, and tightly bounded with Unified Modelling Language, Meta-Object Facility (MOF). It appears to be an appropriate solution to overcome the interoperability barriers, such as the MDI framework mentioned in (Elvesæter et al., 2007). HLA FEDEP is the standard for development and execution of HLA federation. It is quite similar to the waterfall development but with look-back test phase. MDA and HLA FEDEP can be easily aligned, because they have several similar steps. The HLA FEDEP & MDA alignment will be explained in section 3.2. In addition, this framework uses web services to improve the flexibility and compatibility of the HLA. The Web Services allows potential external systems to discover the existing HLA Federation, and then connect to it. Section 3.4 will explain the reason and way of using web services.

Reversible means that this framework uses model reverse engineering technique to discover

part of the models from the legacy system. Model reverse engineering technique aims at avoiding rebuilding the complete legacy system for a new reuse. The objective is to accelerate the development and reduce the cost. As figure 3-1 illustrates, there are two kinds of dotted arrows, which have opposite directions to the five steps development life cycle. These two kinds of arrows represent two different scenarios of model reversal in this framework. Section 3.3 will present the method of using model reverse technique to rapidly develop HLA based interface for achieving federated enterprise interoperability.

HLA means that this framework dedicates to the development of HLA based application. The RTI used in this approach is an open source RTI, poRTIco (poRTIco, 2009). The reason of choosing it is not only because of the software price, but also the objective of initiating a global open framework and receiving comments from contributors who can be interested in this idea. In addition, as mentioned earlier in *Harmonized* part, Web Services will be used to improve the limitation of the traditional HLA. Thus, the HLA approach proposed in this thesis is based on the HLA evolved IEEE 1516^{TM}-2010 standard. This approach will be presented in section 3.4.

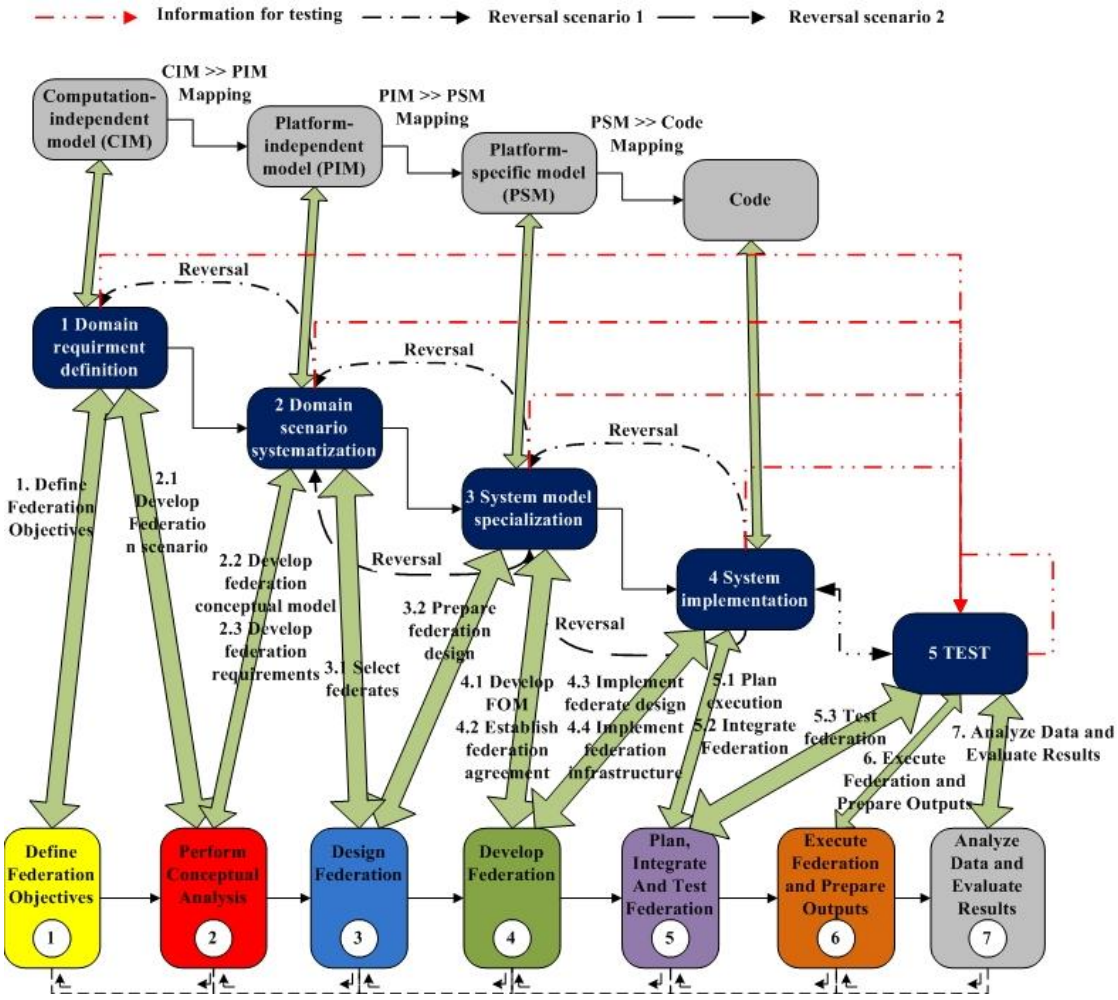

Figure 3-1. Harmonized and reversible development framework for HLA based Application

A schema of the related scenario is shown in figure 3-2. We assume that before enterprises start to launch a cooperative project, all of them have their own information systems. Thus, the goal is to achieve the interoperability among those existing systems in a common project context. The steps of this approach are presented in the following:

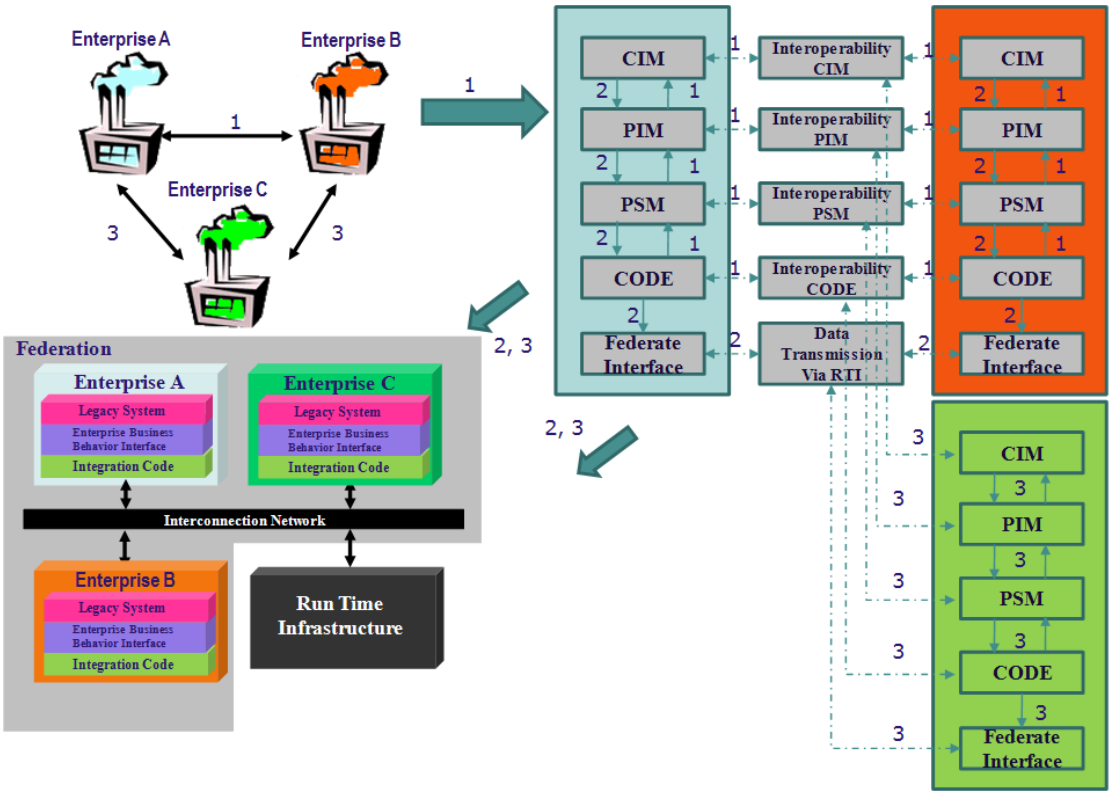

Figure 3-2. Scenario description

- Step 1 (arrows numbered with "1"): model reverse engineering is used to discover the models from the legacy system. The model discovery is guided by the enterprises new requirements and interest. Then, these discovered MDA conceptual models go down again along the alignment of MDA and HLA FEDEP. It means that models are generated from code to PSM then PIM and CIM level. At each level of the MDA models the interoperability problem is tracked according to the principle of the MDI framework.
- Step 2 (arrows numbered with "2"): a test of the final models obtained by model reverse engineering is carried out. After that, the correct models are transformed from CIM to code, and generate a Federate Interface, which can plug into the HLA platform and exchange the information with other companies' information systems via RTI.
- Step 3 (arrows numbered with "3"): if other enterprises want to join this ongoing cooperative project, they also need to follow the step 1 and step 2, to rewind their legacy

systems into MDA conceptual models, and select part of them that can be used for interoperability, then generate the Federate Interfaces, finally, synchronize with other systems.

3.2. The Harmonized HLA&MDA engineering framework

3.2.1. Why harmonized HLA and MDA

As mentioned in the state-of-the-art, HLA is successful in defining how to create a global software execution composed of distributed simulations and software applications. Meanwhile, MDA becomes the standard for promoting the use of models and their transformations to consider and implement different systems. Both of them have made some achievements in their respective domain. However, they both still have drawbacks expected to be improved. This section will explain how HLA and MDA complement each other by using their achievements mentioned in chapter 2, which is conducive for federated enterprise interoperability. This section will firstly further describe HLA and MDA shortfalls in conducing to federated enterprise interoperability. Then, the motivation of the harmonization of HLA and MDA will be given.

As mentioned in section 2.4.3, HLA is a distributed simulation standard that specifies a Federation Development and Execution Process (FEDEP), a synchronization mechanism: Runtime Infrastructure (RTI), and a Data Standards. The FEDEP can standardize the development, which can enhance the model/component reusability for interoperability. RTI can bring different simulation units together for interoperability. The FOM can be seen as the shared template for understanding messages from different simulation systems. However, their ways to achieve interoperability do not fully satisfy the requirement of federated interoperability. The main incongruent factors are as following:

- Tight coupling: the coupling coefficient of HLA is still too high for federated interoperability from some aspects. For example, the models defined by FEDEP are HLA federation specified. These models are hardly be reused if the HLA federation environment changes. There is no formal standard for separating the business logic code and RTI support code, which affects the code reusability (Pokorny et al., 2006).
- Weak compatibility: the FOM-centric approach of HLA and RTI constraints cause this

weak compatibility. As mentioned earlier, FOM is the share data objects for one HLA federation, but it does not mean that it can be recognized by other HLA federations (Granowetter, 1999). Meanwhile, because the federate code must respect to the RTI ambassador services, when attempting to move a federate to another different RTI implementation, the source code modification is definitely necessary (Granowetter, 2004). Thus, it means that the existing work in an existing federation context is hardly be reused in another context, because it was not originally intended.

- Syntax interoperability only: even the FOM helps the data interoperability, but it only defines the syntax for interoperability, not the semantics (StraBburger, 2001). The FOM only defines the structure of the data object. However, only syntax interoperability is not enough for the complex business collaboration. The semantic conflicts will cause a lot of misunderstandings.

In sum, HLA cannot achieve the expected results mentioned in section 1.4.3 alone, such as rapid and dynamic interoperability establishment, and agile environment compatibility.

On the other hand, even MDA has lots of advantages, but some of them only function smoothly in theory, which cause the difficulties of their realisation. The following will present some examples of these difficulties.

- Difficulty of model transformation: as mentioned in section 2.3.4, the MDA model transformation is difficult to control and rarely fully completed. Most of the time, manual intervention and elaboration are needed, which can introduce some unexpected complexities.
- Difficulty of model mapping at the same level: the model mapping at the same level, such as the horizontally mapping mentioned in section 2.3.2, can enhance the model reusability and systems interoperability. However, it is also hard to control and rarely completed. Some additional technologies and methods are needed, such as the ontology approach in the MDI framework.
- Difficulty of representing the behaviour of the complex systems: MDA models standardize the system processes and logics in the CIM level. However, since the model transformation from top to down, the information of the system behaviour is not formalized. Especially, when UML is used to present the system, it cannot fully capture the detail of temporal consideration in the behaviour of the complex systems.

In sum, MDA alone is not the ideal solution for federate approach of enterprise interoperability. The difficulties of its realisation might be the bottleneck of the development of federated approach.

However, it seems that MDA and HLA can help each other in compensating the drawbacks and overcoming the difficulties. The alignment of MDA and HLA can facilitate the construction of simulators and provide the standardized meta-models to this integration (Tolk, 2002) (Parr et al., 2003) (Trbovich et al., 2005). For example, the model levels defined in MDA can help the HLA to define RTI specific or federation specific information in the PSM model, and define the system logics in CIM and PIM model. MDA can also standardize the data objects, which can help the model reverse method to initiate the short-lived ontology glossary (this will be explained in section 3.5). On the other hand, the HLA specific constraints and given context can strictly guide the MDA model transformation.

3.2.2. The proposed Framework

3.2.2.1. Overview of the framework

This section introduces a development lifecycle based on HLA FEDEP and MDA under the five steps engineering framework (as shown in figure 3-3). This new framework has been reported in (Tu et al., 2012b), which aims at adopting the strong points from both HLA FEDEP and MDA. This framework proposes some proper key phases for reusing existing software to achieve a rapid redevelopment of a HLA based system of systems. The task of each phase is as follows:

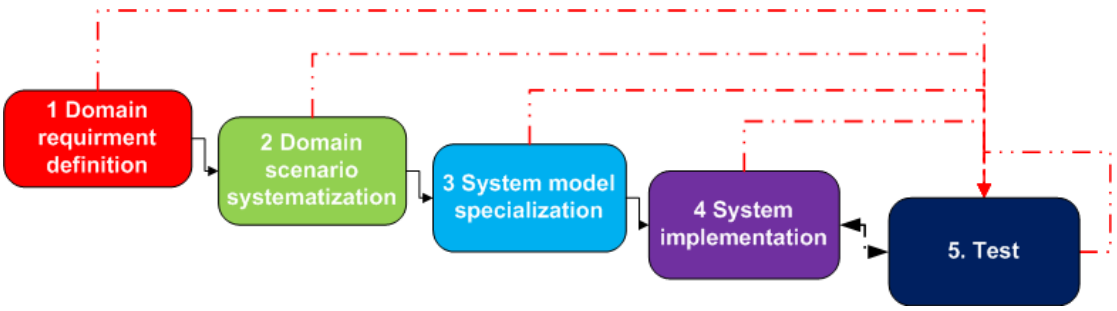

Figure 3-3. Harmonization of MDA and HLA FEDEP

- **Phase 1: Domain requirement definition**. Its main task is to collect sufficient and clear requirements from the participants in order to define the objective of the system, to describe the environment of the system, the scenario of the system and the business process. All these definitions and descriptions have to be reasonable and understandable for all of the participants. The CIM level of MDA has a task that is similar to both the Define Federation Objectives and the Develop Federation scenario together in HLA FEDEP. As a result, their alignment in this phase is to convert the user requirements that are textual based, into a more visual and formal model, such as the UML use case to derive the federation requirement.
- **Phase 2: Domain scenario systematization**. Its main task is to refine the domain scenario and the business process captured in the first phase. It identifies and describes the entities involved in the scenario and business process. Then, it defines the relationships among entities and their behaviours, events for each entity, etc. This phase integrates the PIM level in MDA, which describes the operation of the system but doesn't address the detail platform information yet. It also integrates steps of the Perform Conceptual Analysis, Develop Federation Requirements and Select Federates in HLA FEDEP. In addition, it defines and selects general participants of the federation, describes their relationship, behaviours and event in general.
- **Phase 3: System model specialization**. In this phase, according to the technique chosen and the platform selected, the system needs to be refined, for instance, to refine federation and federate structure, to allocate functions and attributes, etc. Detailed design is carried out at this time. This phase integrates the following parts in MDA and FEDEP.
    - The PSM level in MDA that is in the form of software and hardware manuals or even in an architect's head, is based on detailed platform models, for example, models expressed in UML and OCL[4] (Object Constraint Language), or UML, and stored in a MOF compliant repository.
    - The Prepare federation design, Prepare plan, Develop FOM, and Establish federation agreement in FEDEP produce federate responsibilities, federation architecture, supporting tools, integration plan, VV&A [5] (Verification, Validation and Accreditation) plan, FOM, FED (Federation Execution Data) /FDD (Federation Object Model Document Data) and time management, date management, distribution agreements, etc.

[4] Object Constraint Language is a declarative language for describing rules that apply to Unified Modeling Language (UML) models developed at IBM and now part of the UML standard (OMG, 2006).
[5] VV&A is to assure development of correct and valid simulations and to provide simulation users with sufficient information to determine if the simulation can meet their needs (DoD DMSO, 2006).

- **Phase 4: System Implementation**. Its task is to transfer the specific system model into code, to create the executable federation and executable federate. At this level, MDA has various transformation techniques from model to code. In the FEDEP, Implement Federate designs provide modified and/or new federates and their supporting databases. Implement Federation Infrastructure provides implemented federation infrastructure and modified RTI initialization data. Plan Execution and Integrate Federation provide execution environment description and integrated federation.
- **Phase 5: Test**. Throughout the previous steps of the MDA and HLA FEDEP alignment process, testing is essential to ensure fidelity of the models. Testing phase includes the Test Federation, Execute Federation and Prepare Outputs, and Analyze Data and Evaluate Results in HLA FEDEP. Meanwhile, it also refers to the outputs from the previous steps, such as the original user requirement in the first step, and federation test criteria from second phase.

3.2.2.2. Harmonized single federate structure

Due to the purposes of harmonization of HLA and MDA, this harmonization process will generate a specific structure of HLA federate. This structure can be considered as a converter. The federate has two parts as illustrated in figure 3-4, one is the Adapter and another is the Plug-in.

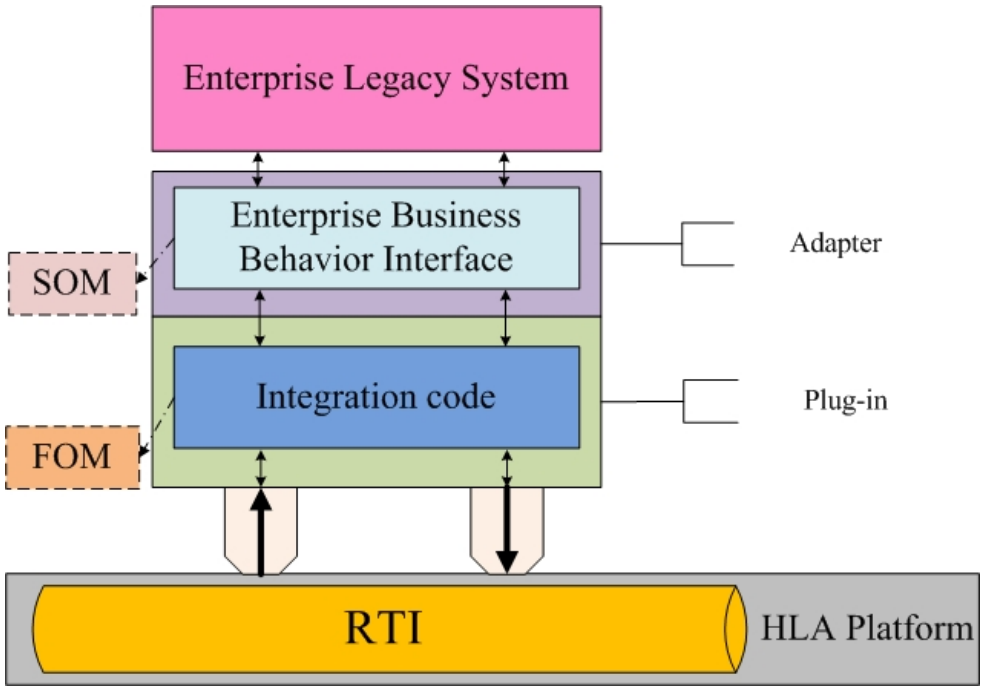

Figure 3-4. Harmonized federate structure

- The Adapter is an Enterprise Business Behaviour Interface that links to the enterprise legacy system. As the name shows, the functionality of the adapter is to overcome the gaps between enterprise legacy system and the HLA environment. As mentioned, the objective of this approach is to make the enterprise capable to cater for the cooperation without changing its legacy system and business mode. Thus, the duty of the enterprise business behaviour interface is to adapt to different legacy systems of different enterprises by implementing specific strategies and algorithms for different enterprises. In addition, it will also accomplish the cipher mission. From HLA point of view, the adapter concerns only the local federate, and keeps it independent from any RTI modification. The adapter makes the federate different from others, then play different roles in simulation. The code generation of adapter is the mission of model reverse method, which will be explained later.
- The Plug-in is an Integration code, which manages the interactions between the enterprise business behaviour interface and the RTI, providing an RTI independent API to the enterprise business behaviour interface, and a simulation independent API to the RTI services. The integration code is the common component for all federates of the existing coordinators and also the reusable components for the future coordinators. In addition, the integration code makes the federate capable to detect and adapt to the environment changes automatically. It maintains the communication connections, cooperation requests, and withdraw announcement. The enterprise will ignore these trivial and technical related operations, but waiting for the message from integration code.

3.2.3. Summary

Section 3.2.2.1 has presented an engineering framework of harmonization of MDA and HLA FEDEP with a five steps development lifecycle. The purposes of the harmonization of MDA and HLA FEDEP are:

1) To reduce the complexity of the HLA based application development by modelling and standardizing it.
2) To enhance the reusability by merging both MDA and HLA features for promoting reusability.
3) To ensure that the model reverse process can follow the ADM (Architecture Driven

Model) way.

Section 3.2.2.2 has introduced a harmonized single federate structure, which divides the federate into two abstract parts. The objective of these abstractions is to ensure that the enterprise business behaviour remains decoupled from RTI services. After the harmonization, all federates will have the same integration code but different Enterprise Business Behaviour Interfaces. Meanwhile, any simulation related services required by the enterprise business behaviour interface are accessed via the integration code, rather than through direct interaction with the RTI.

3.3. Model Reverse method

3.3.1. Why model reverse

As section 1.4.3 mentioned, the expected interoperability environment must allow rapid and dynamic interoperability establishment, agile environment compatibility, easy connection, and collaboration environment control. In other words, this interoperability environment intends to be the "plug and play" environment. The previous section has proposed a harmonized single federate structure, which consists of an "Adapter" - Enterprise Business Behaviour Interface and a "Plug-in" - an integration code. The significance of this structure is the platform independence and reusability by encapsulating the Enterprise Business Behaviour code and RTI specific code. In addition, it is the elaborative design for implementing "plug and play" environment.

Since the expected interoperability environment must support rapid and dynamic interoperability establishment, it is not desired to redevelop the entire existing enterprise systems. In this case, the existing systems will be retained and used for interoperation. Thus, an agile interface – "Adapter" has been designed as a wrapper to allow the existing systems to connect to the interoperability environment seamlessly. This "Adapter" is a lightweight component, which is generated based on the model information reversed from the legacy systems.

The model reverse method introduced in this section aims at obtaining the static models of legacy systems, and also the dynamic models (behaviour models). Meanwhile, this method must follow the development lifecycle of the harmonized HLA&MDA engineering

framework proposed in previous section. Therefore, the model information obtained by this model reverse method will help the "Adapter" generation for rapid and dynamic interoperability establishment and easy connection, and also the "Plug-in" generation for agile environment compatibility and environment management. In addition, this model information will also be used to generate HLA federation web service that will be introduced in section 3.4, and to initialize the "short-lived ontology" glossary that will be introduced in section 3.5.

3.3.2. The proposed model reverse method

This section describes a model reverse method with two different scenarios constraints. These two scenarios are presented as two arrows around the five steps life cycle as shown in figure 3-5. The reversal method will re-characterize the legacy system in order to capitalize on the information and functions of the existing system, and reuse them in a new HLA compliant system. The expected output of this method is the HLA FOM (Federation Object Model) file and HLA federate code block. These outputs will assist to HLA FEDEP / MDA alignment mentioned in section 3.2, to fully achieve rapid development of federation and/or federate based on the legacy IT systems.

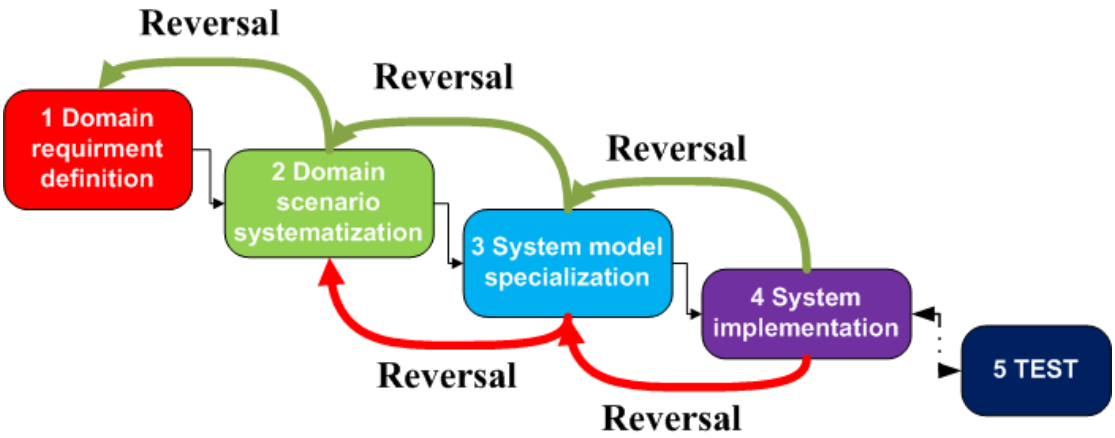

Figure 3-5. Model Reverse Process Scenarios

The difference of the two scenarios constraints is the proportion of model reversal. According to the existence of HLA federation, the reversal process will stop at different steps of harmonized lifecycle mentioned in previous section.

- First scenario (shown as the green "reversal" arrows in figure 3-5): If the HLA federation has not been created yet, the model reversal process needs to start from the code of the

legacy information systems to the first definition phase (domain requirement definition).

- Second scenario (shown as the red “reversal” arrows in figure 3-5): If the HLA federation has already been created, the reversal can stop at the second phase (Domain scenario systematization). It will only reuse the model of the existing federation to create the model for the federate related to the legacy system of new participant.

All the models coming from this reversal process are used to produce a federation and federate rapid development template.

As mentioned, the purpose of this method is to generate HLA FOM and HLA federate code blocks. Since these two outputs have essential differences and subtle relevance, the process of this method is decomposed into the following steps (shown in figure 3-6):

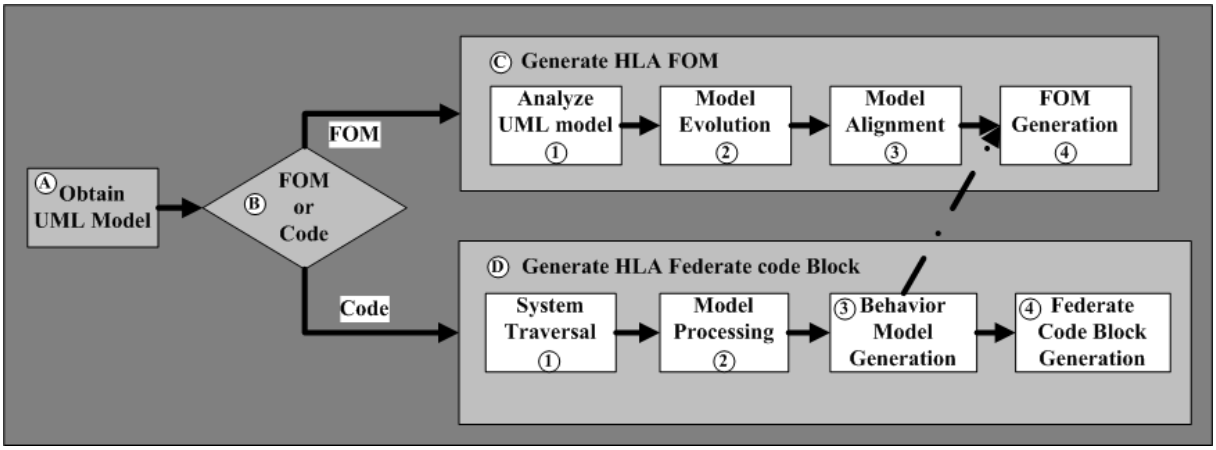

Figure 3-6. Model Reverse Process

A. This process will firstly start from obtainment of UML model by using adapted MoDisco (for Model Discovery) principle.

B. Model Discrimination: The UML models obtained from step A will be used for HLA relevant code generation, HLA FOM and HLA Federate code Block. The information of HLA FOM concerns more the object and interaction that represent the information exchanged with other federates. The HLA Federate code Block is located in Enterprise Business Behaviour Interface shown in the previous section, which contains enterprise business logic. Because HLA FOM and HLA Federate code Block are entirely different model transformation targets, two different processes of model transformation will be carried out based on the UML models reversed from existing systems.

C. Generation of HLA FOM:

C.1. Firstly, this sub-process starts from analysis of UML model that aims at simplifying complex model information and obtain useful and meaningful class models and attributes.

C.2. Secondly, this sub-process commences the categorization of the collaborated enterprises to help model evolution which intends to simplify the model alignment and ensure the quality of aligned models.

C.3. Thirdly, this sub-process begins to find the similar models in model categorization generated by model evolution.

C.4. Finally, based on the aligned models, this sub-process generates the HLA FOM file.

D. Generate HLA Federate code Block

D.1. Firstly, this sub-process starts from system traversal that aims at discovering the possible execution paths of the existing system. The nodes of paths are the simplified UML class models from step C.1. They are linked by function call on the paths.

D.2. Secondly, the possible paths detected by step D.1 need to be recomposed into one or more directed graphs[6]. And then, these directed graphs needs to be simplified by transitive reduction.

D.3. Thirdly, the reduced directed graphs will be transformed into state machine diagrams. These state machine diagrams can be transformed into other models, such as BPMN[7], DEVS[8] model, to represent the business/simulation logic in detail. They can also be used to represent the system behavior directly. The method introduced in this section chooses the latter solution, because of the limitation of the research time.

D.4. Finally, the state machine diagrams will guide the code generation of business logic control module. Afterwards, business logic control module will be combined with RTI specific code block, so that the federate code block is finally generated.

[6] In mathematics, a directed graph or digraph is a graph, or set of nodes connected by edges, where the edges have a direction associated with them (Biggs et al., 1986).

[7] Business Process Model and Notation (BPMN) is a graphical representation for specifying business processes in a business process model (OMG, 2011b).

[8] DEVS abbreviating Discrete Event System Specification is a modular and hierarchical formalism for modeling and analyzing general systems that can be discrete event systems which might be described by state transition tables, and continuous state systems which might be described by differential equations and hybrid continuous state and discrete event systems (Zeigler, 1984).

3.3.2.1. Obtain model information

Model Reversal Structure

A schema of model reversal structure can be seen in figure 3-7. This illustration is based on the MoDisco approach. In MoDisco principle (Jouault et al., 2009), a model (Mi) in the modeling world is a representation of a system in the real world and the nature of the model (Mi) is defined by its meta-model (MMi). It means that model Mi conforms to its meta-model MMi, and every step is guided by a meta-model. The very first step of a model discovery process is always to define the meta-model corresponding to the models that are required to be discovered. Then, the second step is about creating one or many discoverers, which is illustrated in the middle of figure 3-7. These discoverers extract necessary information from the system in order to build a model conforming to the previously defined meta-model. The way to create these discoverers is often manual but can also be semi-automatic.

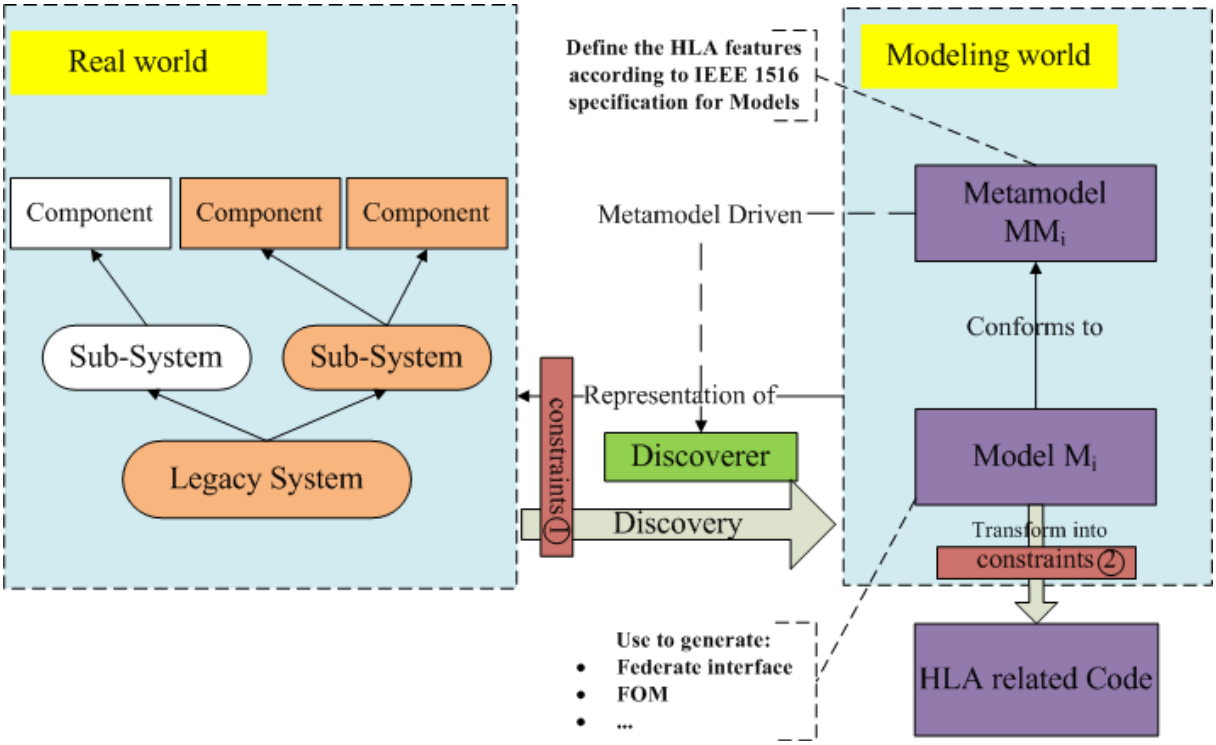

Figure 3-7. a schema of model reversal structure

In addition, in order to adapt MoDisco principle to the federated approach proposed in this thesis, the "constraints" will be added onto the "discoverer" (the green box illustrated in figure 3-7). The "constraints" will be put before the "discoverer" (as the constraint ① shown in the figure 3-7, before system reversal happens) and after the "discoverer" (as the constraint ② shown in the figure 3-7, before the target model transformation happens) according to the

following specification:

- "Constraints ①": these constraints are used to simplify and configure the model reverse process.
  - Simplify the model reverse process: as known, the legacy system consists of lots of diverse sub-systems, which are always based on various kinds of platforms and techniques, thus it is big and only partially useful in the particular context. The reversal of the whole legacy system would be extremely huge and complicated, which departs from the objective. As a result, "constraints ①" aims at specifying the target source, which means that the bound of model reversal must be defined before start to reverse. The boundary must also be defined based on each enterprise's confidential information. This boundary specification will be recorded as a configuration file which can be read by discoverers.
  - Configure the model reverse process: the model reverse application designed for enterprise interoperability will be applied on various enterprise systems. Thus, it must consider interoperability constraint based on the specific scenario, such as participants' relationship, collaboration agreement, work flow, and etc. Before to execute model reverse application on different systems, the model reverse process must be configured based on the interoperability constraint. This configuration will be refined in the part *model evolution* of section 3.3.2.2.
- "Constraints ②": these constraints are used to filter the model information obtained from model reverse tools, and guide the model transformation according to the specific requirements, such as language specific, platform specific, and so on.
  - Model information filter: this first functionality of "constraints ②" can be considered as a "filter". Based on the current model reverse engineering technology, most of the model reverse tools can obtain mass information of models. According to the different motivations, the model information might be useful or useless. The reverse method proposed in this section concerns only the system handles that provide the interfaces for data input and output. In addition, it is very complicated and dangerous to make an interoperability decision based on the complex information. Thus, it is necessary to wipe off the unnecessary information and retain only the valuable information in the considered context. The "filter" will be refined in the part *analyze UML model* and *model alignment* of section 3.3.2.2.
  - Model transformation guide: according to the ongoing research, none of the software

tools can fully reverse a legacy system from code to model. Some of the tools can rewind the code to static model without the dynamic one, and some of them can only discover the data model from database. Meanwhile, as mentioned in section 2.3, the model transformation also causes the loss of information. Therefore, the obtained model information cannot be used directly for interoperability, it must be complemented. For example, in order to develop HLA components that interface with legacy IS, the behaviour models of the actions on the data also need to be discovered for implementing the mechanism for data access, the periodicity of update and the sequences of modifications accepted. Thus, the guider must complement the obtained information in order to generate the required models. This complementary guider will be refined in section 3.3.2.3. The part *generate HLA FOM* of section 3.3.2.2 describes a language and platform specific constraint.

Model conversion

MoDisco tool is an Eclipse GMT[9]component for model-driven reverse engineering. MoDisco tool has two existing discoverers, one is JavaDiscoverer which discovers KDM models from java sources or java models, and another one is CSharpDiscoverer which discovers from C# models. Figure 3-8 illustrates the KDM models which are discovered by JavaDiscoverer.

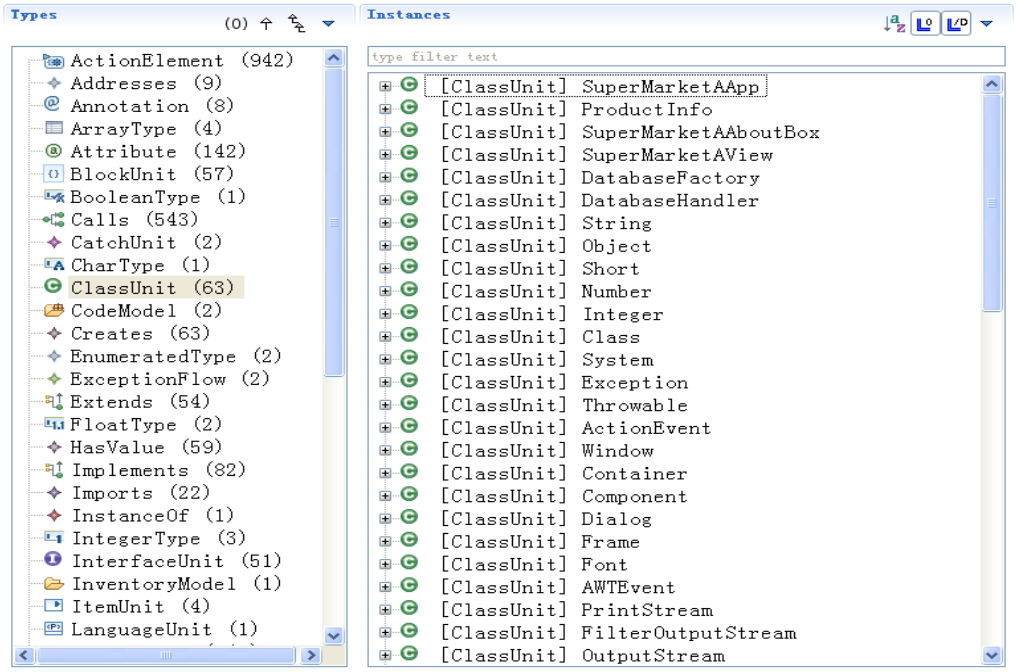

Figure 3-8. KDM models discovered by JavaDiscoverer

[9] GMT is Generative Modeling Technologies. The Eclipse GMT project is to produce a set of prototypes in the area of Model Driven Engineering (MDE).

As shown in figure 3-8, there are many KDM models listed in the left model trees, such as ClassUnit, LanguageUnit, ParmeterUnit and etc. Those models will be converted into UML models later by "KDM to UML Converter". This conversion must follow the mapping listed in table 3-1.

Table 3-1. KDM to UML mapping

KDM	UML
LanguageUnit	Package
CodeModel	Model
CodeAssembly	Model
Package	Package
ClassUnit	Class
InterfaceUnit	Interface
MethodUnit	Operation
ParameterUnit	Parameter
Extends, Implements	Generalization
PrimitiveType	PrimitiveType
MemberUnit	Property, Association

The "KDM to UML converter" is mainly implemented by an ATL[10] model-to-model transformation taking as input a model conforming to the KDM meta-model and producing as output a model conforming to the KDM models into UML meta-model. After the conversion which follows the mapping showed in table 3-1, the UML models will be generated as the figure 3-9 shows. These converted UML models include Packages, Interfaces, Classes, and also the properties and operations of classes and associations and dependencies among the classes.

[10] ATL is ATL Transformation Language that is a model transformation language and toolkit. ATL provides ways to produce a set of target models from a set of source models.

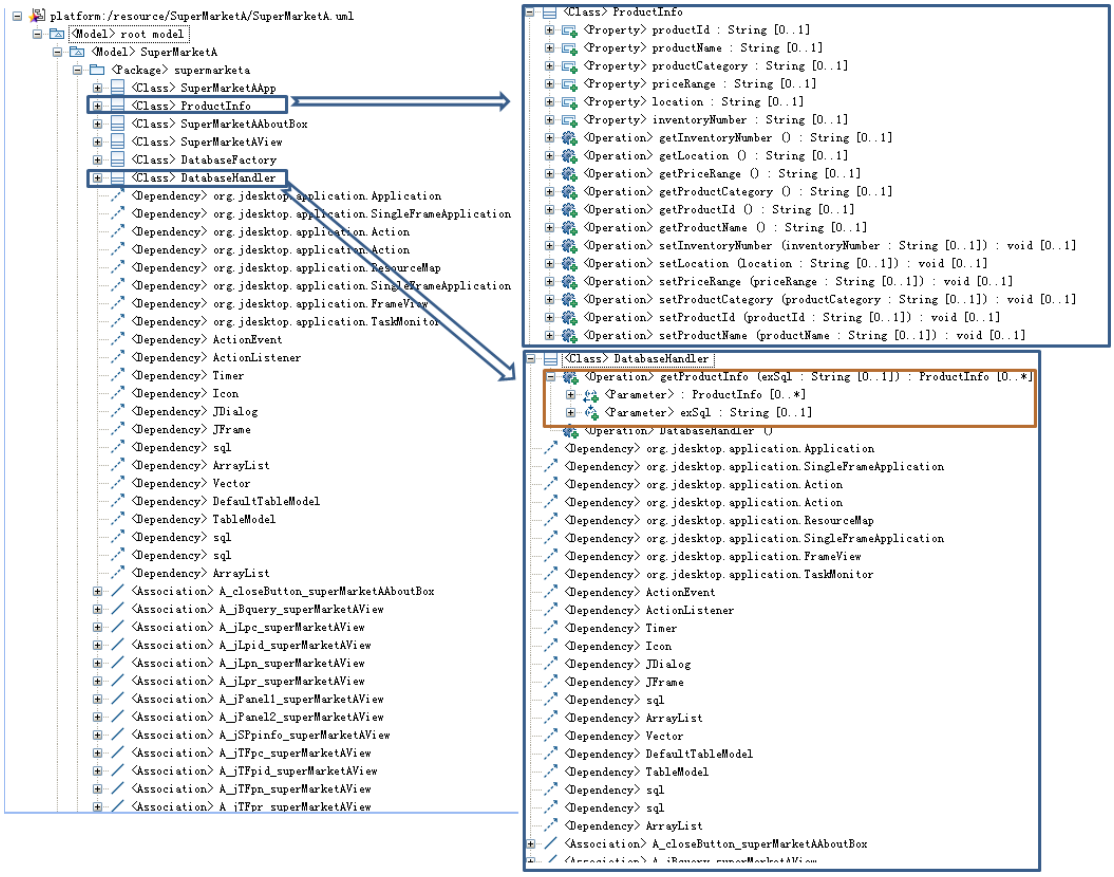

Figure 3-9. UML Model

3.3.2.2. Generate HLA FOM

As mentioned earlier, HLA FOM uses the object-oriented method to define the structure of all information that is available to be exchanged among federates. In HLA simulation, FOM plays as shared concepts between all federates of the HLA federation, which represents the established consensus of the collaborative enterprises. In this case, after the obtainment of UML models of different enterprises shown in the previous section, it is imperative to simplify and unify the complex information, and then generate the HLA FOM.

Analyze UML model

As shown in figure 3-9, the generated UML models contain lots of information, including unnecessary elements for one particular HLA FOM generation. Thus, in order to avoid the ineffectual cost, it is necessary to simplify the models by eliminating the information of redundant and unused classes.

HLA FOM contains object class which represents object-oriented data shared in the federation

that persists during the run time, and interaction class data which are just sent and received information between federates. Thus, the task of HLA FOM generation is to extract these two classes from the reversed UML model. The class diagram is very helpful for generating object class, but the dependency and association among classes might not be very useful. The functions and associations may help the generation of interaction class, but not of all them are helpful. In addition, not all the classes are interesting to be used. In summary, this step will select useful classes, and associations. The interaction class generation also needs the supports from the behaviour models reversal that will be explained in the coming section.

Model Evolution

After previous step, the prerequisite UML models of each enterprise are ready for model alignment. However, sometimes, many cooperative enterprises are involved. Thus, the following questions come out altogether: shall we align all the models once or separately? If separately, who should be aligned first (i.e. defining a reference), who should be the next one? How to keep the best feature and eliminate trash? How to limit the information loss during the model alignment? In this case, the definition quoted from human evolution can help to illustrate these questions. The hominid speciation started 15 million years ago. After that, human inherits the features of ancestors and select them generation after generation. Now, human has evolved into an intelligent species. In this evolution process, human keeps the good genes which help human in adapting to the law of nature and survive. Some species such as dinosaur and mammoth died and disappeared, because they retain the genes which obey the law of nature. These are two kinds of evolution result, either prosperity or extinction. Without doubt, the model evolution in this section must be the good one. In this way, the objective of this model evolution is to maintain the model information which conforms to the law of enterprise interoperability and enterprise requirements.

As mentioned earlier, UML models have been obtained for each single enterprise after previous steps. From the set theory point of view, each single enterprise can be considered as a set which contains UML models as elements. Thus, the set theory can help the model evolution and model alignment. In set theory, the theory of composition of relations (Wang, 2000) defines that if the R1 is a binary relation between set A and set B; the R2 is a binary relation between set B and set C; the R3 is a binary relation between set C and set D, then

$$(R1 \bullet R2) \bullet R3 = R1 \bullet (R2 \bullet R3) \quad (1)$$

$\bullet$ represents relation composition

If we consider the model alignment as a relation (because model alignment is the process of finding similar UML models among the enterprises, it can be considered as a similarity relation), then we can answer the question "who should be aligned first, who should be the next". In other words, it is possible to categorize the cooperative enterprises for model alignment which is a process to maintain the useful information for the most suitable model evolution. The principle of the categorization is to start from enterprises that are in similar or relevant domains, or the closest partners. In this case, the cooperative enterprises will be categorized into several sets. If it is necessary, the categorization of enterprises could be taken place in the smaller set again based on the principle. When the sets of the enterprises are ready, model alignment can be carried out in each set. After that, the categorization process will be executed on the posterities created by each set's model alignment, then model alignment again. So, the model evolution is an iterative process as the human evolution. It passes through many generations, and finally obtains a set of brilliant enough models which satisfies the law of enterprise interoperability and enterprise cooperation requirements.

Model Alignment

Model alignment is carried out in a union of enterprise created by enterprise categorization. This union contains many UML models. The task of model alignment is to find the similar models and unify the information of these models. As mentioned, the UML model used in this phase is the class diagram which consists of attributes and functions. To generate the object class of HLA FOM, we will use the attributes of the class diagram and also use the set theory for theoretical support. Each class can be considered as a set, and the attributes can be considered as the set elements. In this case, the similarity of class can be treated as set similarity which concerns the numbers of similar elements. According to the Jaccard Similarity[11] of set similarity (Jaccard, 1912), the set similarity is defined as follows,

If S and T are two sets that contain limited quantity of elements, then:

$$\text{The similarity of set S and T} = |S \cap T| \ / \ |S \cup T| \quad (2)$$

For example, as figure 3-10 shown, set S contains 8 elements, and set T contains 9 elements. Meanwhile, the number of the elements inside the intersection of set S and T ($|S \cap T|$) is 6, and the number of the elements of the union of set S and T ($|S \cup T|$) is 11. Then, the similarity

[11] Jaccard Similarity is defined as the quotient between the intersection and the union of the pair wise compared variables among two objects.

of set S and T is $\frac{6}{11}$.

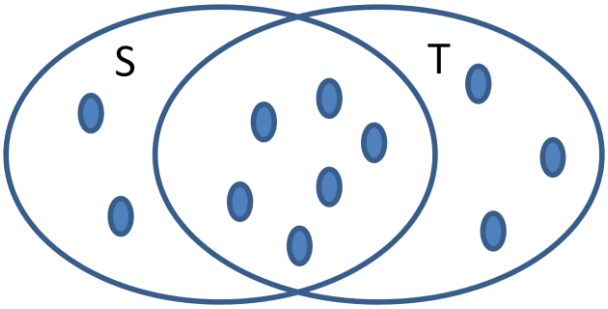

Figure 3-10. Jaccard Similarity of set similarity

Refer to this definition, the class similarity equals to:

$$\frac{\text{the number of similar attributes}}{\text{the number of similar attributes} + \text{ the number of dissimilar attributes}} \quad (3)$$

As one categorization of models can be the models from several different enterprises, the model similarity will not be carried out only on one pair but also on a set. Thus, similarity transmission can be helpful to discover similar pairs automatically and avoid some duplicate actions. Before to describe what the similarity transmission is, it is better to firstly refer to the relation transmission theory of the set theory (Wang, 2000). In this theory, the transitive relation is defined as following:

R is a binary relation of Set X, then if any elements of X like $x, y, z \in X$ have the feature that if xRy (x and y have R relation) and yRz, then xRz, then relation R is transitive. Vice versa, if R is a transitive relation of Set X, then any $x, y, z \in X$, if xRy, yRz, then xRz. For example, the common transitive relations are equivalent relation, descendant relation, and etc.

Because the similar relation is not always a transitive relation, the similarity transmission mentioned here is an intellective detect and determine process of the relation transmission. For example, Class A, B, C belongs to the same model union. If class A is similar to class B, and class B is similar to class C, then the similarity transmission process will detect the possibility of to transmit similar relation from class A to class C, and decide whether this possibility can be worked out.

In order to implement the model alignment among the models of model union, the similarity

transmission will be operated on a relation matrix. The relation matrix is a way of explaining transitive relation definition. It defines that if Relation R is transitive, then if Matrix M has M_{ij} = 1 (it means that i and j has R relation) and M_{jk} =1, then M_{ik} = 1, as shown in the table A of the figure 3-11. And so, in a similar manner, all classes inside the model union will be placed on the matrix columns and rows. In other words, each column and row of the relation matrix represents a class and the value of the M_{ij} represents the similarity value of class I and J. And then, as the table B of figure 3-11 shows, for example, if M_{bi} = 80% (it means that class B and I are 80% similar.) and M_{ij} = 70%, then the similarity transmission process will detect the question mark on M_{bj} which means that class B and J are possible to be similar. If so, the similarity transmission process will determine the value for M_{bj} automatically based on the value of M_{bi} and M_{ij}. Otherwise, no value will be assigned to M_{bj}, which means that the similar relation will not be transmitted from class B to class J, so they are not similar.

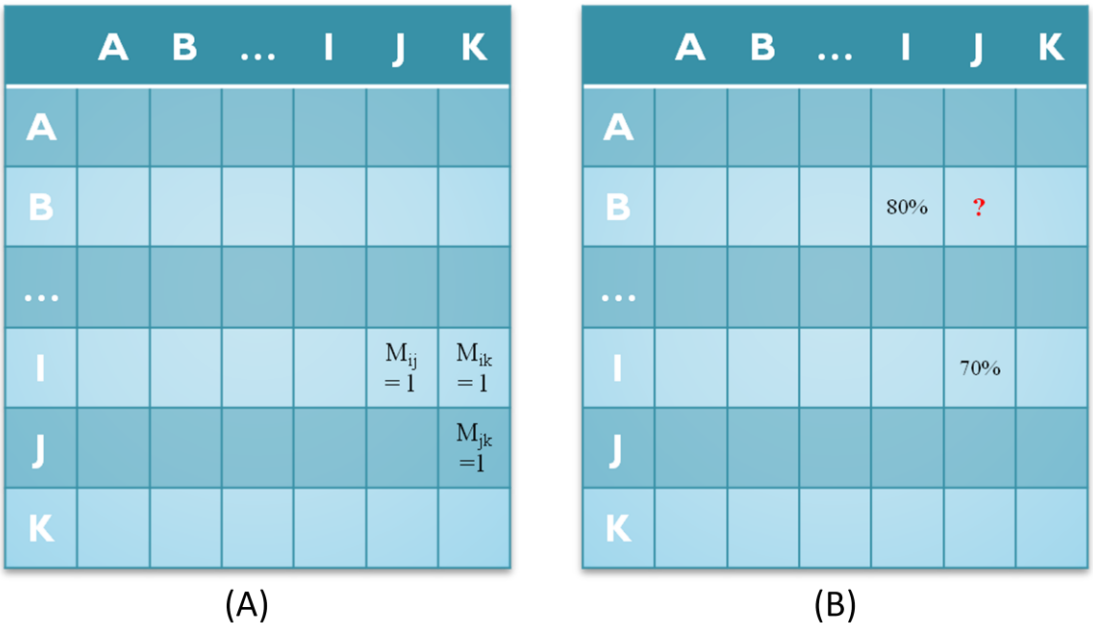

Figure 3-11. Relation Matrix

To carry out this strategy, the model alignment will follow the process shown in figure 3-12.

- Step 1 defines the similarity for one pair of classes based on the evolved formula (3) of Jaccard Similarity of set similarity.
- Step 2 discovers whether there is a possibility to transmit the similarity on the matrix. If yes, go to step 4, else go to step 3.
- Step 3 checks whether there is any blank cell that is required to be assigned with the similarity value on the matrix? If yes, go back to step 1, else finish model alignment.

- Step 4 calculates the transmission threshold value based on the defined similarity values of two pairs and the expected transmission similarity. (This step will be detail in the coming part)
- Step 5 decides whether this possibility of similarity transmission is available or not based on the result of step 4? If yes, go to step 6, else go back to step 3.
- Step 6 transmits the similar relation onto the new pair and assigns the similarity value to it.

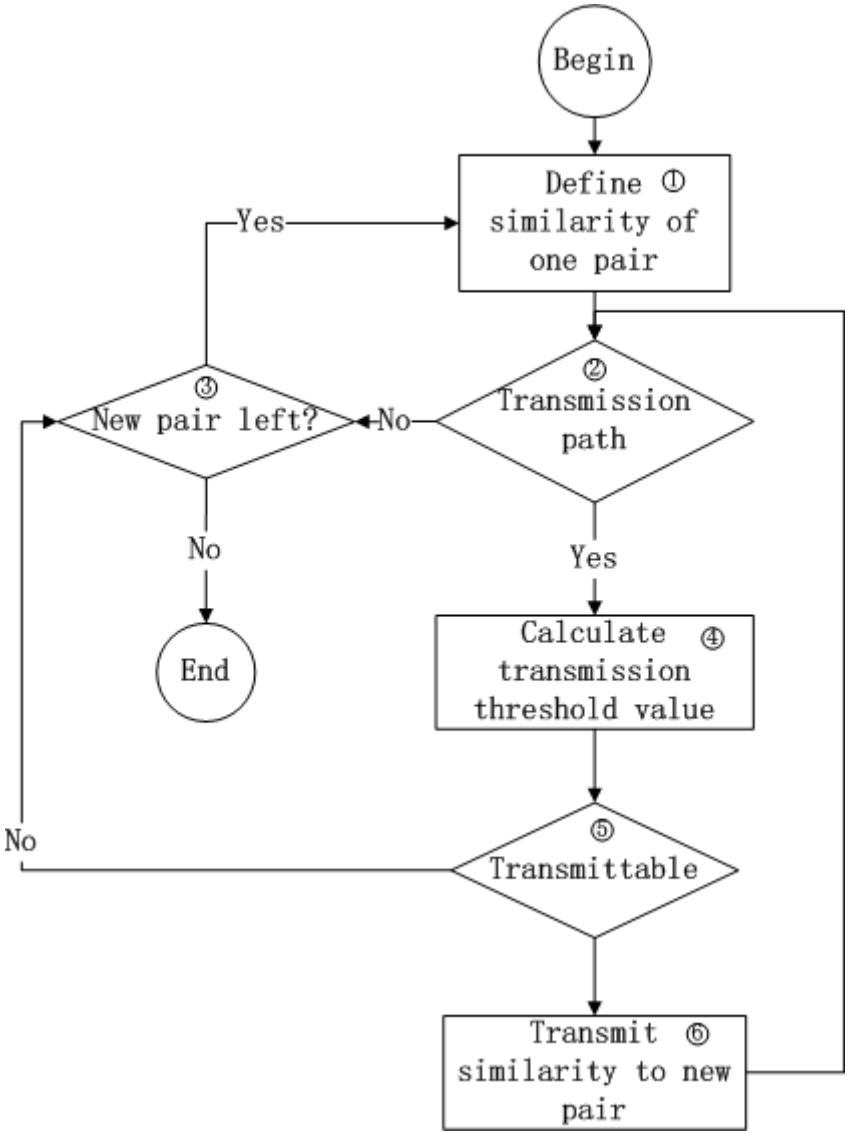

Figure 3-12. Model similarity transmission process

The step 4 of this process is the core phase which decides the tendency of the similarity transmission. This paragraph is going to give an example to explain how to calculate the Transmission Threshold Value (TTV). As the segment A of the figure 3-13 shows, there are

two similar classes, class S and class T. We assume that there is a transitive candidate class G, because it is similar to class T. We symbolize the similarity of S and T as X and the similarity of T and G as Y. We assume that the ETS (Expected Transmission Similarity) of S and G is 70% (defined by user). In addition, we assume that all the classes have the same number of attributes (simplify result from Analyze UML model phase). And then, there are three possibilities need to be considered as the segment B, C and D of figure 3-13 shows.

- Segment B: if the intersection of class T and class G belongs to or equals to the intersection of class S and class T, and none of the elements of complement class S and T exists in the intersection of class S and G, symbolized as $T \cap G \subseteq S \cap T$ and $\forall x \in S - T, x \notin S \cap G$, then it is clear that the similarity of class S and class G equals to the similarity of class T and class G.
- Segment C: if the intersection of class T and class G belongs to the intersection of class S and class T and some of the elements of complement class S and T exist in the intersection of class S and G, symbolized as $T \cap G \subset S \cap T$ and $\exists x \in S - T, x \in S \cap G$, then the size of the intersection of class S and G is easy to identify by counting the number of x and the $|T \cup G|$.
- Segment D: if the intersection of class T and class G belongs to the intersection of class S and class T and none of the elements of complement class S and T exists in the intersection of class S and G, symbolized as $T \cap G \subset S \cap T$ and $\forall x \in S - T, x \notin S \cap G$, then the similarity of S and G is hard to tell. Thus, if this possibility wants to be transitive, then the following calculation can help to obtain the TTV.

  $ETS \leq (|S \cap T| - |T - G|)/(|S \cup T| - |G - T|)$

  $\because$ ETS = 70%

  $\therefore$ TTV = (17 – 3X) / (3 + 23X) $X \in (0.77, 1)$ X represents the similarity of S and T.

  Then TTV $\in$ (0.54, 0.71)

  Thus, after calculation, if the similarity of class T and G is beyond the TTV, then this possibility can be feasible.

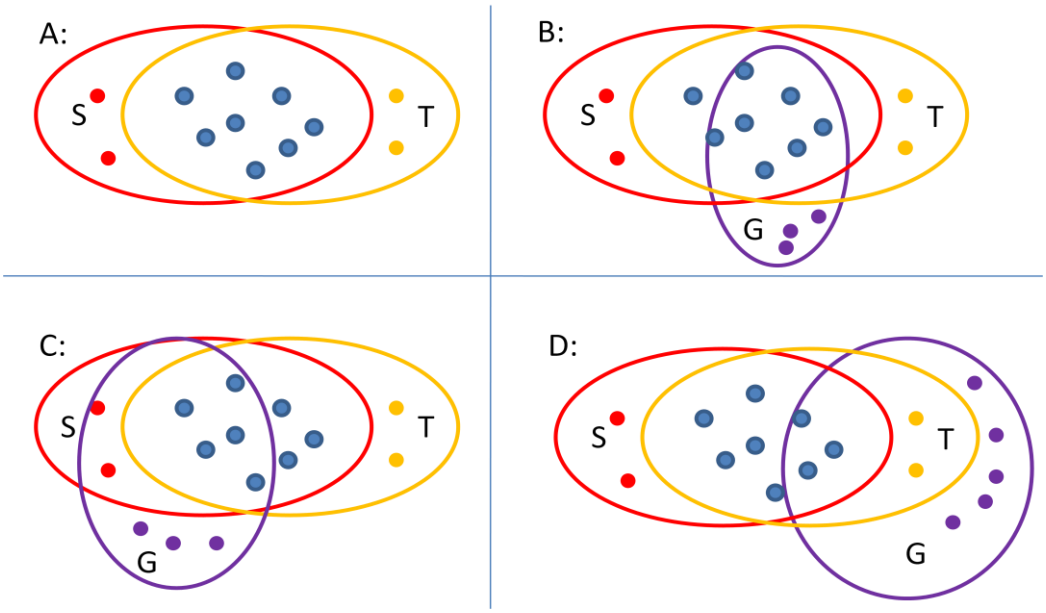

Figure 3-13. The possible coverage of transitive candidate

Generate HLA FOM

After the model evolution and model alignment, we can get a union of UML models which exist in most of the considered enterprises and are useful for enterprise interoperability. Then, we can revise those models such as rename the classes and class attributes, and convert these models into HLA object class. The figure 3-14 shows the structure of the HLA object classes. And then, according to different RTI (Run Time Infrastructure), this HLA object class can be translated into different formats of HLA FOM file.

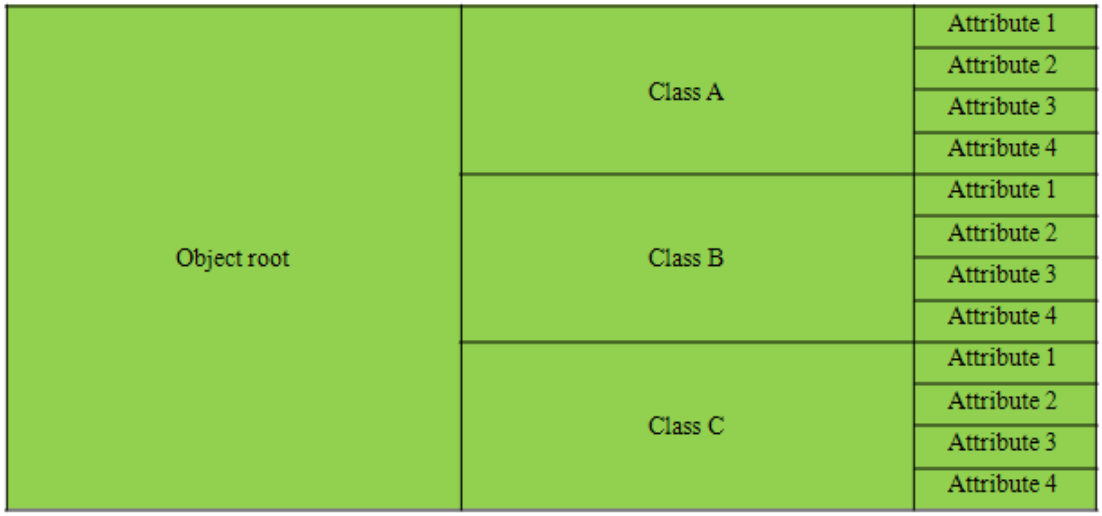

Figure 3-14. HLA object class structure

3.3.2.3. Generate HLA Federate Code Block

This section will explain the method of generating HLA Federate code block. This code block locates in the "adapter" part of the Harmonized single federate structure mentioned in section 3.2.2.2. This code block generation needs the simplified UML models, and also the system behaviour. As mentioned earlier, the model transformation and model reversal lose the information of system behaviour. The simplified UML models are static model that cannot represent the system behaviour. A serial of procedures will be carried out to trace and record some part of the behaviour of existing system.

System Traversal

The system traversal method is aimed at detecting the possible behaviour of existing system. The definition of behaviour is the action, reaction, or functioning of a system, under normal or specified circumstances. As the definition shows, the detection of system behaviour has to take place when the system is executed, and conform to a scenario.

As known, a running system is a black box, in which the data flow, system actions/reactions, and system states are invisible. User can only obtain different outputs by entering diverse input combinations, but without being aware of the detail. Thus, in order to make the detail visible, a tracer tool[12] is necessary. The tracer tool is commonly used in software testing, especially black-box testing. The black-box testing requires numerous high robust test cases[13] to detect any bugs of system execution. Similarly, this system traversal method also needs to define the test cases (called input combination in this method), which can fully cover the possible routine operations. The operations must conform to system operation manual and operating process.

The intention of using tracer tool is to detect the system execution paths. The tracer tool can trace any function calls happened in any classes or among any classes. Meanwhile, the simplified UML models have already been generated. As a result, the system traversal method can generate an execution path (as illustrated in figure 3-15) for each input combination. The execution path will be saved as a linked list that can be read by computer (software program).

[12] Tracer is a specialized software tool for logging to record information about a program's execution.
[13] A test case in software testing is a set of conditions or variables (input combinations) under which a tester will determine whether an application or software system is working correctly or not.

Every class model invoked will be saved as a node of the linked list, and every method invocation will be saved as a pointer (edge).

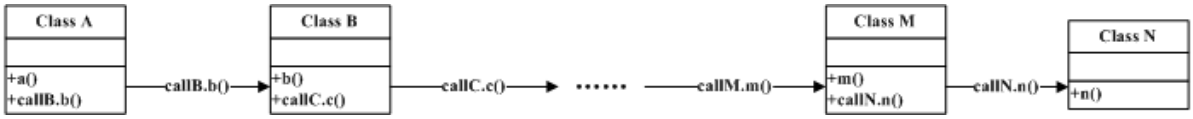

Figure 3-15. An execution path for each input combination

In addition, the tracer tool can also detect the function execution time that can be used for simulation time management.

Model Processing

Because one input combination will have one execution path, numerous execution paths will be detected after the system traversal. However, without rearrangement, these execution paths are intricate. They cannot be used for analyzing the system behaviour directly. Thus, the follow-up mission is to make these paths understandable.

- Step 1: these execution paths need to be categorized according to their relevance. As known, different operations with different input combinations will invoke different methods in different modules or sub-systems, and can lead the system into different states. Besides that, according to different runtime execution contexts, the same operation will turn to different modules or sub-systems, and can also lead the system into different states. In sum, different operations could lead an execution path with different starting point, but the execution path caused by the same operation might also have different starting point accidently. As a result, the starting point of the execution path is chosen as the relevance to partition the group of execution paths. As shown in the picture ① of figure 3-16, the execution paths with the same starting point will be put together.
- Step 2: because the execution paths of one categorization have at least one intersection point (the starting point), they can be synthesized into a complete directed graph with the same starting point (as shown in the picture ② of figure 3-16) (Biggs et al., 1986). The nodes of the graph are UML class models, and the edges are the function calls. This synthesis can eliminate the redundant information such as duplicate nodes and edges, so that the view of all possible execution paths becomes more systematic. However, this directed graph is not concise enough, and it can be reduced again by step 3.

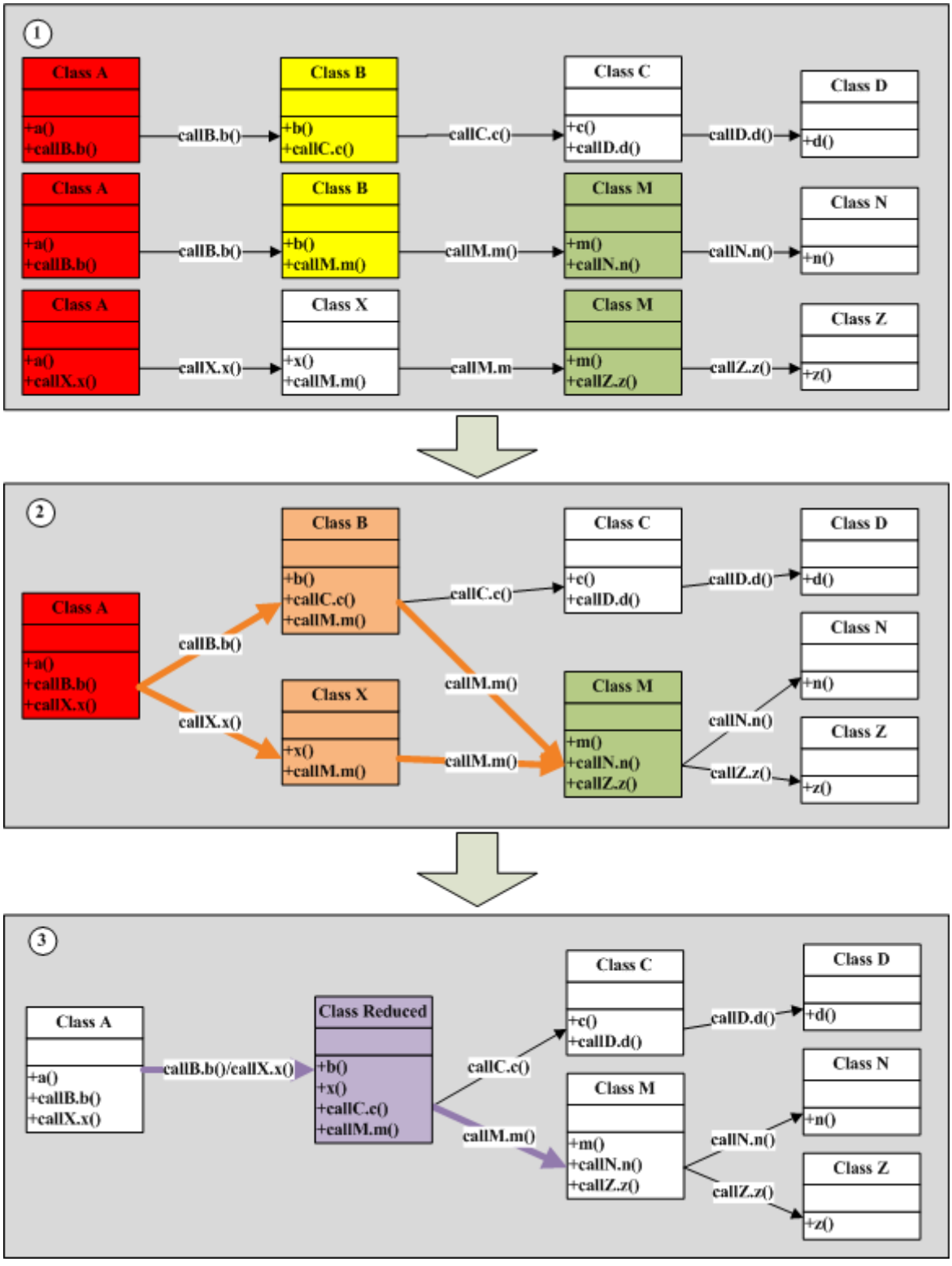

Figure 3-16. Model processing of execution paths

- Step 3: it is very likely to find a circle in a directed graph with many intersection points. As shown in the picture ② of figure 3-16, Class A, B, X and M form a circle with the edges of callB.b(), callX.x(), callM.m(), and callM.m(). According to the theory of

transitive reduction of directed graph, it is possible to reduce this circle. The theory of transitive reduction of directed graph defines that a transitive reduction of a directed graph G = (V, E) is a graph H = (V, F) where F is a minimal subset of E such that G and H have the same transitive closure (Aho et al., 1972). In graph theory, V is set of elements, and E is the set of binary relations of the elements. Thus, the explanation of this theory by using mathematic term is that a transitive reduction of a binary relation E on the set V is a minimal relation E' on V such that the transitive closure of E' is the same as the transitive closure of E. In other words, a transitive reduction of a directed graph G = (V, E) is the minimal representation graph G. For example, the directed graph shown in the picture ③ of figure 3-16 is the transitive reduction of the picture ②. The duplicative edges have been removed, and the nodes on the transitive closure path have been merged. The objective of this transitive reduction is to abstract the execution paths, so that it can be more straightforward, and easier to extract the system states.

After these three steps, the complex and intricate execution paths will be organized into a clearer and more straightforward map, which is easier for discovering system behaviours.

Behaviour Model Generation

Before explaining how to generate behaviour model, it is necessary to determine what level of detail of behaviour model is required. For example, if the behaviour model is only used for describing the system logic in general, i.e. the main I/O relation. Then the state machine is qualified. However, if the behaviour model is used for process interoperability or business interoperability, state machine is not competent enough for displaying business details. In that case, the behaviour model must be transformed into the models that can formalize the detailed business logic, such as BPMN model, GRAI model[14], DEVS model, etc.

As mentioned in section 3.2.2.2, the "adapter" is a simplified interface that simulates the dynamic business logic of the existing system. It is an interface that is responsible for handling participants' requests coming from RTI, and preparing input for the existing system. According to the complexity of the request, it can react immediately or indirectly by invoking the correspondent sub-system of the existing system. Thus, the state machine that can describe system logic in general is enough for guiding the generation of the "adapter". Therefore, this part will introduce a method to generate state machine from the reduced system execution

[14] GRAI represents Graphes àRésultats et Activités Interreliées. It is was developed in the early 1980's by the Laboratory of Automation and Productics of University Bordeaux I to design manufacturing management systems (Chen et al., 1997).

paths (as the directed graph shown in the picture ③ of figure 3-16).

As known, the control flow of a state machine depends on the sequence of events. Each state at least has one pair of received event and sent event. When one state receives an event, it will execute the actions inside that will change the state differently depending on the execution results. Meanwhile, the state change will trigger different sent events which will become the received event of another model's state. According to this description, the directed graph shown in the picture ③ of figure 3-16 can also be considered as a state diagram. Each node represents one state, and each edge represents one event. However, this state diagram is too verbose and can be optimized again.

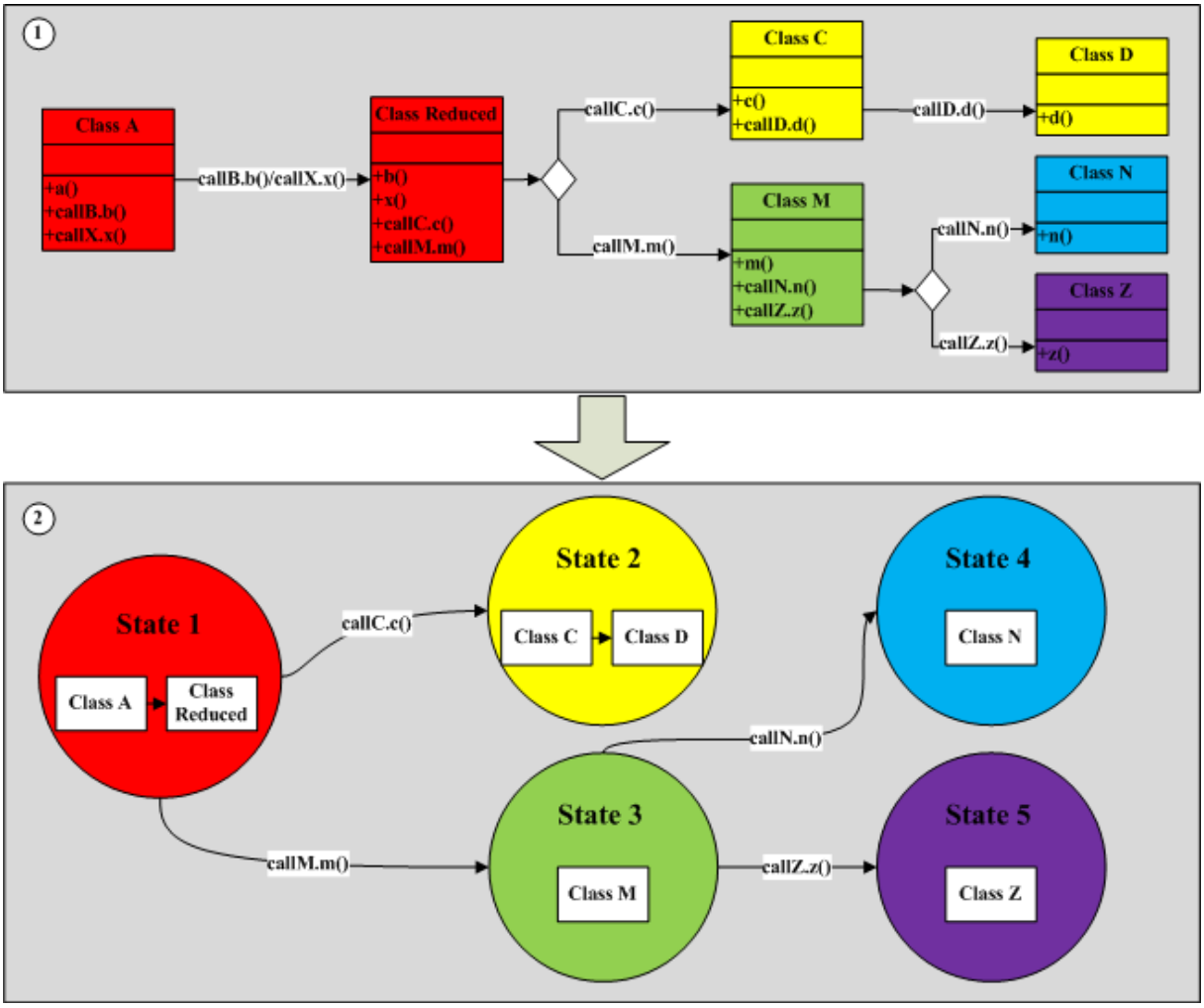

Figure 3-17. Behaviour model generation

As illustrated in the picture ③ of figure 3-16, the directed graph has many branches. Each branch represents an assertion that decides the function redirection. Therefore, an assertion

box can be added where the branch appears (as shown in the picture ① of figure 3-17). If the assertion box is considered as the cut point of the graph, and then the nodes before or after the assertion box belongs to an independent set. As a result, this independent set can be considered as a state. For example, in picture ①, the system smoothly runs from class A to class Reduced, until it meets the assertion box, and then it stops for deciding the next destination. In this case, the actions of class A and class Reduced can be treated as the inside actions of one state. To perform this combination on each assertion box, the state diagram can be optimized as the reduced state diagram shown in the picture ② of figure 3-17. Each state consists of the handles of classes, so that when the state is activated, the program can distinguish the entrance. The function call between classes is the transition of this state diagram, because the function call is interpreted as sent event triggered by the assertion of state change.

This method will not specify the exact number of states, because the "adapter" does not need to distinguish all the system states perfectly. The duty of "adapter" is to make a quick decision of the data flow based on the logic assertions. If the request is simple to ask, it will reply immediately. If the request is too complicated to reply directly, it can invoke the correspondent sub-system for answer.

Federate Code Block Generation

After the behaviour model generation, all the possible system execution paths are summarized into different state diagrams that correspond to a set of operations (input combinations). The state diagrams will be generated into diverse functions that define corresponding logic analysis. Hence, the federate code block needs a control function to dispatch the request to the right function of logic analysis. As the figure 3-18, before code generation, an initial state is added to play as a controller that can determine the request direction.

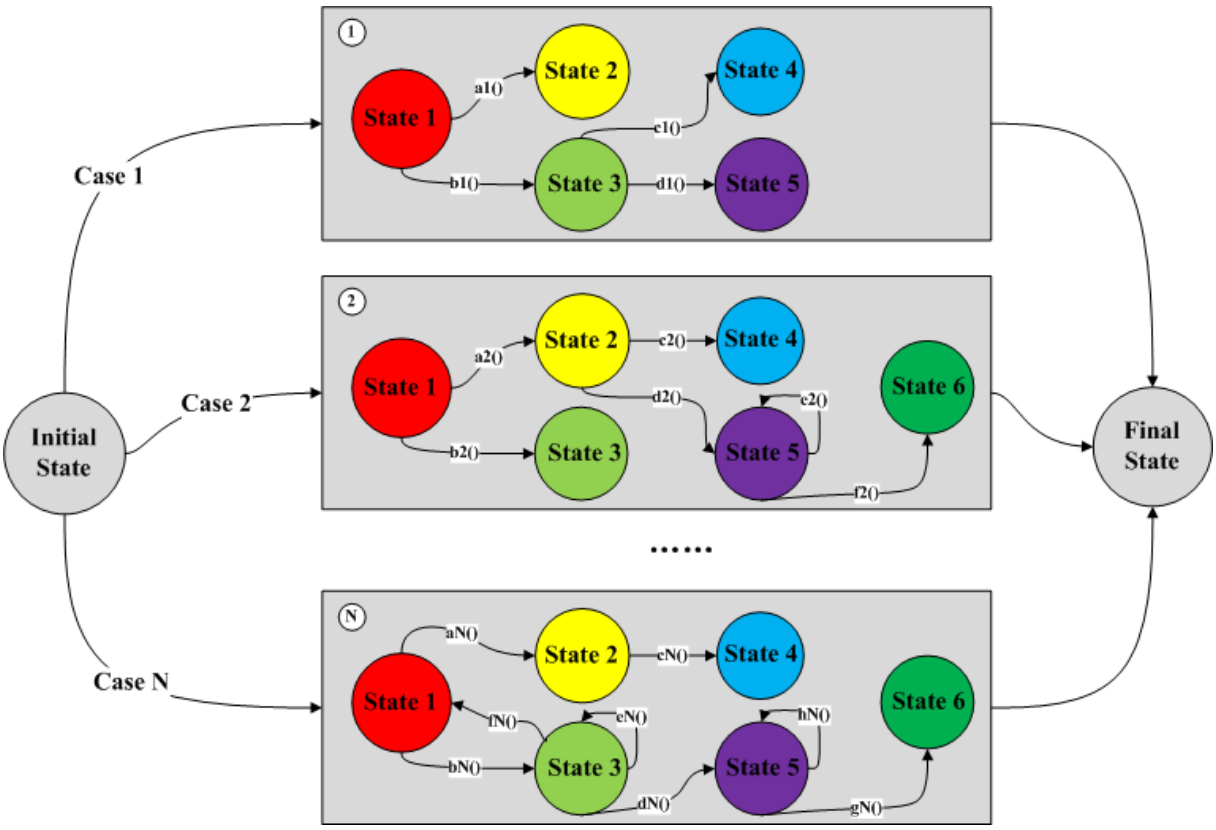

Figure 3-18. Federate code block generation

1) The code generation of the initial state: it needs firstly to classify the operations into different categories as logical conditions, such as data range, regular expression of operations, and so on. Afterwards, the rest of the code of the initial state is the alternative statement (for example, if, else, and else if) or selection statement (for example, switch case in Java) based on the logical conditions.
2) The code generation of state diagrams from 1 to N: if database access or database access handle of the existing system can be invoked, some state diagrams of simple business process can be transformed into a mini simulation code that can briefly represent the original code of the existing system. Otherwise, each internal action of these state diagrams will use the class handles to access the corresponding classes of the existing system. As figure 3-18 shows, the function call between classes is still used as the state transition. Thus, the state manipulation of each state diagram does not need to be redesigned. It just follows the usual logic that exists in the existing system.
3) RTI specific code generation: as mentioned in 2.4.3, federate code must implement the callback functions in the Class RTI::FederateAmbassador, such as function for granting time advance, functions for sending and receiving interaction, functions for reflecting attributes' values, and so on. The "adapter" concerns the functions for sending and

receiving interaction, so that it must reply the participants' requests correctly. Thus, interaction handles must be defined and published for other federates to subscribe, meanwhile, the "adapter" needs to subscribe to other federates' interaction handles. These definitions of interaction handles will be used to complete the HLA FOM. However, concerning the "on-the-fly" negotiation of federated approach, to specify and hardcode all the interaction handles in HLA FOM is inappropriate. Thus, it is better to abstract the interaction handles according to the categories of operations defined in step 1. And then, one code segment needs to be added into the function of the initial state, which can distinguish the types of interaction handles.

4) The code generation of the final state: no matter initial state turns to which state set (from 1 to N), finally the "adapter" will end up at this final state that will call the function of sending interaction to reply the requesters.

Overall, the three steps above complete the HLA FOM, and generate a control function, and several simulation functions. The control function is responsible to acquire participant's request by distinguishing interaction handles, and transmit participant's request to corresponding simulation code by judging from condition statements. The simulation function deals with the request by simulating the business process of existing system.

3.3.3. Summary

This section has introduced a model reverse method that can obtain static models and behaviour models, and transform these models into HLA relevant code.

Section 3.3.2.1 has introduced the way of obtaining model information by using the MoDisco Tool with constraints. The constraints ease the burden of UML model recovering process. The participants must be involved in this phase, because the constraints are the results of the negotiation among participants. For example, concerning the business confidentiality, the participants must designate the sub-systems or functional modules to be reversed.

Section 3.3.2.2 has explained the method of HLA FOM generation. This method firstly trims the reversed UML models by deleting unnecessary models. Afterwards, this method proposes a method called model evolution to classify the participants, so that the next step – model alignment can be easily carried out. Model alignment will pick out the similar models from

the models in the same category, and then restructure them into a new model. After several times of model evolution, a list of new models will be ready for generating HLA FOM. A software application has been developed to implement this method. Section 4.3.2 will explain this implementation and section 5.2.2 will demonstrate this software application. This software application supports the on-the-fly negotiation on the models. It can shorten the develop time of the federate for interoperation. In addition, the models extracted by this method can be used to create the web services for potential participants. This web services creation will be introduced in the next section.

Section 3.3.2.3 has explained the method of generating HLA Federate Code Block. This method firstly gathers all the possible system execution paths by using program tracer. Afterwards, these paths are integrated into several directed graphs that will be transformed into state diagrams. Finally, HLA Federate Code Block is generated based on these state diagrams. The theory of this method has been systematically described. However, it has not been fully implemented, because of the time limitation of my doctoral research. The algorithms of model processing and state diagram generation have been studied out without complete verification, so they will not be presented in this doctoral thesis. We have opened this part to the future work.

3.4. Web-enabled HLA federate design method

3.4.1. Why HLA evolved

The objective of using HLA Evolved Web Services is to provide an easy-pass for the potential participants to join the cooperative project based on traditional HLA.

As mentioned in section 2.4, HLA provides extremely high performance and scalability for achieving interoperability across disparate platform, reusing simulation models, time management, securing simulation environment, and etc. However, these high performance and scalability are restricted within the LAN (Local Area Network). On the other hand, Web Services provides a loosely coupled mechanism for performing coarse-grained services with modest performance over both LAN and WAN (Möller et al., 2005) (Möller et al., 2007). However, compared to HLA, Web Services is weaker in the time management, environment security control, and system state management. Because of these weaknesses, Web Services

cannot fully meet the demand of the federated approach of Enterprise Interoperability in technical level. Even though either HLA or Web Services seems to be imperfect for the federated approach, but the combination of them will be a perfect technical solution for the federated approach. Meanwhile, as mentioned in section 2.4.3.1, HLA evolved IEEE 1516^{TM}-2010 was published in 2010, it gives a notional instruction about how can HLA benefits from the Web Services such as the ease of deployment across wide area networks.

The Web-enabled HLA federate design method proposed in this section complies with the rules defined in HLA evolved IEEE 1516^{TM}-2010. This method can strengthen the compatibility and self-learning ability of the HLA interoperability environment. It allows the interoperability environment to adapt to different potential participants with heterogeneous cooperation purposes and modalities, and upgrade itself in order to conform to this adaptation.

3.4.2. The proposed web-enabled HLA federate design method

3.4.2.1. HLA Evolved Web Services scenario

The general scenario of HLA Evolved Web Services is illustrated in figure 3-19. It assumes that a cooperative project has been launched between several partner enterprises. The information systems of the members run correctly within the HLA federation. During this project, other enterprises want to join this project with different expectation, such as different cooperation time periods, different cooperation domains, different expected results from the cooperation, etc. Rebuilding the existing HLA federation is inappropriate because it will take immerse expense and time. Accordingly, our solution is to add one particular federate called *WebservicesFederate* as shown in figure 3-19. *WebservicesFederate* will allow the members inside the traditional HLA federation to connect with the potential business partners from World Wide Web in a more flexible and safe way (Tu et al., 2011b). This special federate will publish the Web Services that consist of various kinds of services of the existing HLA federation, different access permissions to the existing HLA federation, and the common API for connecting to the existing HLA federation. The “web-candidates” (potential business partners from World Wide Web) could use the common API and services, which are interesting for them, to generate their own local federate, and then connect to the existing HLA federation with different authorities via the Wide Area Network (WAN).

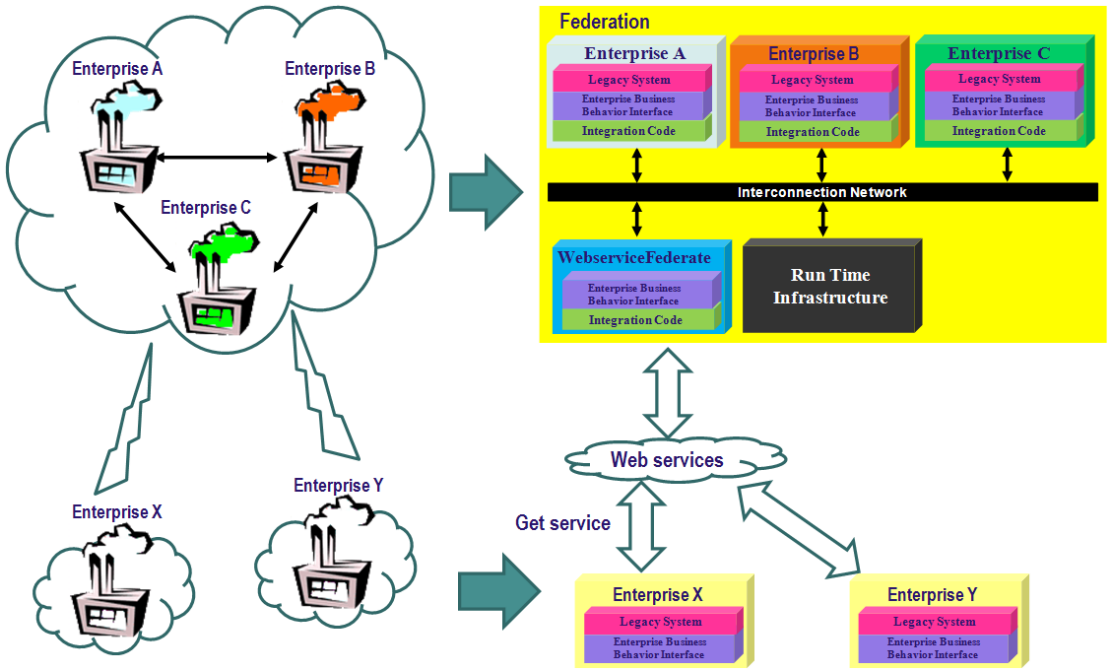

Figure 3-19. HLA Evolved Web Services

For example, in figure 3-19, two enterprises X and Y decide to participate in one existing project. Enterprise X is a supplier and enterprise Y is a client who is interested in the final product of this project. Thus, enterprise X has to know the workflow that is related to his business, and synchronize its information with other participants. While, enterprise Y only requires receiving information from the HLA federation, so, it doesn't have to synchronize with other systems. In that case, enterprise X must ask *WebservicesFederate* for the services with an authority of synchronization with other HLA federates. However, enterprise Y needs the service with the lowest authority which only can receive information from the HLA federation. Finally, both of them are connected with the existing federation via Web Services, even though they get different services.

3.4.2.2. Technical transcription

The figure 3-20 presents the technical transcription of the problems presented in Figure 3-19. The *WebservicesFederate* is called as bridge in this transcription. This bridge uses the Integration code (the result of Harmonized HLA and MDA) to communicate with other members of the existing HLA federation as the other "traditional" HLA federates do. On the other hand, it uses Enterprise Business Behaviour Interface (also the result of Harmonized HLA and MDA) to publish the Web Services which the existing members are capable to provide. The bridge is a multithreading processor, which is a standby federate for detecting

potential partners and handling their applications and requirements. When the bridge receives any request from the "web-candidate", it will launch a thread to handle the new case individually. Thus, this bridge plays as a viaduct with multiple lanes to monitor both the existing federates and the "web-candidate" / "web-partner" (business partners from World Wide Web) and dispatch the messages. In addition, as the figure 3-20 shows, the HLA federate of the "web-partner" only has Enterprise Business Behaviour Interface part but no Integration code. The reason of this design is to ensure the information privacy. As known, the information exchange through the WAN is not considered safe, but one of the advantages of HLA is high insurance of information privacy, so in order to sustain this advantage, the naked information exchange will only be taken place inside the traditional HLA federation. It means that the "Enterprise Business Behaviour Interface" of the "web-partner" will send the encrypted message, and the corresponding part of bridge will decrypt this message and use the communal Integration code to dispatch the message. Thus, the multiple lanes are only paved in the "Enterprise Business Behaviour Interface" of both sides.

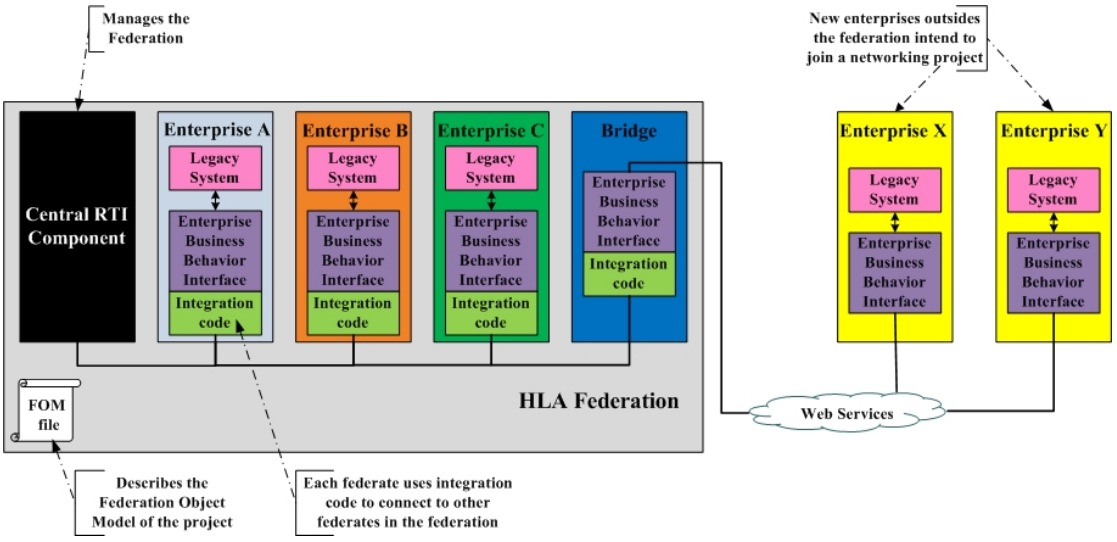

Figure 3-20. Architecture of HLA Evolved Web Services

3.4.2.3. Elected RTI

An open source RTI, poRTIco (poRTIco, 2009) has been chosen for implementation, even if it does not provide Web-RTI functionality. Actually, only one mature commercial RTI, pRTI, supports some Web-RTI functionality (Möller et al., 2007). Even in this one not all IEEE 1516-2010 features are already developed. As mentioned, the current status of commercial developments and the aspiration to develop an open framework has guided the choice to

poRTIco. But, to reach some HLA evolved requirements, new features have been added to poRTIco. As mentioned earlier, a *WebservicesFederate* component has been implement as a bridge, who takes in charge of providing web services, connecting and synchronizing HLA federates outside the HLA federation with HLA federates inside the HLA federation.

As mentioned in section 3.2, after the harmonization of MDA and HLA FEDEP, an integration code is provided with a RTI independent API for HLA Federates. This API can be reused and published as common API. So, the "web-candidates" can reuse this API and follow the second scenario of model reversal, mentioned in section 3.3, to generate their own Enterprise Business Behaviour Interface adapted to the common API. After that, a new federate outside the federation can send the information to the bridge via the Web services interface and be synchronized to the HLA federation.

3.4.2.4. WebservicesFederate design

WebservicesFederate design

A schema of WebservicesFederate design proposed in this thesis is illustrated in figure 3-21. In this design, WebservicesFederate is a special HLA federate, which is inside the Local area network (LAN) but not fully included in the HLA federation. According to this specific structure, WebservicesFederate is divided into two parts: one is WebservicesBridge, which is inside the HLA federation; another is WebServicesServer, which is outside the HLA federation but still inside the LAN. These two parts are connected by a socket. This design is customized for poRTIco RTI. As mentioned, this simulation is based on poRTIco RTI that doesn't support natively Web RTI functionality. In order to implement Web RTI functionality, the approach defines WebservicesBridge and WebservicesServer for WebservicesFederate.

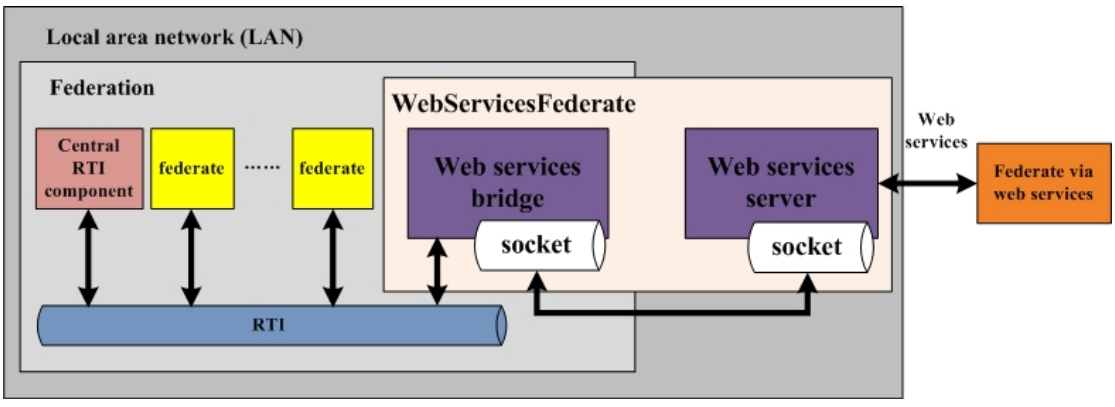

Figure 3-21. Web services federate design

- WebservicesServer: it is used to publish web services interface to potential customers outside the federation. It takes charge of monitoring and replying to the federate via web service. When this server receives the message from the federate outside federation, it generates a User Datagram Protocol [15] (UDP) data package and sends it to WebServicesBridge by the socket connection.
- WebservicesBridge: it uses to synchronize the message from the federate outside the federation with other federates inside the federation. This bridge transmits messages to federate inside the federation by RTI, but exchanges messages with the WebservicesServer by the socket connection. When web services federation establishes, this bridge launches a thread to monitor the events happening in the web service server.
- Socket data package: in order to ensure the security of the federation, common federation attributes are encapsulated into the web service interface, which is published by the web services server. So, WebservicesBridge encodes the attributes into the socket data package, and then this package is decoded by WebServicesServer. Afterwards, WebServicesServer generates the result that is requested by the federate outside the federation. In the opposite way, federates outside the federation can send request based on the web services it customized. While, the WebServicesServer receives the request, it translates the request based on the FOM, and then generates a data package which is decoded by the WebservicesBridge.

General solution for failure tolerance

As Web Services and UDP are involved in this simulation, the failure tolerance needs to be considered. This section proposes an example which only considers two failures: data exchange delay and data package lost.

Firstly, let's describe this example and define its major elements. Because this example is a scale real-time simulation, the scale (simulation time unit) needs to be defined first. Thus, as shown in Figure 3-22, the simulation time unit (Δt) of the federation is assumed to be 3 seconds, which means that a new event will be issued in every 3 seconds. The approach uses the conservative algorithm described in (Fujimoto, 2000) and (Zacharewicz et al., 2008). For example, in Figure 3-22, Federate A sends one event with a Time stamp (T_{stamp}) plus L_A (Lookahead of A) equals 3 to the event queue, so when simulation time passes one Δt, this event is triggered. Every federate can announce its events with T_{stamp} plus Lookahead.

[15] User Datagram Protocol is one of the core members of the Internet protocol suite, and one of the set of network protocols used for the Internet (Postel, 1980).

Lookahead is a special non-negative value, which establishes the lowest value of time stamps that can be sent in its Time Stamp Order (TSO) messages. In the simulation, the lookaheads of WebServicesfederate and the HLA federates outside of federation are assumed to be 0. Meanwhile, the lookaheads of the HLA federates inside the federation are bigger than 0 and depend on their own process. When simulation time moves forward, RTI sends $Event_j$ of $federate_j$, whose $T_{stampj} + L_j > LBTS_i$ (Low Bound on Time Stamps), is triggered and sent to the related $Federate_i$.

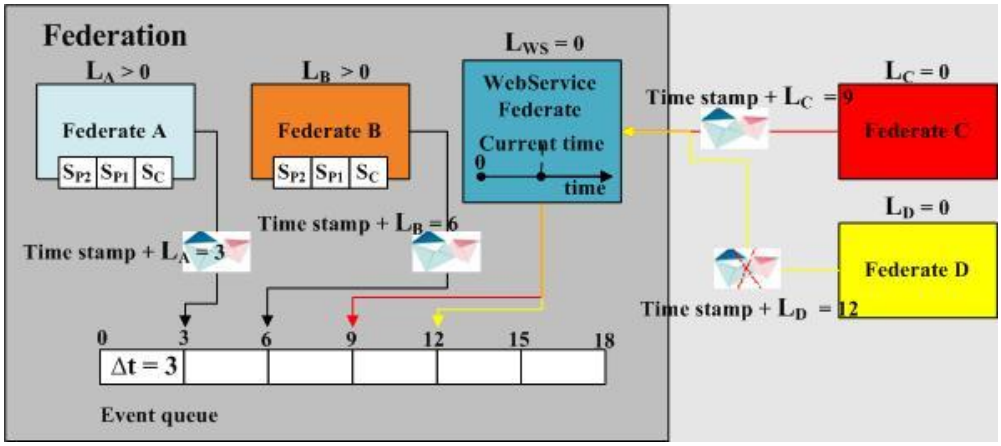

Figure 3-22. General solution for failure tolerance

Due to the performance of Web Services and UDP and also this simulation context, the approach proposes that each federate can store three states, S_C, S_{P1}, and S_{P2}. S_C is the current state. S_{P1} is the previous state (roll back one Δt). S_{P2} is the state before the S_{P1} (roll back two Δts). The reason for saving three states is to backup necessary information in order to answer overdue customer requests from *WebServiceFederate*. The reason of only saving three states is to limit the times of re-ACK(ACKnowledgment) between the *WebServiceFederate* and federates outside the LAN, which can ensure the message channel between the *WebServicesbridge* and the *WebSerivesserver* fluent and strictly control the increase of each federate's memory load as well as the amount of redundancy in the federate. In addition, in this simulation context, the time scale allows federates inside the LAN to keep their current state for a quite long period, so three backup states are enough for querying (it does not roll-back the state for overdue customer request. It only provides the state query service. This roll-back querying does not affect the message synchronization inside the federation). Normally, the approach also proposes by no reply after the third PING (Packet Internet Grope), the web connection is broken.

The solution for failure tolerance is the following: In this project, some of the HLA federates are federates outside the HLA federation. They send events to the federation and synchronize with other federates via Web services, so the time delay within web transmission and the possibility of package lost should be considered.

- *Data exchange delay*: for example, in figure 3-22, the federate C sends one message with a time stamp plus L_C equals 9 to the WebServicesFederate. Normally, when the WebServicesFederate receives this message, the current simulation time ($T_{current}$) should be less than the T_{stamp} plus L_C, but, if this message transmission has several seconds time delay, this message arrives $T_{stamp} + L_C < T_{current}$, which means that this event has already expired. As a result, there is no reply for the federate C. The solution for data exchange delay is if $T_{current}$ is bigger than $T_{stamp} + L_i$ of the $message_i$, then the WebServicesFederate asks for the past state of requesting federate. There is another situation if the authority of $message_i$ (MA_i) is low, the federation ignores this message.
- *Package lost*: for example, in figure 3-22, the federate D sends one message with T_{stamp} plus L_D equals 12 to the WebServicesFederate. However, if the package lost during the web transmission, then this message cannot join the simulation of the federation before its own time stamp. As a result, there is no reply for the federate D. The solution for package lost is to set the attribute in the federate D called waiting time (T_{wait}). If T_{wait} is bigger than Δt, then federate D resends the message. The maximum resend time (F_{resend}) is two times Δt. If the WebServicesFederate receives the resend message, it calculates the time difference ($T_{difference}$) and decides which state of the requesting federate is used for the simulation. Another situation is when the authority of the message is low, the federation ignores this message.

The general algorithm of the failure tolerance is the following:

- For federate outside the federation:

```
F_resend = 0;
while (F_resend < 2) {
  if ( T_wait > Δt ){
     resend message;
     F_resend++;
  } else {
     F_resend = 2;
```

}

}

- For WebServicesFederate:

if ($T_{current}$ *>* $T_{stampi} + L_i$*) {*

if (MA_i *!= low) {*

$T_{difference}$ *=* ($T_{current}$ *-* $T_{stamp} - L_i$*)/*Δt*;*

switch ($T_{difference}$ *) {*

case 0 : state = S_C*; break;*

case 1 : state = S_{P1}*; break;*

case 2 : state = S_{P2}*; break;*

case 3 : ignore message; break;

}else{

Ignore message;

}

} else {

if ($T_{stampi} + L_i$ *>* $LBTS_j$*){*

send event to Federate j;

state = runSimulation();

} else {

state = S_C*;*

}

}

- For federate inside the federation:

while (simulation time passes Δt *){*

$S_{P2} = S_{P1}$*;*

$S_{P1} = S_C$*;*

S_C *= runSimulation();*

}

3.4.3. Summary

This section has introduced the method of designing Web-enabled HLA federate based on the open source RTI, poRTIco. This method has proposed a new component, WebserviceFederate, straddling between HLA federation LAN and WAN to fulfil HLA 1516-2010 new

requirements. The WebserviceFederate is designed to bridge the gaps between HLA Evolved approach requirements and the HLA 1.3 API provided by portico. This method has also proposed a solution for failure tolerance, which can recover the lost information caused by data exchange delay and data package lost. This failure tolerance solution can also ensure that the HLA federation runs smoothly including web services. Even if one federate of "web-partner" is disconnected because of network fault, the WebserviceFederate will play as a standby federate until it connects again.

The objective of this method is to achieve easy connection for potential participants, authority management, and interoperation environment management for HLA federation (interoperation environment). A software application has been developed to implement this method. Section 4.4 will detail this implementation and section 5.2.3 will demonstrate this software application.

The method is based on HLA technology, so the establishment of dynamic interoperability still has a common standard to follow even if it is only in the technical level. Even so, this research work can be considered as an answer to new challenges engendered by future internet requirements at the semantic level, and to create, in particular, enterprises more dynamically interoperable.

3.5. Short-lived ontology method

3.5.1. Why short-lived ontology

The previous sections have introduced the framework for defining development lifecycle and structure of HLA federate, model reverse method for obtaining valuable information of existing system, and web-enable HLA federate for agile technical support. Up to now, the infrastructure of federated approach has been set up, but one more important element, information analysis, is absent to activate this approach. One of the expected results is transient information exchange and analysis without common format at conceptual barrier. Section 3.4 has proposed the HLA evolved Web Services solution for transient information exchange, but has not solved the problem of transient information analysis without common format. This section will introduce the short-lived ontology to handle this problem.

As mentioned, ontology is used to organise and handle data by semantically interconnecting them. Many existing enterprise interoperability researches and projects have used ontology to translate the message with different semantic meanings and structures, or map diverse models. Most of the researches and project used the common format or predefined format for translation or mapping, which cannot satisfy the on-the-fly requirement of federated approach. Therefore, short-lived ontology is proposed to minimally avoid the common format by predefine the format during the dynamic negotiation.

3.5.2. Overview of short-lived ontology

"Short-lived ontology" is a particular non persistent ontology (Zacharewicz et al., 2009), with a very short lifetime. To the extreme it can exist (and persist) only during a communication between interlocutors. The Figure 3-23 illustrates informally the communication mechanism of "Short-lived ontology".

- Case a: the "enterprise 1" sends information and the ontology to understand (decode) it at the same time. This ontology is supposed to be only valid for this information. The ontology is not persistent above the relation of the two enterprises.
- Case b: the "enterprise 1" sends only the information to "enterprise 2". Once "enterprise 2" receives the information, it interprets the meaning using its local ontology if it is able to decode the information. If not, it asks for the ontology associated to the message to the sender of the message. The enterprise can conserve the new received ontology to reuse it with further data sent by the same emitter or another one also compliant to the same ontology. A "best before end" date or a countdown of validity can be associated to the ontology.

In the case a, the information can be exploited directly thanks to the ontology received at the same time. However, the information size exchanged is more important, in addition, it can be intercepted and the confidentiality can be broken. In the case b, the confidentiality is enforced but it requires more exchanges between the two partners and consequently increasing the communication duration.

According to the definition of federated approach, case b is the "on-the-fly" solution. Case b can also ensure the information confidentiality. From that postulate we introduce the concept

of “short lived” ontology (this ontology definition is based on the definition in (Gruber, 1995)), where ontology can be, in some case, suppressed after use or have a finite duration validity. This “short-lived ontology” approach will be used to dynamically handle the interoperability issue in data concern.

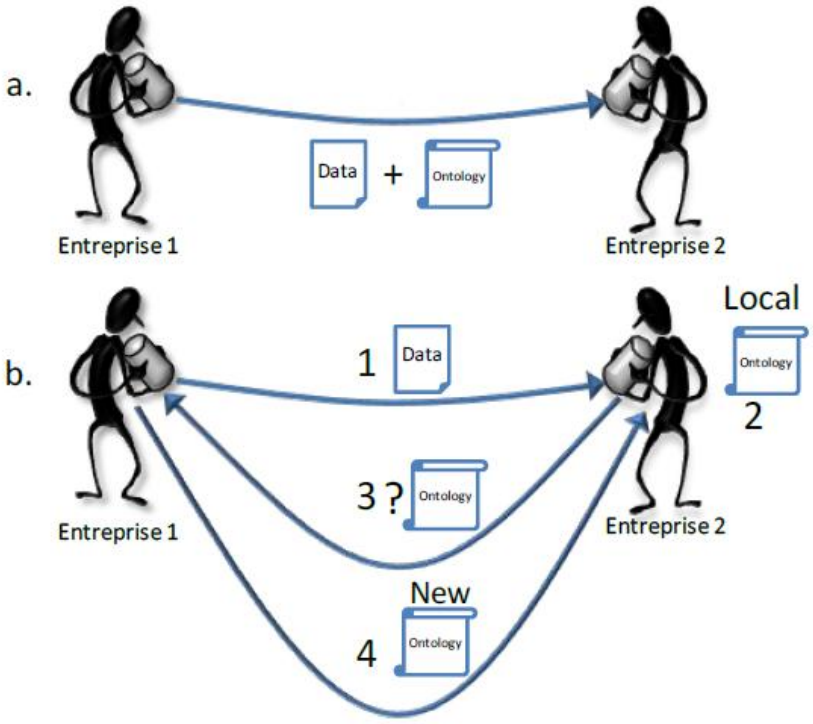

Figure 3-23. short-lived ontology

3.5.3. Short-lived ontology for federated approach

As mentioned in section 2.5.3, there are three kinds of ontology mapping approach for information integration (H.Wache et al., 2001), single ontology approach, hybrid ontology approach, and multiple ontology approach. Compared to the interoperability approach in enterprise interoperability framework：

- The single ontology approach is more suitable for integrated approach. Because single ontology approach needs a global ontology to provide a shared vocabulary for the specification of the semantics. All the information sources are related to this global ontology. While, integrated approach needs a common format for all models to develop systems.
- The hybrid ontology approach is similar to the unified approach. Because, hybrid

ontology approach requires a shared vocabulary to be built upon the individual local ontologies of different information sources. The shared vocabulary contains basic terms of a domain, and all the local ontology are required to refer to it. In sum, unified approach needs a common predefined format only exists at meta-level for mapping.

- The multiple ontology approach is supportive for achieving federated approach. Because, multiple ontology approach has no common and minimal ontology commitment about one global ontology. Each information source is described by its own local ontology. Federated approach requires dynamical adjustment and accommodation without predefined common format.

Therefore, the short-lived ontology must follow the principle of the multiple ontology approach. The technical schema of the short-lived ontology is shown in figure 3-24. When a message requester (Enterprise B shown in the right side of figure 3-24) receives information, it will try to decode the information by using its local ontology glossary. This ontology glossary is initiated by using the set of similar models, which is generated in phases of model evolution and model alignment mentioned in section 3.3.2.2. If the translation from local ontology glossary is not understandable for Enterprise B, it can demand to the emitter (Enterprise A shown in the left side of figure 3-24) to deliver the ontology translation associated to this message. After Enterprise B obtains all the information required, the received ontology translation can be deleted. However, the terms inside the ontology translation can also be temporarily saved in the local ontology glossary of Enterprise B. This local ontology glossary is a self-learning system with limited space (in order to save the memory and also avoid the redundancy), which means that this glossary can be self updated automatically. Every ontology term of this ontology glossary has a weighting coefficient for its ranking, which can measure the popularity of ontology term. If the coefficient of the ontology term decreases to the bottom, this ontology term will be deleted from the local ontology glossary.

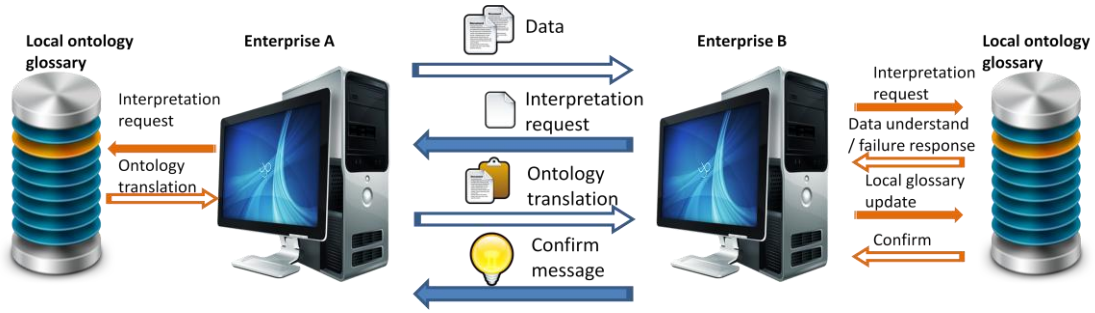

Figure 3-24. Technical schema of the short-lived ontology

In addition, because enterprises are isolated from this message translation process, the above-mentioned process must be handled by the Enterprise Business Behaviour Interface which is the output of the Harmonization of HLA and MDA. As mentioned in section 3.3.2.3, the "adapter" of the Enterprise Business Behaviour Interface has to process the participants' requests, and also transmit the response to these participants. Thus, this short-lived ontology method can be considered as information pre-processing and after-treatment of the "adapter". The information pre-processing will decode the request and then pass it to the "adapter", if it fails in decoding, it will require the translation from the requesters. The after-treatment is responsible for transmitting the response to requesters, and translating response if requesters cannot understand it. In order to link up with the initial state and final state of the state diagram generated by model reverse method mentioned in section 3.3, the state diagrams of the information pre-processing and after-treatment must be defined. The output of the information pre-processing must be discernible for the initial state. It means that this output must be in the range of possible system input combinations defined in the initial state, so that the initial state can precisely decide the direction of the information flow and change the system state. The after-treatment must help the final state to process the answer from the existing systems, and then reply the requesters. Part A of figure 3-25 shows the state diagram of message emitter, and Part B of figure 3-25 illustrates the state diagram of message receiver. Actually, one single federate must implement these two state diagrams for the implementation of the Enterprise Business Behaviour Interface. Message emitter will be implemented as the after-treatment, and message receiver will be implemented as the information pre-processing.

- Message emitter has four states: initial/final, message sent, interpretation preparation, and interpretation sent.
    - Initial/final: it is the initial or final state. It is waiting for "send message" order to change the state, or waiting for "confirm" events to stop the process.
    - Message sent: after sending the message, it is waiting for the feedback. If the feedback is "confirm message", it will turn to final state. But if the feedback is "request interpretation", it will change into the "interpretation preparation" state.
    - Interpretation preparation: it will be activated, if the interpretation is needed. It will end up with sending the interpretation.
    - Interpretation sent: after sending the interpretation, it is waiting for the feedback. If the feedback is "confirm interpretation", it will turn to final state. But if the feedback is "deny interpretation", it will return to the "interpretation preparation" state.

- Message receiver has five states: initial/final, message analysis, business processing, interpretation preparation, and wait for interpretation from requester.
  - Initial/final: it is the initial or final state. It is waiting for "Receive message" order to change the state, or waiting for "Send response" event to stop the process.
  - Message analysis: after receiving the message, it will determine whether the message is understandable. If so, it will move forward to business processing. Otherwise, it will search the local ontology glossary for translation.
  - Business processing: if the message is understandable, it will process this message, and then send the response.
  - Interpretation preparation: it will be activated, if the interpretation is needed. If the local ontology glossary can provide the answer, then it will end up with sending the interpretation. Otherwise, it will send the interpretation request to the message emitter.
  - Wait for interpretation from requester: it is waiting for the message emitter's answer. If the answer is ok, it will end up with confirming interpretation, and turn to final state. But if the answer is still not understandable, it will send the interpretation request again and wait for the answer.

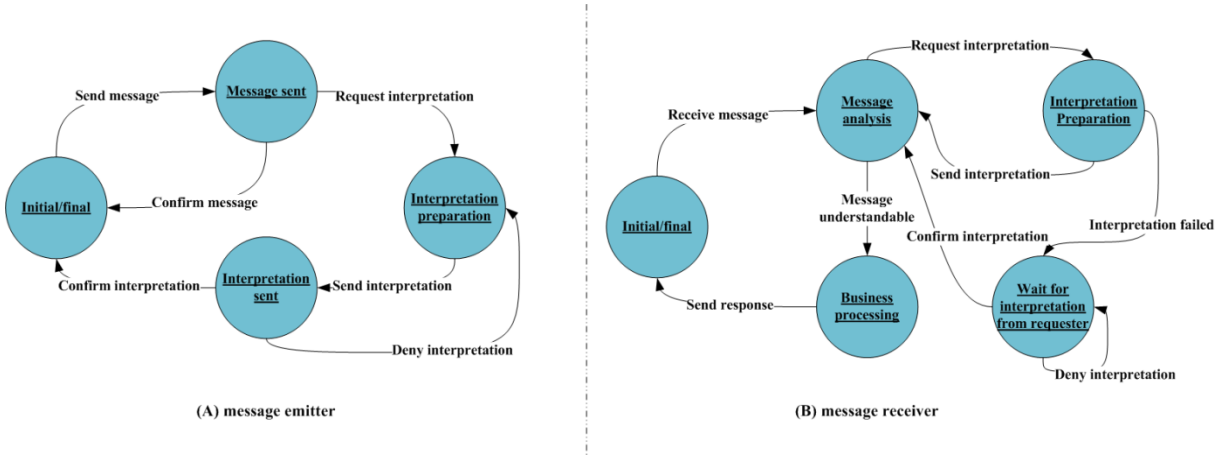

Figure 3-25. State diagrams of message emitter and receiver

3.5.4. Summary

This section has introduced the short-lived ontology method for implementing “on-the-fly” negotiation that is one of the expected results. This section has explained the general idea of the short-lived ontology, and also the mechanism of short-lived ontology for federated approach according to the result achieved in the previous sections. The mechanism includes the method of initiating and upgrading the local ontology glossary, and the technical schema of the short-lived ontology interpretation request/response. In addition, the state diagrams are designed conform to this technical schema, so that the short-lived ontology method can be linked up with the model reverse method to develop an intelligent agent for achieving federated enterprise interoperability.

However, as the same situation as the part of HLA Federate Code Block generation, this method has not been fully implemented yet. The algorithms of the technical schema of the short-lived ontology interpretation request/response have been studied out without complete validation, so they will not be presented in this doctoral thesis. We have opened this part to the future work. Another PhD candidate of our laboratory is working on this part. He has proposed a novel ontology alignment approach with multiple strategies and aggregated based on Method Analytic Hierarchy Process (AHP). This approach supports the dynamic and automatic aggregation of different matching results (Song et al., 2012).

3.6. Conclusion

This chapter has presented the Harmonized and Reversible HLA based framework and methodology. This approach is a novel idea that combines the existing methods and techniques mentioned in chapter 2 to achieve federated enterprise interoperability. The overall contribution is summarized in figure 3-26. This research firstly proposed a harmonized HLA and MDA Framework which aims at implementing federated enterprise interoperability. Under this framework, there are three methods, model reverse model, web-enable HLA federate design method, and short-lived ontology method. The framework defines the general guideline for the implementation of these three methods. These three methods also complement each other in order to achieve the expected result of the federated approach of enterprise interoperability.

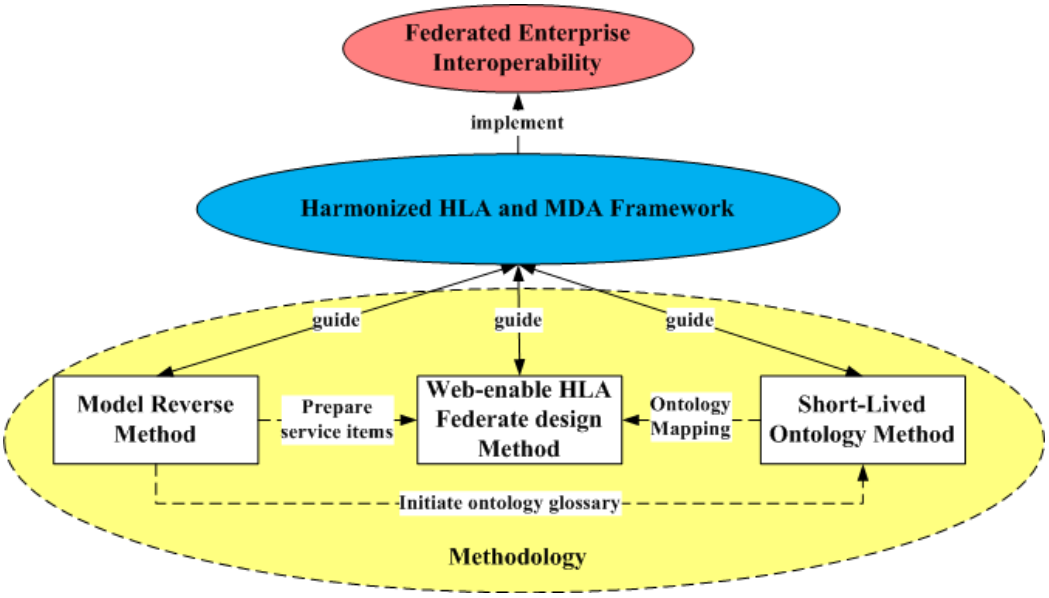

Figure 3-26. Overall contribution of this research

Section 3.2 has presented the Harmonized HLA & MDA engineering framework. This framework provides a new five steps development lifecycle starting from conceptual models to code implementation. This lifecycle combines HLA FEDEP with MDA. MDA is responsible for standardizing the modelling process, so that the models are general and common, which can enhance the model reusability. On the other hand, HLA provides a technical environment, which allows the model transformation to perform towards a clear target with constraints. As the result of this framework, the harmonized single federate structure provides a novel view of HLA federate, which dissociates the business behaviour code from RTI specific code. This dissociation reduces the model coupling, which can enhance the system reusability and maintainability. In addition, this dissociation promotes the implementation of "plug and play" mechanism, which can help to achieve the rapid, and dynamic interoperability establishment, and agile environment compatibility.

Section 3.3 has proposed the model reverse method. This method uses MoDisco tool to discover UML model that is initial data of this method. A process of model evolution and model alignment has been performed on the UML models, which achieves the interoperability modelling in "on-the-fly" negotiation. The processed models can be used to generate HLA FOM and initiate local ontology glossary that is introduced in section 3.5. Another process of behaviour model discovery has been proposed to generate system state diagrams that can be transformed to system simulation code. This process avoids completely redevelop the existing systems, and allows them to establish interoperability rapidly. The objective of this model

reverse method is to implement the harmonized single federate proposed in section 3.2, in order to achieve “plug and play”.

Section 3.4 has proposed the method of web-enable HLA federate design based on the open source RTI, portico. This method fulfils HLA evolved IEEE 1516^{TM}-2010 standard. A novel federate called WebserviceFederate is designed to bridge the gaps between HLA Evolved approach requirements and the HLA 1.3 API provided by portico. This method uses the results of model reverse process, such as similar models for HLA FOM, and behaviour models, to generate the web services. Thus, the potential participants can use the web services to rapidly generate their own “adapter” to join the existing HLA federation. This method intends to achieve “easy connection” for potential participants, and authority management and interoperation environment management for HLA federation (interoperation environment).

Section 3.5 has introduced the short-lived ontology method. The federate approach of Enterprise Interoperability requires that the interoperability accommodation and adjustment should not impose the existing models, languages and methods of work as the common format. The short-lived ontology is used to support this “on-the-fly” negotiation semantically.

The theory of the harmonized and reversible HLA based methodology has been systematically described. However, the behaviour model reverse method and the short-lived ontology method have been proposed, but only been partially implemented, because of the priority of implementation and time limitation of my doctoral research. The algorithms of model processing, state diagram generation, and the technical schema of the short-lived ontology interpretation request/response have been studied out without complete verification. We have opened these parts to the future work.

Chapter 4. Implementation of a Model driven and HLA based Reverse Engineering Tool

4.1. Introduction

This chapter will introduce the architecture and the implementation of functionality modules of Model driven and HLA based Reverse Engineering Tool based on the framework and methodologies presented in the previous chapter. This tool is based on poRTIco RTI and developed in Java language. It is implemented on Eclipse, and can be run in Windows NT or UNIX system with JDK 1.6.0 (or higher) environment and poRTIco environment. JAX-WS (JAX-RPC)[16] is used for implementing web services. JFreeChart[17] is used for illustrating the simulation result.

4.2. The architecture of Model driven and HLA based Reverse Engineering Tool

The objective and functionality of this tool is identified by breaking down the name "Model driven and HLA based Reverse Engineering Tool":

- *Reverse Engineering* means that this tool can acquire models of enterprise information systems by rewinding the existing systems.
- *HLA based* means that the target platform of this tool is HLA. The end user will connect to this platform through a federate of HLA federation.
- *Model driven* means that this tool must solve the interoperability issues based on models of rewound systems, and then reform the models into the interoperable models, which can be converted into executable code according to the target platform.

Thus, the objective (or output) of this tool is an interoperable ISs communication platform based on HLA. The functional modules of this tool are (1) a build time module including model reversal, model adjustment, and target model & code generation, and (2) a run time including message dispatch and management. The architecture of this tool is illustrated in figure 4-1.

[16] JAX-WS (Java API for XML-WebService) is the evolution version of JAX-RPC that provides Web services API operations by using the annotation of Web services in an open configuration information and configuration information on SOAP messages (Oracle, 2012).

17 JFreechart is an open-source framework for the programming language Java, which allows the creation of a wide variety of both interactive and non-interactive charts, such as X-Y chart, pie chart, Gantt chart, and etc (JFreeChart, 2008).

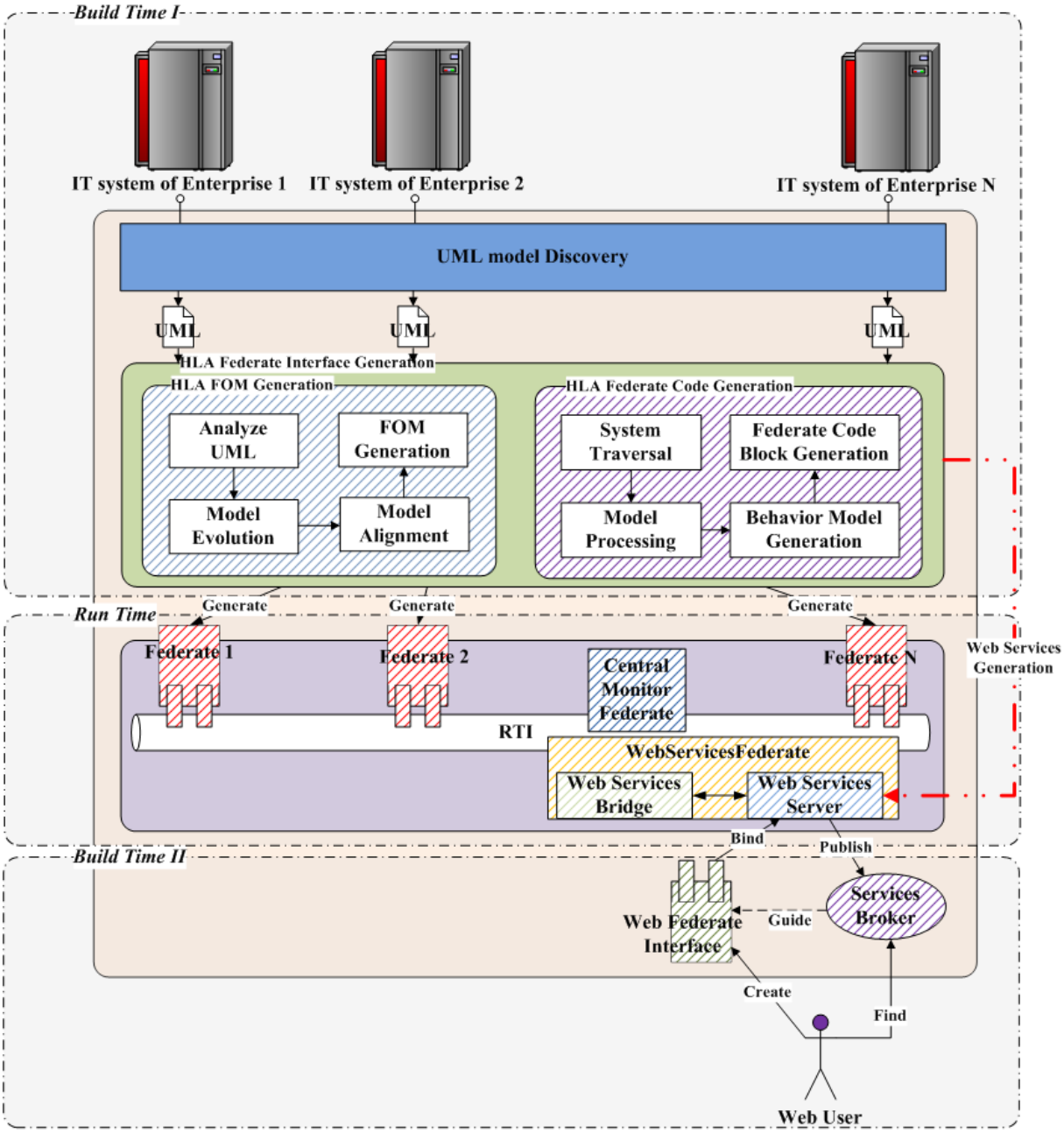

Figure 4-1. The architecture of Model driven and HLA based Reverse Engineering Tool

- Build time: HLA&MDA harmonization and model reversal will be performed at this time. As mentioned in chapter 3, according to the differences of the perspective, interest, authority, and join-time slot of the participants, the reverse level will be different and the HLA Federation is divided into traditional part and web-evolved part. Thus, the build time will be divided into two parts to cater for diverse requirements of different performances.
  - Build time I: it is the time for initiating the interoperation environment. It is the first priority of the interoperability development and this tool.

- Build time II: it is the extra and agile part of the interoperation environment, which takes in charge of discovering the potential participants, helping new participants to adapt to the environment, and managing these special participants.

- Run time: it is the execution time of the interoperation. The HLA federation will manage the interactions among the participants, maintain the status of the participants, and control the interoperation environment.

4.3. Build Time I

The Build Time I is responsible for interoperability environment establishment. It means that Build Time I must bring all the participants' existing IT systems together for Enterprise Interoperability. Thus, the main task of Build Time I is to discover models from legacy systems, and perform the interoperability modelling on these models, which is corresponding to the first model reverse scenario mentioned in section 3.3.

The harmonized federate mentioned in section 3.2, which consists of Integration code and Enterprise Business Behaviour Interface, is the expected output of this module. Thus, one division of the work of this module is HLA FOM generation, which is based on static model discovery, analysis, and reform. This part will systematize the global scenario of the interoperation, e.g. definition of primary entities and basic interactions, and then specify this scenario into HLA and JAVA related model, e.g. HLA FOM and correspondent JAVA object bean. Another division is HLA federate code generation, which is based on dynamic model discovery, analysis, and reform. This part will systematize the scenario of the individual interoperable entity, e.g. description of entity behaviours and statuses, and then specify this scenario into HLA and JAVA related model, e.g. HLA SOM and correspondent JAVA action bean. The basis division of this module is UML model discovery, which provides the raw material, UML model, to the other two divisions.

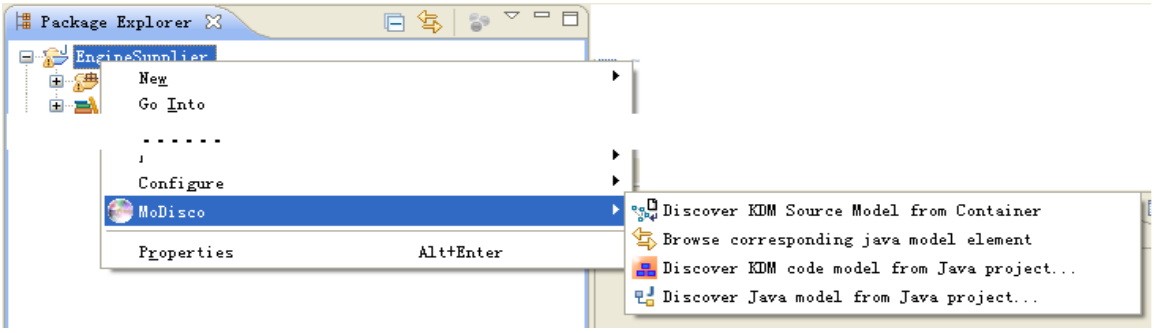

Figure 4-2. Modisco Tool usage of KDM, Java Model obtainment

4.3.1. UML model discovery

This division can obtain the raw UML model with the aid of MoDisco Tool. As mentioned earlier, MoDisco tool is an Eclipse GMT component. It is available on the Eclipse Website http://www.eclipse.org/MoDisco/downloads/ with the latest version 0.10.0 released on June 13th, 2012. After installing it (the full instruction of how to install MoDisco is explained in (MoDisco, 2012a)), the right-click popup menu of Eclipse will be changed by adding a new menu bar with MoDisco logo. By clicking the right mouse button on one project in the "Package Explorer", a popup menu with a menu bar labelled "MoDisco" (as highlighted in figure 4-2) will show. Following the options insides this bar, you can obtain KDM model and Java model. After KDM model is obtained, a popup menu with a menu bar labelled "MoDisco" (as highlighted in figure 4-3) can be activated by clicking the right mouse button on KDM model item. This item can be found under the structured tree of selected project. Following the options insides this "MoDisco" bar, a menu bar labelled "Discover UML model from KDM model" can be found, which can be used to obtain UML model. The detail of the usage of MoDisco Tool is introduced in (MoDisco, 2012b).

Figure 4-3. Modisco Tool usage for obtaining UML Model

4.3.2. HLA FOM generation

This division consists of four sequential sub-modules, Analyze UML, Model Evolution, Model Alignment, and FOM generation.

4.3.2.1. Analyze UML

The UML models obtained from the MoDisco Tool are saved in XML format (as the tree structure shown in figure 4-4) and as an .uml file. Each item of the tree structure has a "xmi:id" and "name", so that it can be uniquely identified. The "xmi:id" will also be used for class dependency and association. Each item also has some other information of correspondent

level, but not all useful. Besides that, the .uml file would contain bulk information, which is not interesting for HLA FOM generation, such as external java package, common java datatypes, and etc. Thus, an "Analyze UML" sub-module is required to select the useful information, and arrange this information.

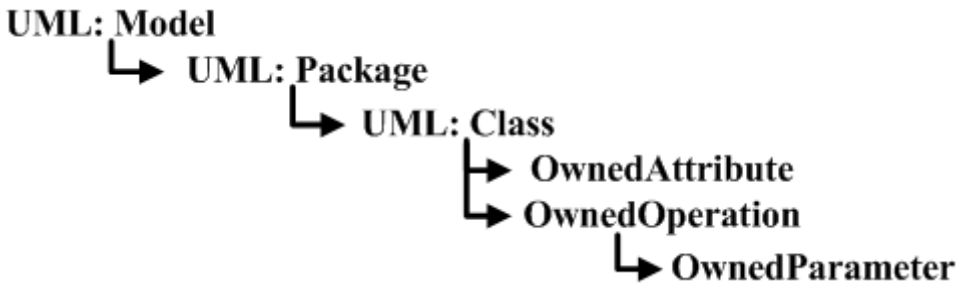

Figure 4-4. UML file structure

The procedures of the Analyze UML sub-module include:

- Parsing and simplifying UML model: to load the .uml file and parse the file based on W3C XML technology specification (W3C, 2008). As mentioned that the .uml file would contain unnecessary information, so during the model parsing, the program will ignore the elements labelled as "external", "source references", "common java datatypes" and etc.
- Sorting UML model: after .uml file decomposition, the information of each item need to be collected and catalogued, so that the elements of the UML model can be found or recalled easily by the program. In order to achieve this, data structure of each item is defined in the way shown in figure 4-5. Each UML node has attributes "xmiId" and "name" as unique identity, and has attributes "upper" and "lower" as the bidirectional pointer to its father nodes and child nodes in the tree structure shown in figure 4-4. In others word, the attributes "upper" and "lower" memorize the aggregation relationship between different UML nodes. The ClassNode has additional attributes "dependencyLink" and "associationLink" which recode the class relationship. The values of these two attributes are list of "xmiId"s. The OperationNode has an additional attribute "parameterList" which is used to save the parameters of one function.
- Storing UML model: because the nodes have different types, nodes will be sorted into different categories, then as figure 4-6 shown, four HashMaps are defined to save different categories of UML nodes. Since the nodes have bidirectional pointer, these four HashMaps are linked into a combination of doubly linked lists. Then, from any node exist

in the HashMaps, it is possible to trace any expected target.

- Choosing UML model: this second round to simplify the UML model after first step cuts out the redundant information. However, this round will not be finished automatically by computer. The program will illustrate entire UML models on the user interface, so that, participants can choose the information they want to or must share with others. As mentioned in section 3.3.2.2, at the phase of generating HLA FOM, class dependencies, associations, and functions may not be interesting, thus this information will not be selected. The similar situation will also happen in the federate code generation. Those situations are the reason of storing UML model in the way shown in figure 4-6. Model elements are individually saved but connected by links. When we decide not to show one category of element, we can break the links and hide these elements. When we want to delete one element, we can erase it from the hash without affecting others, but also erase all the related information by tracing along the links. The algorithm for tracing elements along the links is shown in figure 4-7. This algorithm is a recursive algorithm (Collins, 2005) (Cormen et al., 2003). The principle is to start from one node to check whether it has child nodes. If yes, then travel to those nodes and do the same thing, until a node without child nodes is found (we call it leaf node). Afterwards, operate on this leaf node and then return to the father node, and erase the leaf node from the child nodes list of the father node, which can make the father node into a leaf node. Recursively doing this operation, we can finish the traversal of the link.

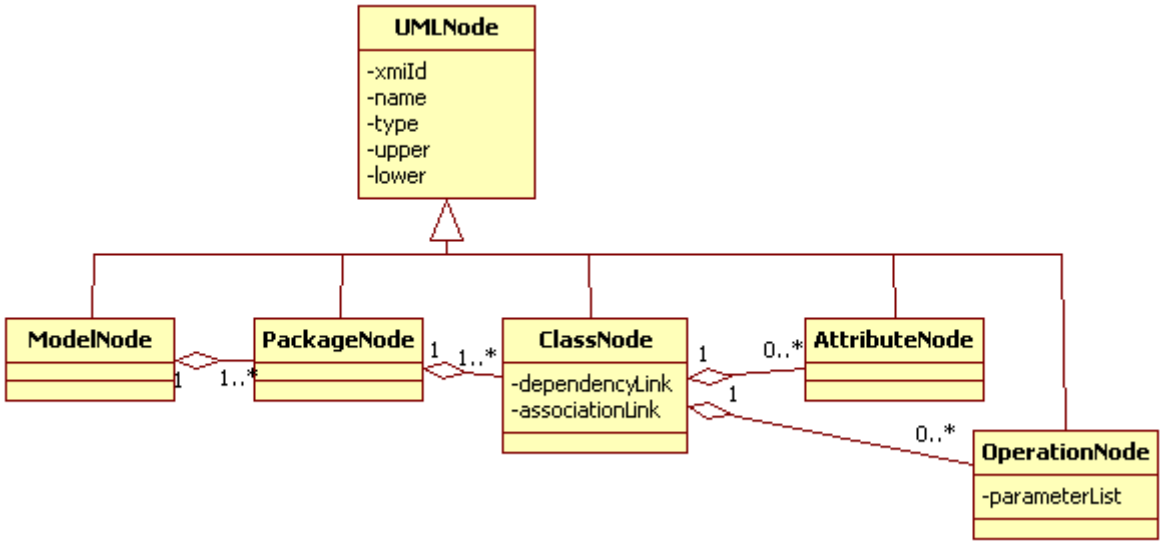

Figure 4-5. Data structure of UML nodes

```
private HashMap<String, PackageNode> packageHash; //Hash for recording packages
private HashMap<String, ClassNode> classHash; //Hash for recording classes
private HashMap<String, AttributeNode> attributeHash; //Hash for recording attributes
private HashMap<String, OperationNode> operationHash; //Hash for recording operations
```

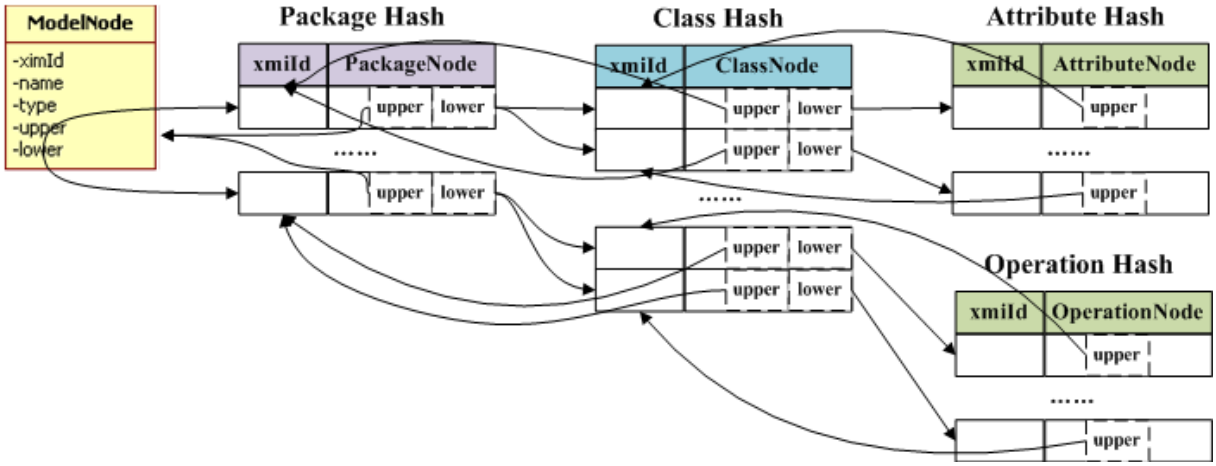

Figure 4-6. UML nodes storage structure

```
/**
* This function's task is to trace any elements on one link
* from one starting node by using recursive algorithms.
* startingNode is the starting point selected
* by the user or recursively chosen by program
*/

Function traceingLinks(UMLNode startingNode)
{
    // to check whether the selected node has child nodes
    while (startingNode->lower is not empty)
    {
        // to get the child nodes list
        List lowerNodeList = startingNode->lower;
        // to examinate each child node by this iteration
        for ( int i = 0; i < Length of lowerNodeList; i++)
        {
            // recursive call
            traceingLinks(lowerNodeList.get(i));
            // update the child nodes list
            erase the lowerNodeList.get(i) from startingNode->lower;
        }
    }
    operate on startingNode; // to operate on the seleceted node
    return; // return to the point of function call or recursive call
}
```

Figure 4-7. The algorithm of tracing elements along the links

4.3.2.2. Model Evolution

As explained earlier, to generate HLA FOM, it has to find the similar entities among the

participants by comparing their UML models, which are the output of the previous step. Even the UML models have already been simplified, but to compare several participants' models at the same time is still a huge project. Thus, the model evolution theory introduced in section 3.3.2.2 can be used to start the third round of UML model simplification.

This sub-module requires manual operation. The participants have to decide the model evolution groups based on their business relationship. Let us take the business relationship shown in figure 4-8 as an example. Enterprise B and C are the subsidiaries of enterprise A, so they are grouped together. Enterprise F is the raw material supplier of enterprise E, meanwhile, the enterprise D provides the product transformation services to enterprise E and F. Thus, enterprise D, E and F are grouped together. The group of G, H and I is set in the same way. As this step, the first round of the model evolution is ready. And then, the enterprises in the same group will start the model alignment which is the mission of the following sub-module. After alignment, each group will have a list of common object models, which are the input of second round of the model evolution. Then, because the enterprise E is the biggest semi-manufactured goods supplier, the group in red and the group in blue will perform model alignment together in the second round of the model evolution. Finally, the output of evolution 2 will perform model alignment with the output of group in purple at the third round of the model evolution, in order to obtain the final common object models for HLA FOM generation.

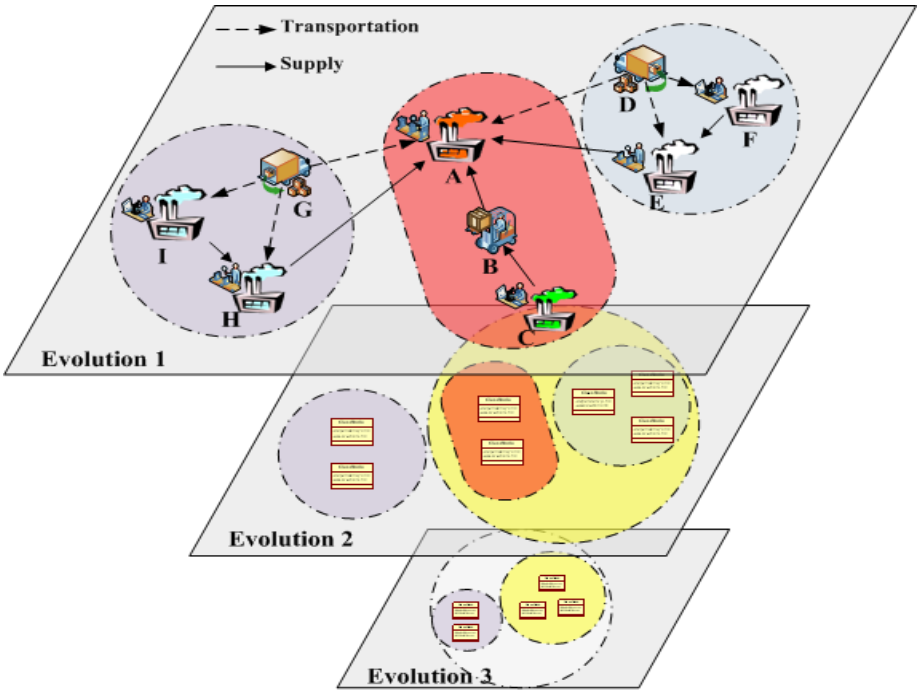

Figure 4-8. Model evolution

As shown in figure 4-8, the participants are connected by binary relation, such as “supply” and “transportation”. The participants and their relations make up a binary relation network that can be represented as a dyadic array. The type of relation will be ignored in the implementation, because we only consider about the customer intimacy, such as, long-term relation or short-term relation.

In order to implement the model evolution, we will assign value to the binary relation between two participants. For example, “(A, B) = 1” means that the binary relation between enterprise A and B is 1. The value range is 1 to N. The exact value of N depends on how many times of model evolution will be performed. This value assignment of each binary relation must be performed at the first two phases of HLA&MDA harmonized development lifecycle mentioned in section 3.2. Phase 1 (Domain requirement definition) must define the customer intimacy of participants. Phase 2 (Domain scenario systematization) must refine these intimacies into corresponding value from 1 to N. these values will be saved in a dyadic array that can be read by model evolution program. The model evolution program will traverse the dyadic array by starting from the main point, such as, the enterprise A shown in figure 4-8, which is the main manufacturer. The program will use the inorder traversal algorithm (Cormen et al., 2003) to discover the participants which are connected by binary relation valued 1, and put them together in one set for first model evolution. The participants, which have binary relation valued 2 with the main point, will join the second model evolution. Successively, the following model evolution will be performed on the participants with binary relation valued 3 to N. For example, if “(A, B) = 1” and “(B, C) = 1”, then enterprise A, B and C can be put in one set for the first model evolution. If (A, E) = 2, then enterprise E cannot be in the same set with enterprise A for the first model evolution, but it will align with the evolution result of enterprise A, B, C in the second evolution.

The inorder traversal algorithm for model evolution is illustrated in figure 4-9. This algorithm is based on recursive algorithm. Program will start from one row-coordinate on the dyadic array (represents one participant), and check all the binary relation values on this row, which represent the binary relation between this participant with others. If the binary relation value equals to 1, then program will recursively call this function with the corresponding column-coordinate to find the participants who have binary relation valued 1 with this participant (because the dyadic array used for saving the binary relations of the participants is dyadic array, the column-coordinate also represents one participant). After this recursive

function stops, the program can obtain a set of participants for model evolution.

```
/**
* This function use recursion to implement inorder traversal of the binary
* relations of the participants, so that the participants
* who should be in the same evolution set with startPoint will be found.
*/

ArrayList findEvolutionSet (int startPoint) {

    ArrayList evolutionSet; // used to save the members for evolution
    evolutionSet.add(startPoint); // add one member to evolutionSet  for evolution

    // relationArray is the dyadic array that saves the binary relations of the participants
    for (int i = 0; i < relationArray[startPoint].length; i++)
        // check the binary relation between startPoint and the participants on the row where startPoint locates
    {
        if (relationArray[startPoint][i] == 1)
        {
            ArrayList setfollowing;
            setfollowing = findEvolutionSet(i); // use recursion to implement inorder traversal
            evolutionSet.append(setfollowing); // append members to evolutionSet  for evolution
        }
    }

    return evolutionSet;
}
```

Figure 4-9. The algorithm of model evolution

4.3.2.3. Model Alignment

The Task of Model Alignment sub-module is to conduct the model evolution from the top level to bottom level. The methodology of the conduction explained in section 3.3.2.2 is to distinguish the similar entities/features among the models from disparate enterprises, align them, and preserve them for the next model generation. So, the prime point of this sub-model is the distinction of the model similarity.

Figure 4-10 illustrates the finite state diagram of the model similarity distinction.

- State 1: The program will firstly require user to import the Simplified UML Model (SUM) file of one group of enterprises, which is the output of analyze UML model sub-model.
- State 2: When all the SUM files are uploaded, user has to define the Expected Transmission Similarity value for this distinction.
- State 3: Afterwards the program will illustrate the SUM matrix and turn to the state of waiting user enters the similarity for model pair. If there is not suitable pair for definition, then this distinction program will stop.
- State 4: When program detects the possibility of similarity transmission, it will highlight the possible pairs. And then, user can check these pairs to confirm or deny this possibility. If the possibility is confirm, the program will detect the possibility of similarity

transmission again, and highlight more possible pairs, until all the possibility are confirmed or denied. After that, the program will turn back to the state 3 to wait for the definition from user.

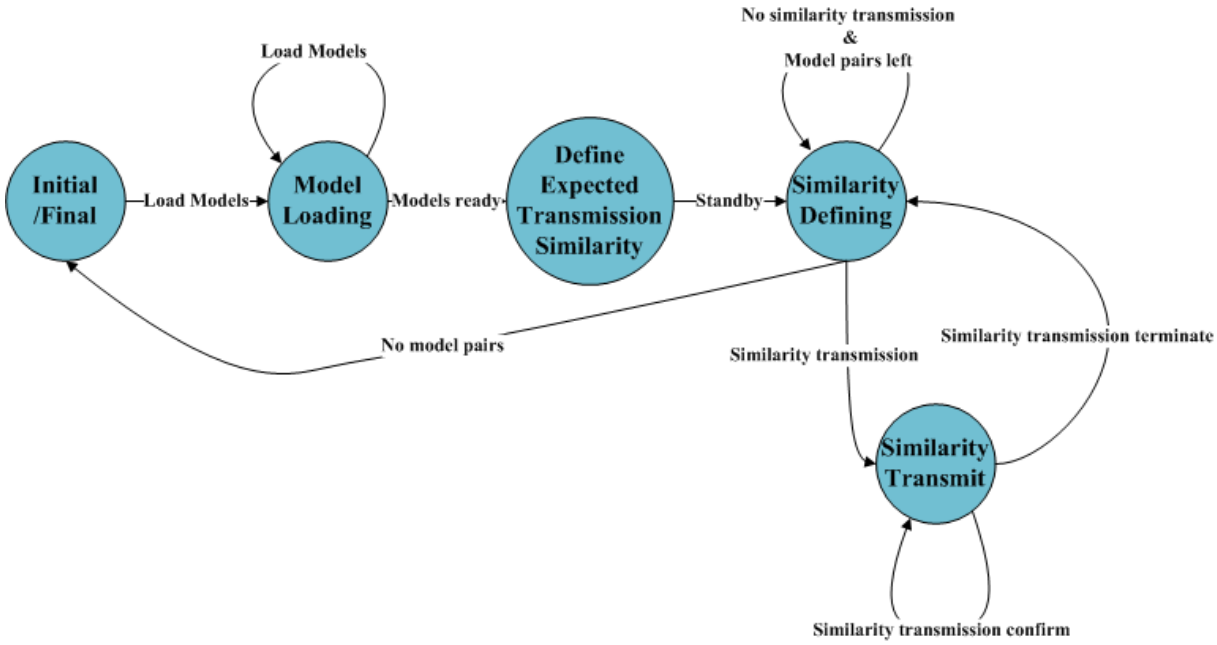

Figure 4-10. Model similarity distinction state chart

The similarity transmission detection is an automatic process following the algorithm shown in segment “Model Alignment” of section 3.3.2.2. For example, user has firstly defined the similarity of the pair of class S and T, and then defined the pair of class T and G as shown in figure 4-11. Then, according to the relation transmission theory, it is possible to transmit the similarity to the pair S and G. However, the program has to calculate the Transmission Threshold Value based on the user input and the Expected Transmission Similarity value, so that the program can accept this possibility. It means that if the similarity of class T and G defined by user is beyond the blue line shown in figure 4-11, then the program will highlight the cell of class S and G on the matrix to the user.

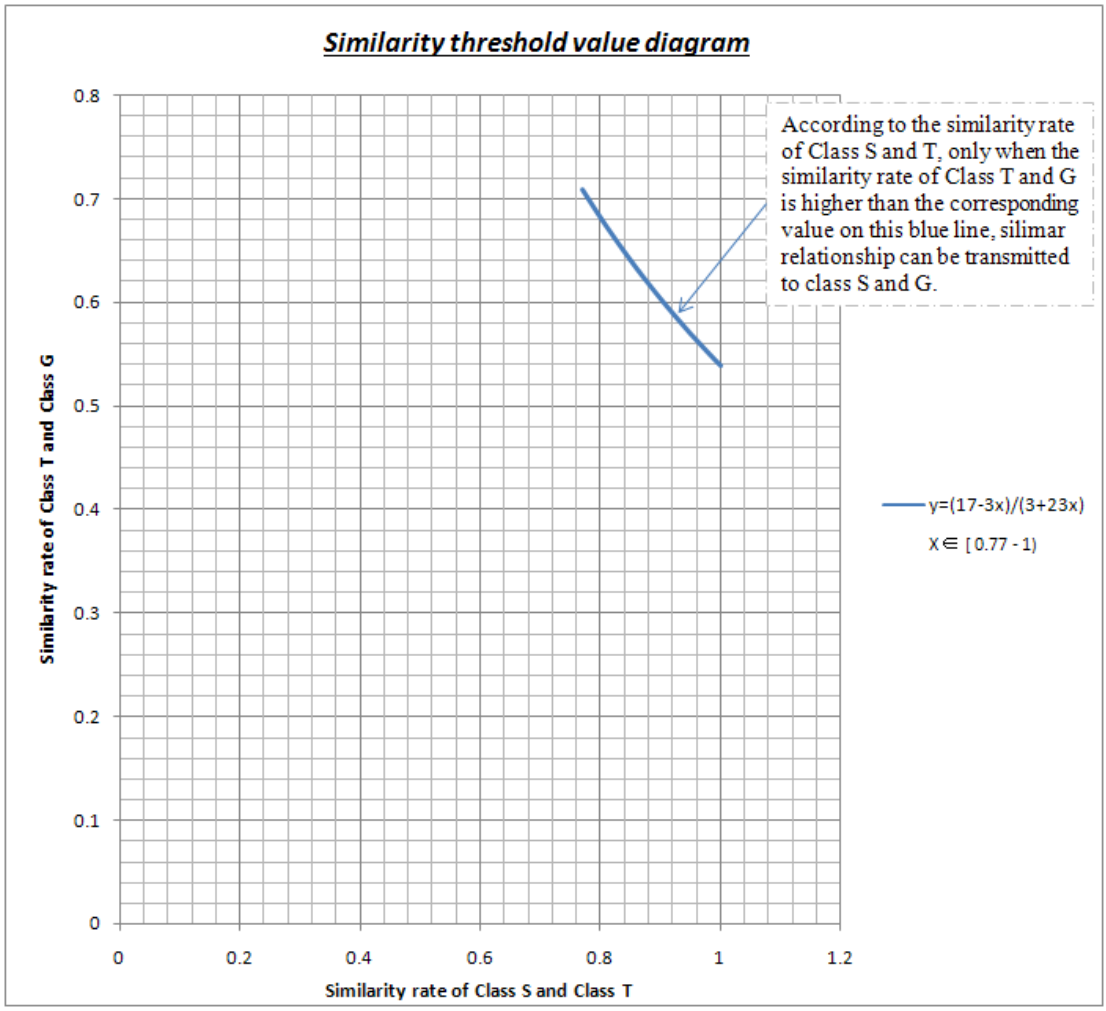

Figure 4-11. Calculation of Transmission Threshold Value

The figure 4-12 illustrates the pseudo code of the similarity transmission detection with the example shown in figure 4-11. When user defines the similarity for one pair of models, the program will call the function of highlightSimiPossibility with parameters of row-coordinate and column-coordinate of the relation matrix, and defined similarity value. Then, the program will iteratively check the pairs of the matrix on the same row to discover the pairs of models with similarity value. Because the defined the expected similarity is 70%, the first constraint for the existing similarity value (x) is [0.77, 1), and the equation of calculating Transmission Threshold Value (y) is (17 – 3x) / (3 + 23x). Thus, if the x of Matrix[row][i][18] satisfies the first constraint, its value will be used to calculate the value of y. Finally, if the similarity value of the new defined pair equals to or is higher than y, the program will highlight the table cell whose row-coordinate is i and column-coordinate is col[19].

[18] the Matrix is a dyadic array that represents the relation matrix of participants' models. “row” is the row-coordinate of the new defined pair. “i” is the column-coordinate of the current pair that is checking.
[19] “col” is the column-coordinate of the new defined pair.

```
/**
* This function is to find the possibility to transmit the model similarity
*/

void highlightSimiPossibility(int row, int col, double value){
    // the Matrix is a dyadic array that represents the relation matrix of participants' models
    Matrix[col][row] = value; // the Matrix must be symmetric, so copy the value to corresponding cell

    for (int i =0; i < Matrix[row].length; i++)
        // check the cell on the same row, to discover the pairs of models with similarity value
    {
        int x = Matrix[row][i]; // get the similarity value
        if(x >= 0.77 && x < 1) // check the first constraint of transmission possibility
        {
            int y = (17 - 3*x)/(3 + 23*x); // calculate the Transmission Threshold Value that is corresponding to x
            if (value >= y)
            {
                // if equal to or higher than Transmission Threshold Value,
                // then call the function to highlight to table cell whose row-coordinate is i and column-coordinate is col
                highlightTablecell(i, col);
            }
        }
    }
}
```

Figure 4-12. The pseudo code of the similarity transmission detection

4.3.2.4. FOM generation

After the model alignment, the program will illustrate the similar model list to the user, so that user can select the useful attributes of those models, and then generate a new common model and rename it. While the model evolution is finished, throughout many times of model alignment, the program has generated a list of new common models. Afterwards, user can transfer this list of models into HLA FOM file by following different RTI FOM file format. Because HLA FOM also obeys to the object oriented principle, this model transformation is only a syntax transformation, but not a semantic one. For example, according to the portico RTI FOM file format, UML:Class matches to Objects: Class, and UML: Attribute matches to Objects: Attribute.

4.3.3. Generate HLA Federate Code Block

As mentioned in section 3.3.2.3, we will use program tracer tool to discover all the possible execution paths of the existing system, and then merge the paths into a directed graph. Afterwards, the directed graph will be reduced for generating state diagram. As mentioned, this method has not been completely implemented, because of time constraint of my PhD research. This method requires numerous experimental data to define general system behaviour and rules for execution paths reduction, which we did not obtain enough. This section will introduce the implemented part of this method.

4.3.3.1. System traversal

The system traversal is the phase to obtain all the possible system execution modes. This phase needs to do many experiments on diverse systems, so that different system behaviours can be discovered to identify different reduction scenarios. The selected program tracer tool for obtaining experimental data is Jprofiler. The trial version of this tool can be downloaded from http://www.ej-technologies.com/index.html. The online help for configuring and using this tool can be found in (JProfiler, 2012). We have used Jprofiler to trace the execution of many systems. The figure 4-13 shows an example of the result of Jprofiler. Jprofiler can show all the possibilities of system execution by following every function call. As the first step of model reverse method has discover the UML models from the existing systems, each function call can be re-tracked back to its class. In that case, the Jprofiler result can be transform into the execution paths shown in figure 3-15.

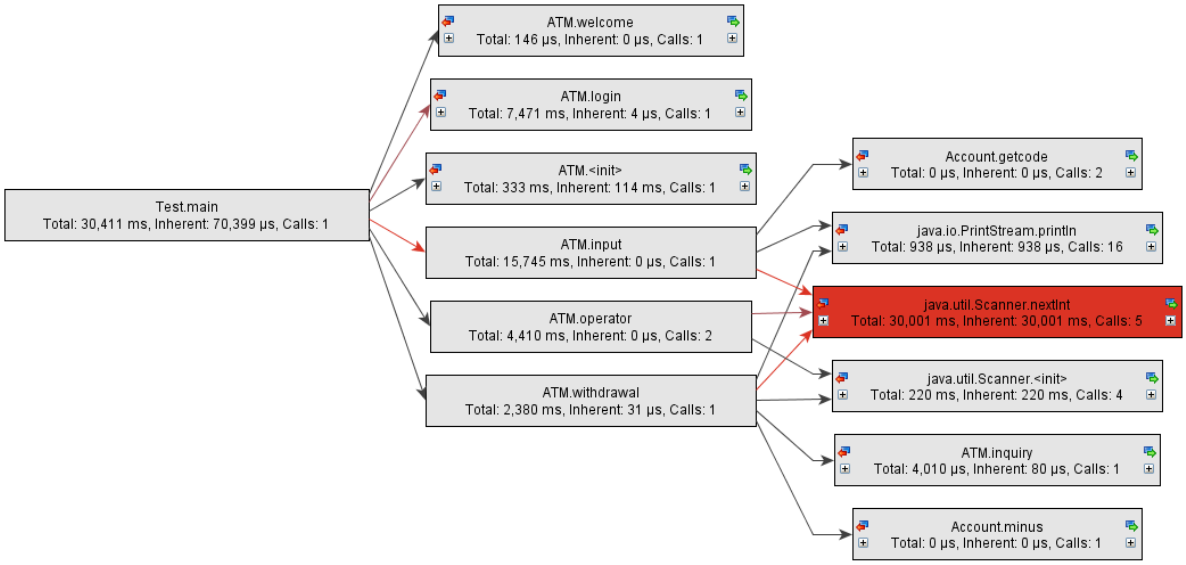

Figure 4-13. Jprofiler result

The figure 4-14 illustrates the representative structure of the node of the execution path. Each node has three attributes, className as unique identity, functionCallList for saving function call originated from this node, and objectValue for saving the object values of instances created by different function calls.

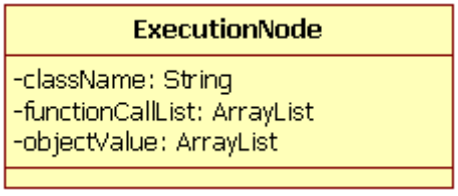

Figure 4-14. Execution Node

4.3.3.2. Model Processing

Model processing phase is responsible for reducing the system execution paths. This phase requires reduction rules to merge the nodes on different execution paths. Currently, we have defined four rules.

- Rule 1: Completely same. If the nodes have same class name, function calls and object values, they will be merged immediately.
- Rule 2: Objects with different values. If the nodes have same class name and function calls, but different object values, they will be considered as similar. They will still be merged, but different object values will be saved and matched to different function calls that triggered by different values. It means that this merged node has value assertion that might cause the system state change.
- Rule 3: Lack of function calls. If the nodes have same class name, but different function calls, they will be consider as similar. They will still be merged, but different function calls will be appended on the functionCallList. It means that this merged node has different options to change the system states.
- Rule 4: Simple loop reduction. As shown in figure 4-15, node B and node C are called by one same node, node A, and they will also call another same node, node D. In that case, these four nodes form a simple loop. This simple loop will be reduced into a simple path by merging node B and Node C and the edges of the directed graph. The functionCallList must be merged into one list. ObjectValue must also be saved in one list. The className of the merged node must be merged as well. Node A must match the function calls to the corresponding system handlers with node B and C.

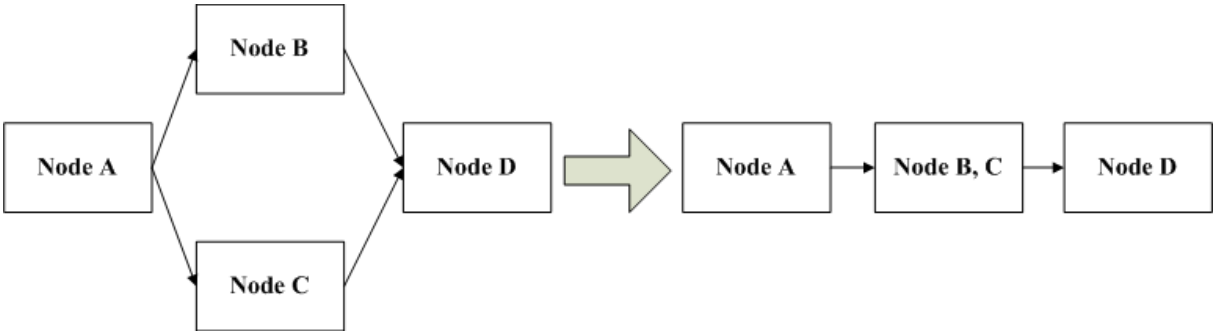

Figure 4-15. Simple loop reduction

4.3.3.3. Behaviour Model Generation

The phase of behaviour model generation is responsible for transforming the reduced directed graph of the system execution paths into state diagram. According to the method introduced in section 3.3.2.3, one algorithm for generating state diagram is provided as figure 4-16 shows.

This algorithm is a recursive algorithm. The program starts from the first point of one directed graph of the system execution paths. Then, it calls stateGeneration function recursively for each node on the execution path. The stateGeneration function has a return type of ArrayList that is used to save system states. The system state is represented as a class structure called StateNode, which consists of two attributes, nodesList and stateTransitionList. The nodesList is used to save the nodes of execution path which will be put together in one state. The stateTransitionList is used to save state transitions which are the function calls of the last node in one state. The stateGeneration function requires two parameters, executionNode and sameState. The executionNode is an instance of ExecutionNode class, which represents the current node for states discovery. The sameState is a list that saves the nodes which belong to the same state.

```
/**
* This function's task is to generate state diagram
*/

ArrayList stateGeneration (ExecutionNode executionNode, ArrayList sameState) {

    ArrayList stateList = new ArrayList(); // stateList is used to save state diagram
    StateNode stateNode = new StateNode(); // stateNode represents state, contains the set of nodes and transition

    int callNumber = executionNode.functionCallList.size(); // get function call number

    // if function call number is 0, it means that state terminate
    // if current node has been check, it means that there is a recall
    if (callNumber != 0 && executionNode does not exist in the stateList){
        if (callNumber == 1) { //if function call number is 1, it means that the coming node will be in the same state
            sameState.add(executionNode);
            ExecutionNode newNode = nodeMap.get(executionNode.get(0).split(@)); // get next node
            ArrayList newStateList = stateGeneration (newNode, sameState); // recursively call this function
            if (newStateList == null) { // if newStateList is null, it means that there is a recall, this node is end of one state
                //generate one state
                stateNode.addNodeList(sameState);
                stateNode.addTransition(executionNode.functionCallList.get(0));
                stateList.add(stateNode);
                return stateList;
            }
            return newStateList;
        } else { // if function call number is more than 1, it means that the coming node will be in the other state
            //generate one state
            sameState.add(executionNode);
            stateNode.addNodeList(sameState);
            stateList.add(stateNode);
            // start to generate the other state
            for (int i = 0; i < callNumber; i++) {
                stateNode.addTransition(executionNode.functionCallList.get(i));
                ExecutionNode newNode = nodeMap.get(executionNode.get(i).split(@)); // get next node
                ArrayList newStateList = stateGeneration (newNode); // recursively call this function
                //generate states
                for (int j = 0; j < newStateList.size(); j++) {
                    StateNode newStateNode = new StateNode();
                    newStateNode.addNodeList(newStateList.get(j));
                    stateList.add(newStateNode);
                }
            }
            return stateList;
        }
    } else {
        if(executionNode exists in the stateList){
            return null;
        } else {
            //generate one state
            sameState.add(executionNode);
            stateNode.addNodeList(sameState);
            stateList.add(stateNode);
            return stateList;
        }
    }
    return null;
}
```

Figure 4-16. State diagram generation algorithm

The program firstly checks the number of function calls of the executionNode, and whether the executionNode is repetitively checked. This checking will decide different cases of the executionNodes alignment for generating a state.

- If the number of function calls is 0, it means that it is the end of one sub-path of system execution. The nodes in the sameSate list must be merged into one state.
- If the executionNode is repetitively checked, it means that it is a recall loop. If the sameSate is not null, the nodes inside must be merged into one state. If the sameState is null, the function call (state transition) will be saved in the previous stateNode.
- If the number of function calls is 1, it means that the next executionNode will be in the same state. The function will save the current executionNode in the sameState, and then recursively call this function with the next executionNode.
- If the number of function calls is bigger than 1, it means that system will turn to different states. The function will merge the nodes in the sameSate list into one state, and put the function calls into the stateTransitionList of one instance of class StateNode. Afterwards, the function will iteratively check each of the next executionNodes that could be called by the current executionNode.

Finally, this stateGeneration function returns a list of instances of class StateNode. Each instance saves a set of the nodes on the execution path for mapping to the system classes (as system handlers). Each instance also has a list of function calls that are used to represent the state transitions.

4.4. Run Time

Run time is the HLA execution time. The Build Time I has prepared HLA FOM and Federate Code, which are the essential parts of HLA federation execution and the single federate. In other words, the output of the build time I accelerates the establishment of HLA federation environment, and the code generation of the HLA federate Interface for each participant.

The class diagram of the establishment of run time is illustrated in figure 4-17. In this simulation, there are four kinds of federates: federate for central control, federate for Web-enable, federate for initial participant (inside traditional HLA federation), and federate for potential participant (outside traditional HLA federation, but connected via web services).

As mentioned in section 2.2, no matter what kinds of federates need RTI::RTIAmbassador and RTI::FederateAmbassador to implement the basic communication functions inside the LibRTI, such as callback function. Thus, class Federate has an association with a class RTIFactoryFactory which can create the RTIAmbassador and execute the HLA federation or join the federation. Class Federate will also extend class FederateAmbassador, so that it can implement the management of the message exchange.

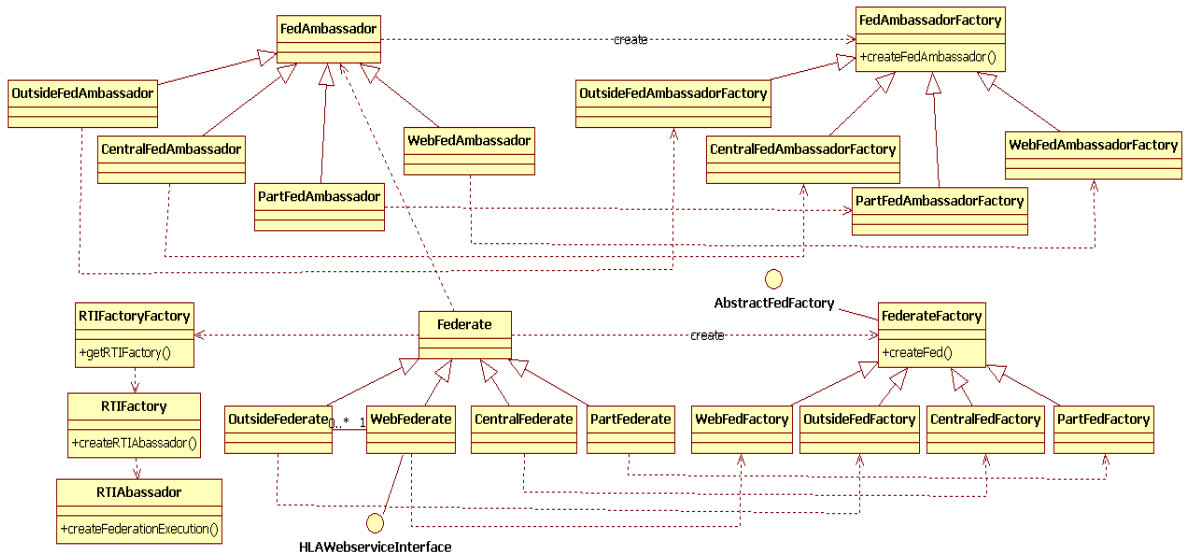

Figure 4-17. Class diagram of run time

Meanwhile, because the participants have different interests in this collaboration, so the representative federate of each participant will share different messages and have different ways of sharing them. Therefore, each federate has a correlated federate ambassador. At the meantime, each participant has different business processes and partnerships, so each representative federate must have different behaviours, and it will publish different objects and subscribe to different objects. In order to implement this diversification, the factory design pattern (Cooper, 2000) is used to allow the agile creation of FederateAmbassadors and federates. This factory pattern helps to reuse the fundamental RTI related functionalities and assist the Build Time I to automatically generate diverse federates.

As shown in figure 4-17, there are four sub classes of the abstract class “Federate”, Class CentralFederate, Class PartFederate, Class WebFederate, and Class OutsideFederate. The functionalities and features of these sub classes are listed as follows:

- Class CentralFederate is implemented as the monitor of the HLA federation environment. It has to maintain the execution status of the online federates, and recode their active time period. Besides that, it also needs to report and log any exceptions happen in the run-time, in order to conduce to the good maintenance of the HLA federation environment.
- Class PartFederate can be implemented into diverse federates based on the different Enterprise Business Behaviour Interfaces (adaptors mentioned in section 3.2.2.2) generated by sub module HLA federate code generation of Build Time I.
- Class WebFederate is implemented as the Web Services server that publishes HLA federation services. Thus, besides the inheritance from abstract class "Federate", the WebFederate also needs to implement the HLA WebServiceInterface. The potential participants will be synchronized with traditional federates via this federate. In fact, The WebFederate has to be separated into two classes, WebservicesBridge and WebservicesServer, because of the elected RTI – portico RTI. The federate inside the HLA network environment cannot use local IP to connect with the web user. The WebservicesBridge will be deployed inside the HLA network environment to communicate with other federates via RTI, and the WebservicesServer will be deployed outside it to preside over the web communication, but they will be linked through the socket connection. The deployment of federates is illustrated in the figure 4-18.
- Class OutsideFederate is implemented as the federate interface for the potential participant (from web). This federate will be explain in detail in next section.

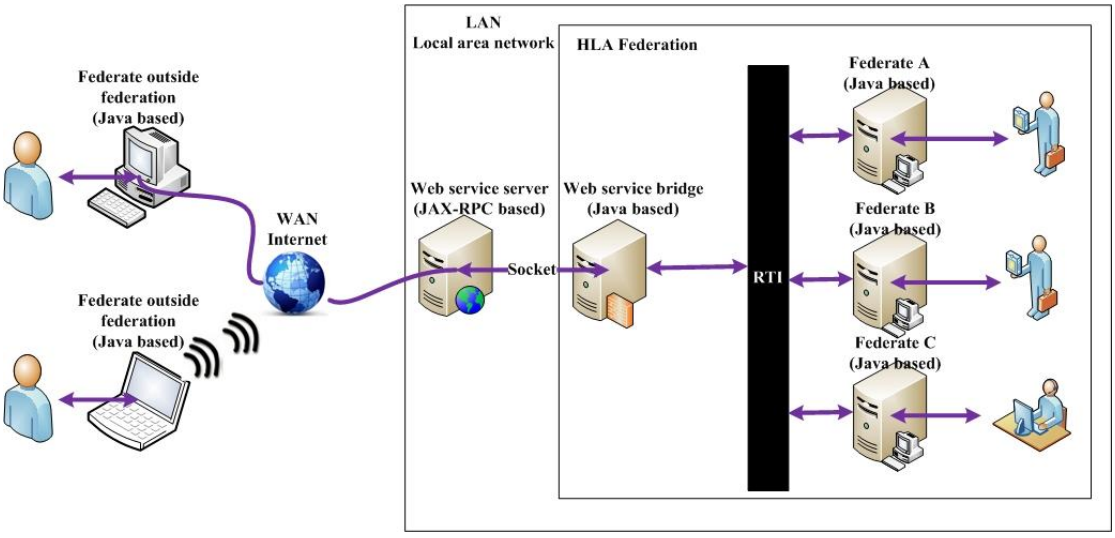

Figure 4-18. The deployment of federates

4.5. Build Time II

The Build Time II is responsible for interoperability establishment between potential participants and the existing interoperability environment. Thus, the main task of Build Time II is to discover models from legacy systems of new participants, and match these models to the existing ones, which is corresponding to the second model reverse scenario mentioned in section 3.3.

The Build Time II will be performed during the HLA federation execution time. After the previous two steps, the HLA federation environment is ready for the participants to start their interoperation. As the traditional HLA principle, this HLA federation is completed and cannot be changed. In order to welcome the potential participants, the WebservicesServer has been designed according to the HLA evolved principle. Thus, even the HLA federation has closed its port to the unexpected federate, the web users can download the WSDL (Web Services Description Language) file from the Services Broker, and generate the OutsideFederate to communicate with WebservicesServer.

After the Build Time I generate the initiative information of the HLA federation, the program will generate this information in the Web Services (as the red dash-line named “web services generation” shown in figure 4-1). For example, the objects in the FOM file will be transferred into service items, and the interactions will be transferred into service functions. The web users can tick the service items and functions that they are interested in, so that their OutsideFederates can subscribe to the objects and interactions of HLA FOM. Meanwhile, as the web users have different authorities to access the HLA federation, the service items and functions will also be constrained by the WebservicesServer. The message from OutsideFederates will be synchronized with other traditional federate through the WebservicesBridge. The figure 4-19 shows the code segment of WebServicesBridge. The WebServicesBridge will run a thread for one participant from web. One thread will create a socket to wait for the message from WebservicesServer. After the message processing, WebServicesBridge will call the callback functions of the federate ambassador to update the attribute, send the interaction, and request time advance.

```
thread.start();// run one thread for monitoring

ServerSocket svrsoc = null;
Socket soc = null;
DataInputStream in = null;
PrintStream out = null;
InetAddress clientIP = null;
String str = "";
try {

    svrsoc = new ServerSocket(8000); // set up a socket for monitoring
    System.out.println("Server start....");
    soc = svrsoc.accept();

    in = new DataInputStream(soc.getInputStream()); // get information from WebServiceServer
    str = in.readLine();
    while (!"quit".equals(str) && !"".equals(str)) { // analyse the information
        out = new PrintStream(soc.getOutputStream());
        clientIP = soc.getInetAddress();

        String[] strSplit = str.split(";");

        while (true)
        {
            /**
            * message processing......
            */

            // update the attribute values of the instance //
            updateAttributeValues();
            // send an interaction
            sendInteraction();
            // request a time advance and wait until we get it
            Thread.sleep(1000);
            advanceTime(1.0);
            log("Time Advanced to " + fedamb.federateTime);
        }
        svrsoc.close();
        svrsoc = new ServerSocket(8000);
        System.out.println("new waiting start....");
        soc = svrsoc.accept();
        in = new DataInputStream(soc.getInputStream());

        str = in.readLine();
    }
} catch (Exception e) {
    // exception treatment......
}
```

Figure 4-19. The code segment of WebServicesBridge.

In addition, the web participants need to build up their own local ontology glossary for information analysis. It is a part of the implementation of short-lived ontology. As mentioned in section 3.5.4, the short-lived ontology method has been completely implemented. It has also been opened as the future work that is the research subject of a new PhD candidate in our research group. Thus, we use his ontology alignment approach (Song et al., 2012) to help the local ontology glossary implementation.

After model evolution and model alignment, a set of similar models are obtained, which can be encapsulated into an ontology in format OWL (Ontology Web Language) as shown in the part A of figure 4-20. The reversed models of web participant can also be generated into an ontology as shown in the part B of figure 4-20. In the generated ontology, it mainly contains the classes and subclass as the XML's structure represents. The properties are represented as descriptors.

```
<owl:Class rdf:ID="ProductInfo">
  <owl:equivalentClass rdf:resource="&Product;ProductInformation"/>
 </owl:Class>
<owl:Class rdf:ID="ProductInfoDescriptor" />
<owl:Class rdf:ID="ProductID">
  <rdfs:subClassOf rdf:resource="#ProductInfoDescriptor" />
</owl:Class>
<owl:Class rdf:ID="ProductName">
  <rdfs:subClassOf rdf:resource="#ProductInfoDescriptor" />
</owl:Class>
<owl:Class rdf:ID="ProductCategory">
  <rdfs:subClassOf rdf:resource="#ProductInfoDescriptor" />
</owl:Class>
<owl:Class rdf:ID="Price">
  <rdfs:subClassOf rdf:resource="#ProductInfoDescriptor" />
</owl:Class>
<owl:Class rdf:ID="Location">
  <rdfs:subClassOf rdf:resource="#ProductInfoDescriptor" />
</owl:Class>
<owl:Class rdf:ID="InventoryNumber">
  <rdfs:subClassOf rdf:resource="#ProductInfoDescriptor" />
</owl:Class>
<owl:ObjectProperty rdf:ID="hasProductInfoDescriptor">
  <rdfs:domain rdf:resource="ProductInfo" />
  <rdfs:range  rdf:resource="#ProductInfoDescriptor" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasProductID">
  <rdfs:subPropertyOf rdf:resource="hasProductInfoDescriptor" />
  <rdfs:range rdf:resource="#ProductID" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasProductName">
  <rdfs:subPropertyOf rdf:resource="hasProductInfoDescriptor" />
  <rdfs:range rdf:resource="#ProductName" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasProductCatergory">
  <rdfs:subPropertyOf rdf:resource="hasProductInfoDescriptor" />
  <rdfs:range rdf:resource="#ProductCatergory" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPrice">
  <rdfs:subPropertyOf rdf:resource="hasProductInfoDescriptor" />
  <rdfs:range rdf:resource="#Price" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasLocation">
  <rdfs:subPropertyOf rdf:resource="hasProductInfoDescriptor" />
  <rdfs:range rdf:resource="#Location" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasInventoryNumber">
  <rdfs:subPropertyOf rdf:resource="hasProductInfoDescriptor" />
  <rdfs:range rdf:resource="#InventoryNumber" />
</owl:ObjectProperty>
......
```

(A) Ontology Description of Federation

```
<owl:Class rdf:ID="CargoInfo"/>
<owl:Class rdf:ID="CargoInfoDescriptor" />
<owl:Class rdf:ID="CargoID">
  <rdfs:subClassOf rdf:resource="#CargoInfoDescriptor" />
</owl:Class>
<owl:Class rdf:ID="CargoName">
  <rdfs:subClassOf rdf:resource="#CargoInfoDescriptor" />
</owl:Class>
<owl:Class rdf:ID="CargoClassification">
  <rdfs:subClassOf rdf:resource="#CargoInfoDescriptor" />
</owl:Class>
<owl:Class rdf:ID="Price">
  <rdfs:subClassOf rdf:resource="#CargoInfoDescriptor" />
</owl:Class>
<owl:Class rdf:ID="InventoryLevel">
  <rdfs:subClassOf rdf:resource="#CargoInfoDescriptor" />
</owl:Class>
<owl:ObjectProperty rdf:ID="hasCargoInfoDescriptor">
  <rdfs:domain rdf:resource="CargoInfo" />
  <rdfs:range  rdf:resource="#CargoInfoDescriptor" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasCargoID">
  <rdfs:subPropertyOf rdf:resource="hasCargoInfoDescriptor" />
  <rdfs:range rdf:resource="#CargoID" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasCargoName">
  <rdfs:subPropertyOf rdf:resource="hasCargoInfoDescriptor" />
  <rdfs:range rdf:resource="#CargoName" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasCargoClassification">
  <rdfs:subPropertyOf rdf:resource="hasCargoInfoDescriptor" />
  <rdfs:range rdf:resource="#CargoClassification" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPrice">
  <rdfs:subPropertyOf rdf:resource="hasCargoInfoDescriptor" />
  <rdfs:range rdf:resource="#Price" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasInventoryLevel">
  <rdfs:subPropertyOf rdf:resource="hasCargoInfoDescriptor" />
  <rdfs:range rdf:resource="#InventoryLevel" />
</owl:ObjectProperty>
......
```

(B) Ontology Description of new participant

Figure 4-20. OWL ontology example

In order to find equivalent concepts between the ontologies of existing HLA federation and new participant, ontology matching will be performed with a multi-strategies-based approach (Song et al., 2012). In this approach, two source ontologies are the inputs.

- Firstly, a pre-process will be carried out to eliminate and tokenize source ontology into single elements.
- Secondly, for each pair of elements, a strategy will be applied to select one or more suitable matchers. There are three matchers (Song et al., 2012) are used in the approach from different aspects of source ontology: string, structural and semantic. Each selected

matcher will generate one similarity value.

- Thirdly, in order to aggregate different matching results, an analytic method with AHP (Analytic Hierarchy Process) (Song et al., 2012) is adopted to learn the weight of each matcher. The process is based on three similarity indicators, which could reflect the essential features of source ontology, to assign the intensity of importance when measuring the criteria against the goal. A final correspondence will be generated with the learned weights.

A threshold can be used to filter the discovered alignments. When the similarity is greater than the threshold, the alignments are kept, otherwise, the alignments are considered as invalid. A final correspondence is defined as $\{e_1, e_2, r, v, id\}$, where e_1 and e_2 are two identified elements with relation r and similarity value v and a unique identifier id. With constructs built-in OWL, the equivalent links will be setup.

4.6. Conclusion

This chapter has described the mechanism for implementing the Harmonized and Reversible HLA based methodology. Almost all methods introduced in chapter 3 have been implemented, but the behaviour model reverse method and the short-lived ontology method have been partially implemented. The architecture for implementing the model driven and HLA based Reverse engineering tool has been elaborated. To sum up this architecture, it consists of three parts (as shown in figure 4-21), build time I, run time, and build time II. Each part has difference tasks of modelling and simulation.

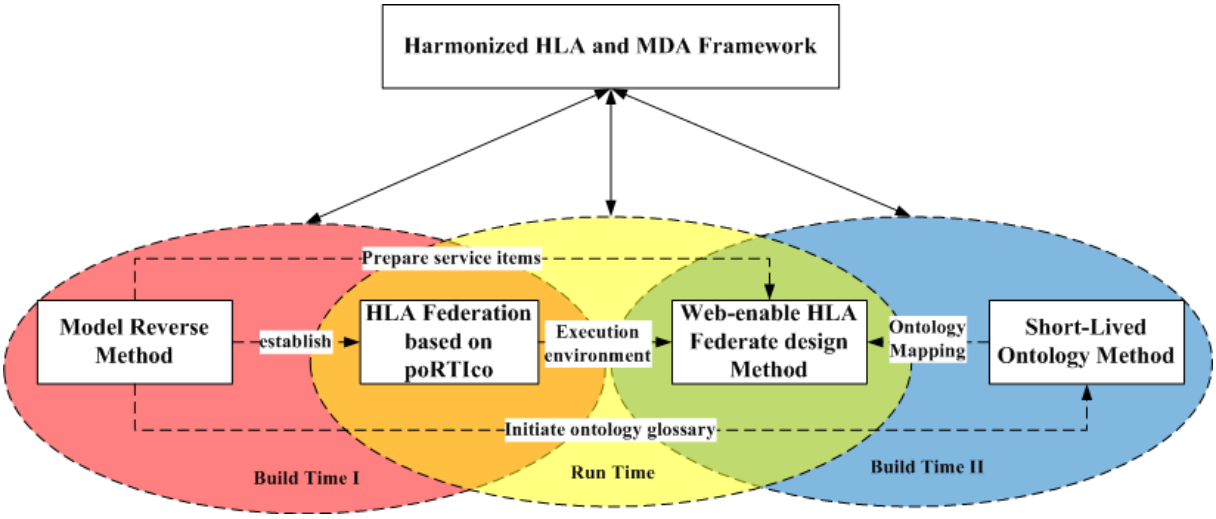

Figure 4-21. Inter-relationships among modules

- The part of build time I is the primary phase. It must implement the model reverse method and development the HLA Federation based on poRTIco RTI. Model Reverse Method includes model reversal, model adjustment, and target model & code generation. It is responsible for preparing simulation environment of Enterprise Interoperability, which concerns rapid and dynamic interoperability establishment. It is also responsible for preparing services items for web-enable federate development, and initiating ontology glossary for web participants, which aim at implementing agile environment compatibility, and the collaboration environment management.
- The part of build time II is a flexible phase. It only performs when a new participant wants to join from the web. The task of this part is to implement agile environment compatibility that allows web participants to join the collaboration as “plug-and-play”. This part consists of web-enable HLA federate design method and short-lived ontology method. As mentioned earlier short-lived ontology method is partially implemented. The implemented part of this method is used in this phase to help the web participants to initiate their local ontology glossary.
- The part of run time is for simulation, including message dispatch and management. It concerns transient information exchange and analysis. Meanwhile, the generation and connection of web-enable federate happens also in the run time.

Chapter 5. Case study

5.1. Introduction

As mentioned in chapter 4, we have developed a model driven and HLA based reverse engineering tool. This section will present a case study of using this tool based on laboratory data. This case aims at illustrating the feasibility of the methodology mentioned in chapter 3, and the efficiency of the implementation introduced in chapter 4. This case describes a scenario of car purchasing and car manufacturing. This scenario includes customer ordering, car manufacturing, material purchasing/delivering, and product delivering. The actors of this case are car manufacturer, clients, semi-manufactured goods/ automobile parts suppliers, raw material suppliers, and potential participants. The actions of this case are collaboration establishment of car manufacturer and suppliers, goods order and distribution between car manufacturer and clients, material purchasing and delivering between car manufacturer and suppliers, and collaboration establishment of potential participant and existing members. The goal of this case is to use the federated approach to achieve efficient establishment of interoperability environment, rapid order dispatch, intelligent information analysis and easy pass to the federation for clients and new participants.

5.2. Demonstration

This section will demonstrate the simulation of the case described in the previous section. We will specify and illustrate the functionality of each module. This section is organized according to the usage of model driven and HLA based reverse engineering tool.

- Firstly, section 5.2.1 will present to harmonization of MDA and HLA FEDEP, which defines the development lifecycle, and requirements.
- Secondly, section 5.2.2 will present the build time I part of this tool. This section will illustrate how to implement model reverse method to prepare the establishment of HLA Federation for simulating enterprise interoperability.
- Thirdly, section 5.2.3 will present both run time part and build time II part of this tool. This section will illustrate how to simulate enterprise interoperability among participants in the HLA federation. This section will also explain how web participants use the web services to generate their federate to join existing HLA federation.

5.2.1. Harmonization of MDA and HLA FEDEP

As mentioned in section 3.2, the harmonization of MDA and HLA FEDEP supports standardization & modularization design and development of the federated approach presented in this thesis. Thus it is the guideline of the enterprise interoperability requirement analysis and the enterprise interoperability environment establishment.

- Phase 1: Domain requirement definition - The scenario of this case can be simply decomposed into (1) cars purchasing and distribution between car manufacturer and client, (2) material/ automobile parts purchasing and delivering among car manufacturer, semi-manufactured goods/ automobile parts supplier and raw material supplier, who are the initial members of this interoperation, and (3) alignment establishment between potential participant/ customers and existing members. The scenario (1) requires that clients can send the car purchasing order and trace the order process, while, the car manufacturer has to produce the car following the customer requirement and make the order process public to the clients. The scenario (2) is the common commercial processes happen when car manufacturer or semi-manufactured goods/ automobile parts supplier is in the situation of stock shortage, but, this case requires these processes efficient and low cost. The scenario (3) of this case is swift link and self-adjusted link for web users (potential participant and customers).
- Phase 2: Domain scenario systematization - as defined in the first phase, there are four main entities, car manufacturer, client, semi-manufactured goods/ automobile parts supplier, raw material supplier and potential participant. They are connected by various kinds of request and response functions. The use case diagram (figure 5-1) illustrates these entities and their relationship.
- Phase 3: System model specialization - In this case, HLA and JAVA are chosen, the scenario defined in the previous phases needs to be transferred into HLA and JAVA models. In this case, the car manufacturer and suppliers are the sponsors of the HLA federation, who start this collaboration. The clients and the potential participant will join this collaboration later after the HLA federation has been established, so they will be the HLA Evolved federate mentioned in section 3.4. All the actors join this collaboration for certain reasons such as to sell their products or purchase products, so their IT systems have their own objects for representing their interests. However, because of the federate approach, it is not proper to define common attribute for the objects of everyone's

"interests". Thus, in the HLA context, a JAVA object of HLA message is defined for "interests" with attributes of sender, receiver and message context. All the members are connected by various kinds of request and response functions, so different HLA interaction classes are defined for these function, such as, the "Material Delivering" HLA interaction class. Inside these classes, there are the parameters of HLA specific timestamp and message object defined before.

- Phase 4: System Implementation - the objects and interaction classes defined in phase 3 will be generated into HLA FOM file, so that it will be part of the plug-in ("integration code") mentioned in section 3.2.
- Phase 5: Test - while giving the definition of the previous four phases, the test case will be prepared and be used in each phase's evaluation and final validation.

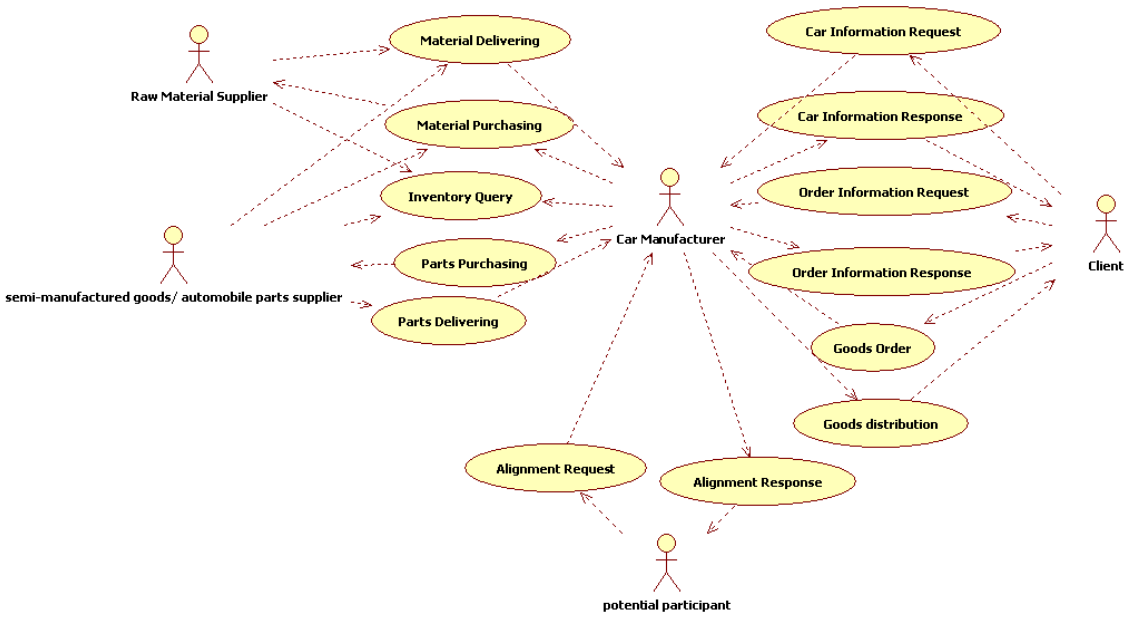

Figure 5-1. Use case diagram

5.2.2. Build Time I

5.2.2.1. HLA FOM generation

Analyze UML

After using the Modisco Tool, a .uml file is generated with the format mentioned in section 4.3.1 (the detail of the .uml file is shown in Annex 1). As mentioned in section 4.3.1, only part of the information of the .uml file is useful for the HLA FOM generation. Thus, the

Model driven and HLA based Reverse Engineering Tool provides a module called “UML Reader and Analyst” (as shown in figure 5-2) to load the UML information. This module will load the UML class information and illustrate it into the hierarchy, “root -> package -> class -> attribute”, as the column name of the table shown in figure 5-2.

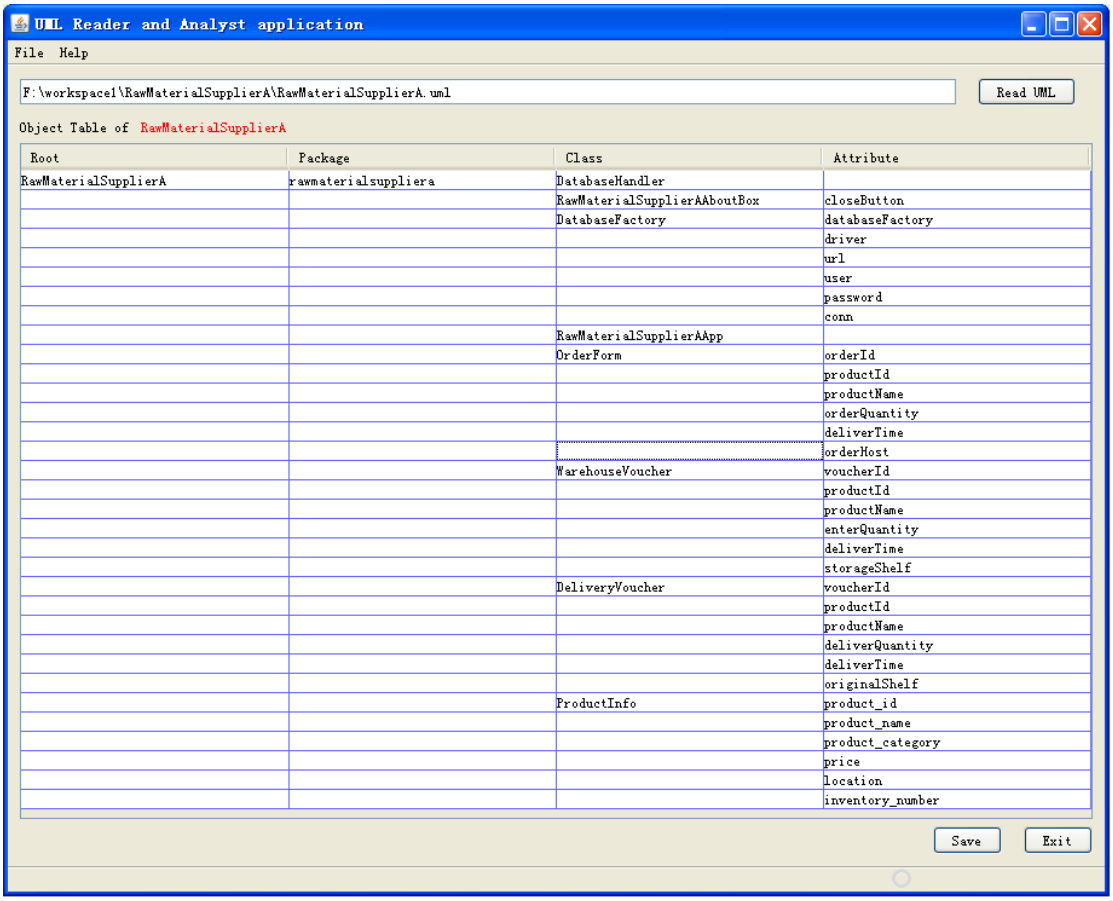

Figure 5-2. UML Reader and Analyst Application

After loading the .uml file, this application will provide a well arranged view of class relevant information as the table shown in figure 5-2. Afterwards, user can delete the useless UML class information by selecting the row or the cell where it locates in. After removing all the unnecessary information for HLA FOM generation, user can click button “save” to confirm that the rest information in the table is the expected information. And then the rest information will be saved into an xml file as shown in figure 5-3. This xml file is the input of the module “model alignment”, so it only retains the classes with their attributes. It can also be considered as the first version of the HLA SOM of the correlative enterprise federate.

```
<?xml version="1.0" encoding="GB2312" standalone="no" ?>
- <Root name="RawMaterialSupplierA">
  - <Package name="rawmaterialsuppliera">
      <Class name="DatabaseHandler" />
    - <Class name="OrderForm">
        <property>orderId</property>
        <property>productId</property>
        <property>productName</property>
        <property>orderQuantity</property>
        <property>deliverTime</property>
        <property>orderHost</property>
      </Class>
    - <Class name="WarehouseVoucher">
        <property>voucherId</property>
        <property>productId</property>
        <property>productName</property>
        <property>enterQuantity</property>
        <property>deliverTime</property>
        <property>storageShelf</property>
      </Class>
    - <Class name="DeliveryVoucher">
        <property>voucherId</property>
        <property>productId</property>
        <property>productName</property>
        <property>deliverQuantity</property>
        <property>deliverTime</property>
        <property>originalShelf</property>
      </Class>
    - <Class name="ProductInfo">
        <property>product_id</property>
        <property>product_name</property>
        <property>product_category</property>
        <property>price</property>
        <property>location</property>
        <property>inventory_number</property>
      </Class>
  </Package>
</Root>
```

Figure 5-3. Simplified UML class information

Model Evolution & Model Alignment

After reserved and simplified UML classes of each enterprise are ready, it is time to perform the model evolution. As mentioned in section 4.3.2.2, the participants have decided the model evolution groups based on their business relationship. Thus, in this case, different suppliers and car manufacturer have been categorized into different groups. This section will show an example of model alignment with one group that consists of one raw material supplier, one automobile parts supplier, and a car manufacturer.

The figure 5-4 shows the user interface of the model alignment module. The operation panel

of the first phase of the model alignment includes four parts. (1) On the top left corner, there is tree list (within the red box) for listing the imported projects SOMs. (2) Below this tree list, there is another tree list (within the blue box) for listing the URLs of the imported XML files. On the right side of the tree list area, it is the model analysis area. (3) On the top of this area, there is a table (within the green box) for performing model similarity transmission. (4) Under this table, there is the class diagram illustration area (within the purple box).

Firstly, the user has to load the xml files of the simplified UML class information of the raw material supplier, automobile parts supplier, and car manufacturer. And then, the tree list within the red box will list out the class information of imported projects, the tree list within the blue box will list out the URLs of the imported XML files. Meanwhile, the first row and the first column of the table within green box will be initiated with the classes' names of imported projects. Afterwards, if the user selects one cell of the table, then the class diagrams of the correlative row and column will show out in the class diagram illustration area. By comparing the class diagrams, user can define the similarity for the selected classes. In case the classes are hard to read and find on the table, the user can also select the classes they want to compare on the tree list within the red box, and then the corresponding table cell will be selected at the same time, and the class diagrams will shows up as well.

The cells of the model similarity transmission table are bound to a trigger event based on the similarity transitive algorithm introduced in section 3.3.2.2. Thus, while the user is defining the class similarity for each cell manually, the cell scans the whole cells on the corresponding row and column in order to detect the possibility of similarity transmission. If a possibility is detected, then the correlative cell will be highlighted and user can turn it later to define the similarity.

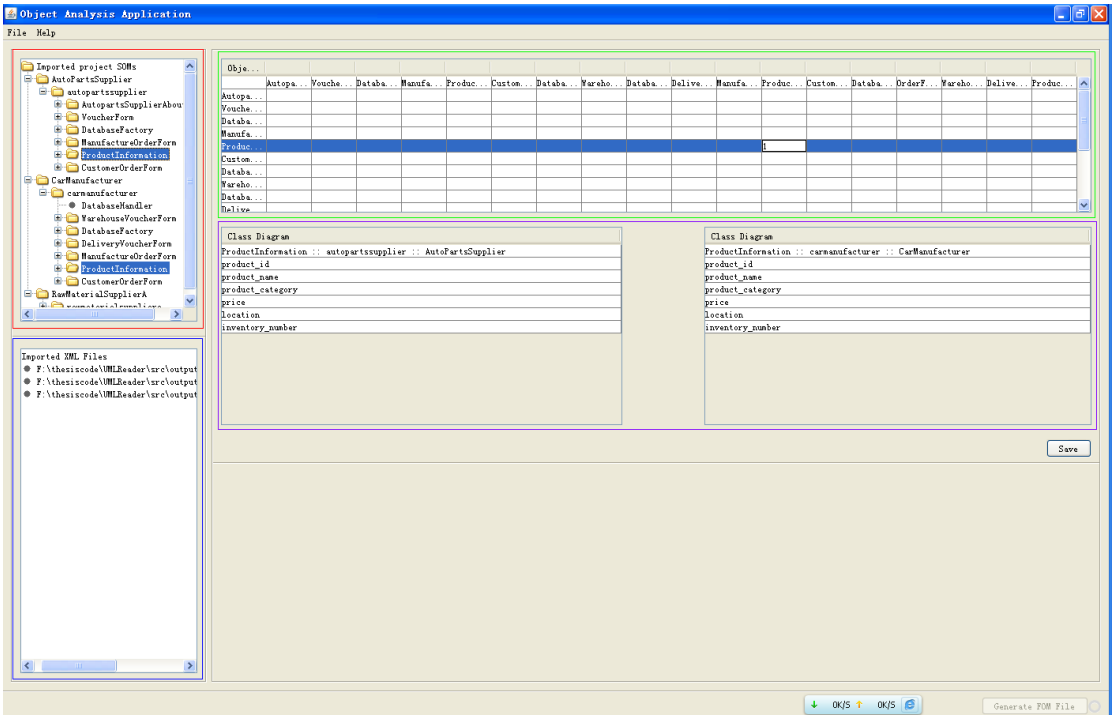

Figure 5-4. Model Alignment User Interface

When the user confirms that there are not proper classes to define the similarity, user can click the save button, which means that the first phase of the model alignment has been finished. And then, under the class diagram illustration area, a new operation panel (as shown in the figure 5-5, within the blue box) shows up for aligning the model structure.

After the first phase of the model alignment, the user has found several groups of similar classes. Each group of similar classes can be reformed into a new class for the next model generation or HLA FOM generation. So that, the user has to give a unified name to each group of similar classes and their attributes. Therefore, the new operation panel within the blue box provides the rename functionality by listing out the groups of classes and their attributes. The user can delete or rename the attributes, and input the new name for the classes in the text field at the bottom.

When, the model evolution has been finished, the user can click the button “generate FOM file”. And then, the HLA FOM file will be initiated according to the correlative RTI format. For example, this simulation chooses the portico RTI, then, the HLA FOM file will be organized in the format shown in the figure 5-6. Objects are embedded inside the pair label “Objects”. Each class has a “class” label with “class name” and the attributes are embedded inside it. The FOM file shown in figure 5-6 is incomplete, missing the interaction class part

which will be generated with the help from next section. An example of FOM generation code is shown in the Annex 2.

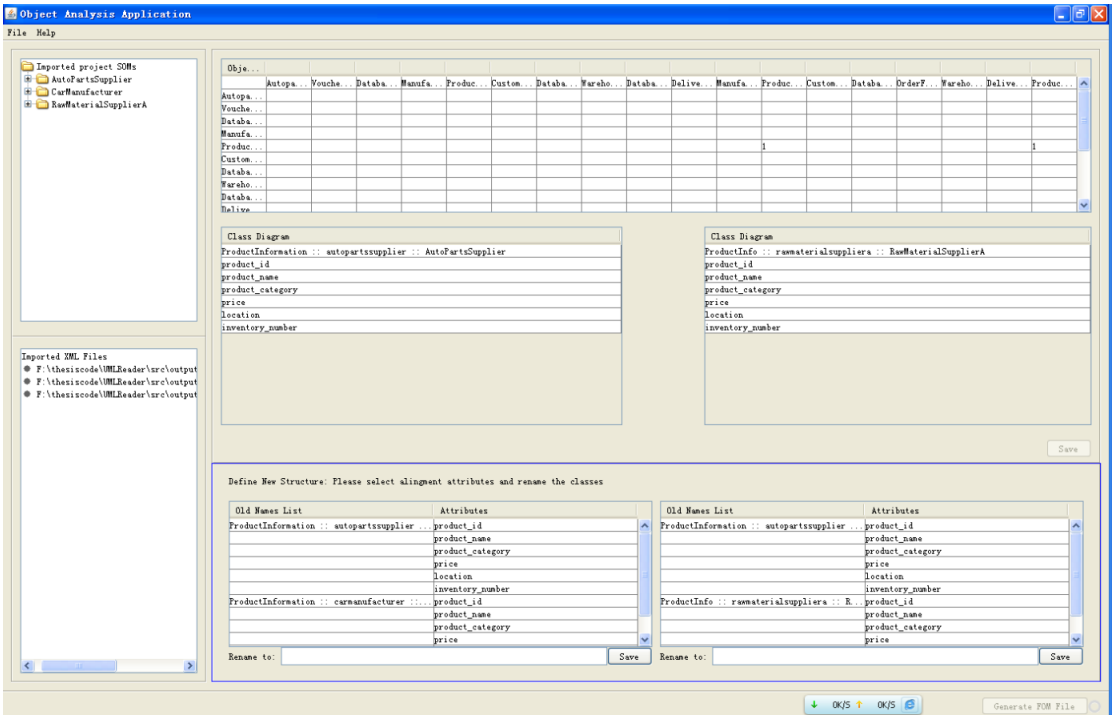

Figure 5-5. Align model structure

```
(FED
        (Federation Portico-Test)
                (FEDversion v1.3)
                (spaces
                        (space TestSpace
                                (dimension TestDimension)
                        )
                (space OtherSpace
                        (dimension OtherDimension)
                        )
                )
        (objects
                (class ObjectRoot
                        (attribute privilegeToDelete reliable timestamp)
                        (class RTIprivate)
                        (class VoucherForm

                                (attribute storageShelf reliable timestamp TestSpace)

                                (attribute voucherId reliable timestamp TestSpace)

                                (attribute voucherType reliable timestamp TestSpace)

                                (attribute productId reliable timestamp TestSpace)

                                (attribute productName reliable timestamp TestSpace)

                                (attribute deliverQuantity reliable timestamp TestSpace)

                                (attribute deliverTime reliable timestamp TestSpace)

                                (attribute originalShelf reliable timestamp TestSpace)

                        )

                        (class ManufactureOrderForm

                                (attribute deliverTime reliable timestamp TestSpace)

                                (attribute orderId reliable timestamp TestSpace)

                                (attribute productId reliable timestamp TestSpace)

                                (attribute productName reliable timestamp TestSpace)

                                (attribute orderQuantity reliable timestamp TestSpace)

                                (attribute deliverTime reliable timestamp TestSpace)

                        )

                        (class ProductInfo

                                (attribute inventory_number reliable timestamp TestSpace)

                                (attribute product_id reliable timestamp TestSpace)

                                (attribute product_name reliable timestamp TestSpace)

                                (attribute product_category reliable timestamp TestSpace)

                                (attribute price reliable timestamp TestSpace)

                                (attribute location reliable timestamp TestSpace)

                                (attribute inventory_number reliable timestamp TestSpace)

                        )

                        (class CustomerOrderForm

                                (attribute orderHost reliable timestamp TestSpace)

                                (attribute orderId reliable timestamp TestSpace)

                                (attribute productId reliable timestamp TestSpace)

                                (attribute productName reliable timestamp TestSpace)

                                (attribute orderQuantity reliable timestamp TestSpace)

                                (attribute deliverTime reliable timestamp TestSpace)

                                (attribute orderHost reliable timestamp TestSpace)

                        )

                )

        )

)
```

Figure 5-6. portico RTI based HLA FOM file

5.2.3. Run Time and Build Time II

As mentioned in section 4.5, the build time II is performed during the run time. The Build time II takes in charge of building the federate interface for web user to join the collaboration project running in the HLA federation. Thus, in order to ensure the fluency of demonstration, the applications of these two modules have to be presented together in one section.

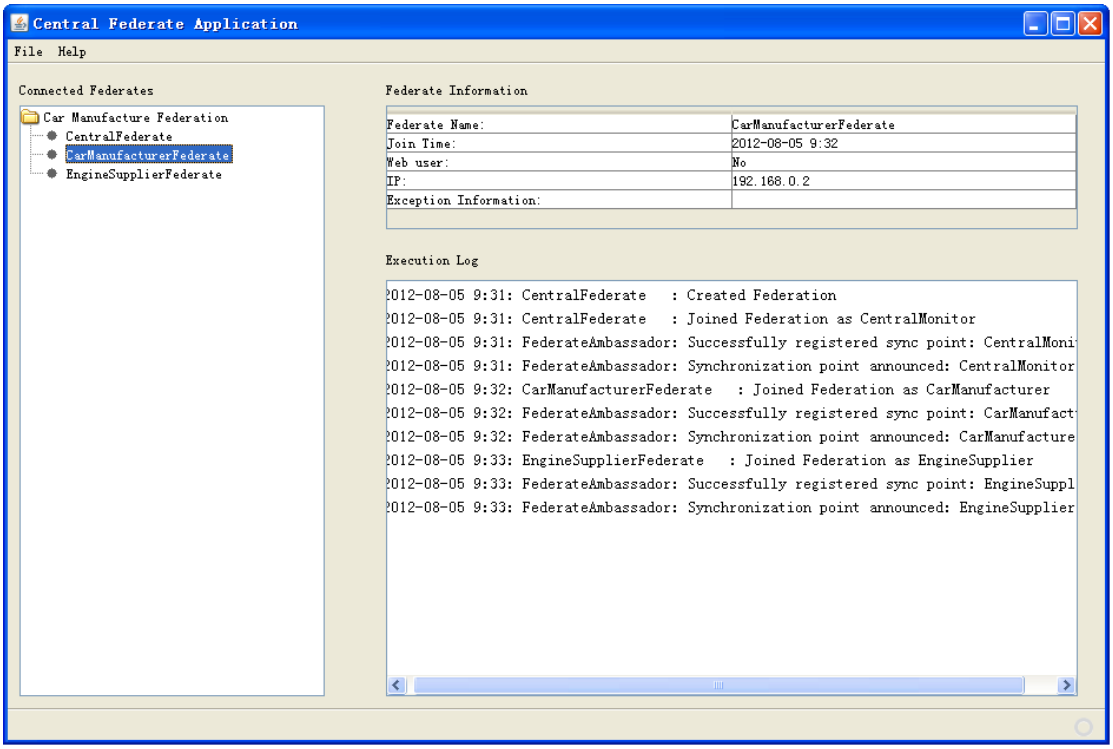

Figure 5-7. Central federate for Portico RTI

The run time is the HLA federation execution time. As the Portico RTI is an open source RTI without central monitor of the HLA federation execution, ***the central federate*** is designed to play this role. The figure 5-7 illustrates the user interface of the central federate. On the left side, there is a tree list for presenting the running federates inside the federation. Once the federate of one participant joins the federation, its name will be appended on this tree list. So that, we can know who are involved in the collaboration in time. On the top right corner, there is a table for illustrating the federate information. When user selects one federate on the tree list, all the information of this federate will shows up in this table. The information includes federate name, join time, web user, IP and exception information. The web user item is used to distinguish whether the federate is a traditional federate or is an evolved federate. If it is not a web user, the IP is the local network IP. Otherwise, the IP is the wide area network IP. The exception information item can help user to locate the exception, which can reduce the

maintenance time and complexity. Below the federate information table, it is the text area of the HLA execution log. The log recodes the time of federation creation, the join time and leave time of federates, the role names of federates, and the exception information of the federation execution.

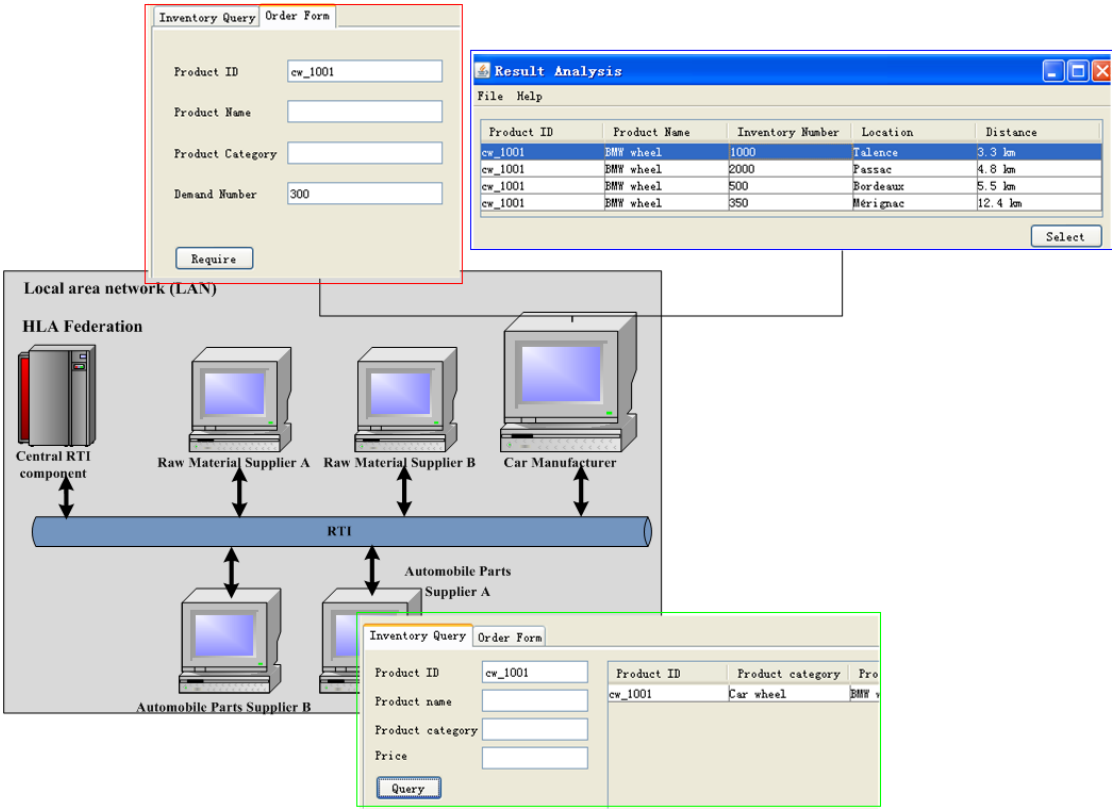

Figure 5-8. Federate user interface

The federate of each participant is the protagonist of the HLA federation execution. Build Time I has prepared the Integration code and Enterprise Business Behavior Interface for each federate. The Integration code takes in charge of the communication with RTI. The Enterprise Business Behavior Interface simulates the enterprise business process and it is the basis of the federate user interface development.

The figure 5-8 shows an example of the user interfaces. The user interface of car manufacturer federate within the red box on the top left is for automobile parts purchasing. The operator of the car manufacturer federate can input the information of needed goods, such as product name, product category, and demand. After the confirmation of this request, this request will be sent to the RTI by calling RTI specific code, such as, the sendInteraction (String interactionName, ArrayList parameterList) function (Annex 3 shows an example of

RTI specific code), and then the RTI will dispatch it to the federates of automobile parts suppliers. Because they have already subscribed to the purchasing order message published by car manufacturer. Afterwards, the automobile parts suppliers will query their database based the request information in the order of car manufacturer (as the user interface within the green box on the bottom of figure 5-8). If the automobile parts suppliers have enough inventories of required goods, they will answer this requirement. Then, the federate of car manufacturer will analyze all the feedbacks, decode the messages, and then generate an analysis report. The analysis report is shown in the user interface within the blue box on the top right of figure 5-8. The report includes the names of the suppliers, their location, the distance from car manufacturer to them, the inventory number, the price and etc. The application also provides a clearer view of the summary of total cost based on the data provided by this report (as shown in figure 5-9). The application can calculate the sum of the goods' price and the freight charge, so that the user can visually compare the suppliers. Finally, the car manufacturer will select the automobile parts suppliers that he satisfies with based on this report. For example, the figure 5-9 shows that the cost of purchasing from supplier in Bordeaux is most reasonable.

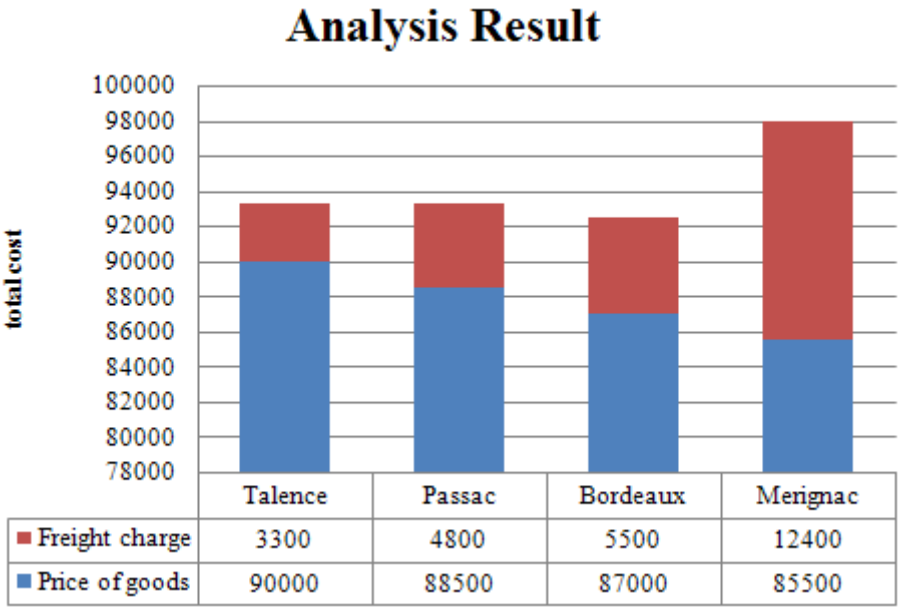

Figure 5-9. Analysis Result

The WebservicesFederate makes the traditional HLA federation agiler and more flexible. It bridges the potential participants and clients from the web with the members of the collaboration project within the HLA federation.

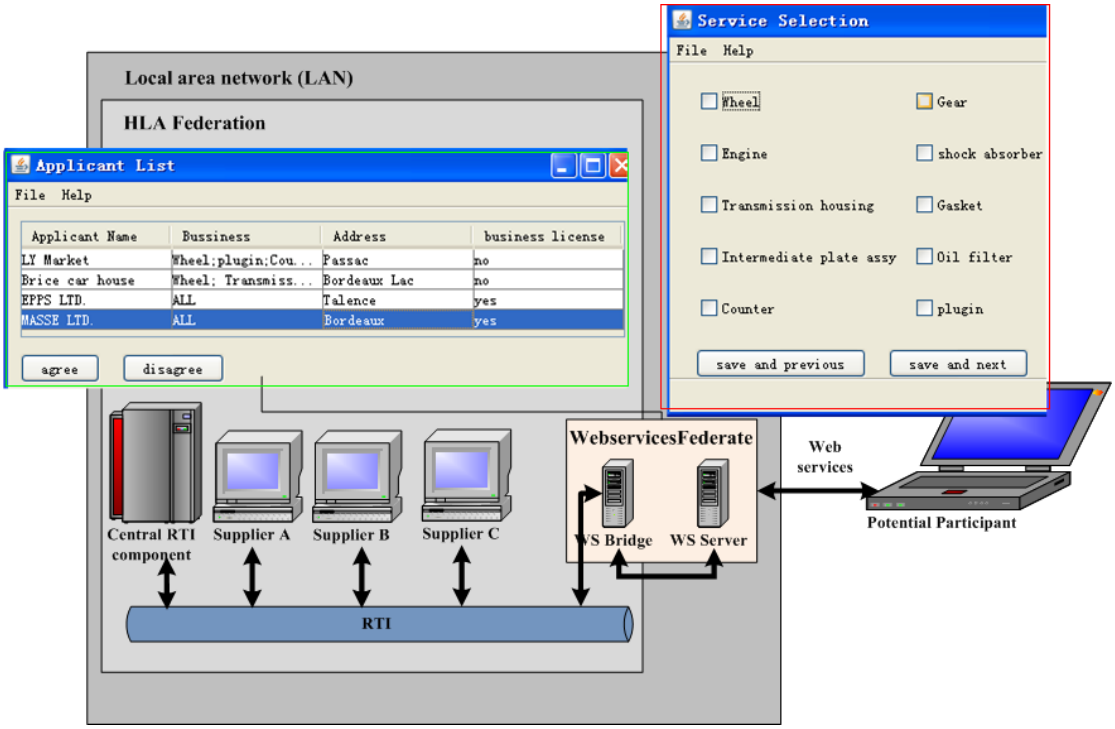

Figure 5-10. Alignment establishment between potential participant and existing members

The illustration of the user interface of alignment establishment between potential participant and existing members is shown in figure 5-10. For example, if one supplier of automobile parts wants to be the member of the collaboration project, he can find the Web Services published by WebservicesFederate (as the user interface within the red box on the right side of the figure 5-10 shown). Afterwards, he can choose the collaboration items and fulfill his own information, and then submit the application form to the existing federation. The administrator of the HLA federation will verify the applicant list (as the user interface within the green box on the left side of the figure 5-10 shown), and decide which application can be accepted. After passing the evaluation, the applicant can download the Web Services package to generate his own federate interface by fully following the procedure mentioned in section 4.5.

In addition, the new participant will perform the ontology alignment with the existing participants in the HLA federation to create his local ontology glossary, which will be used for automatically analyze the information from existing HLA federation. The table 5-1 shows an example of the result of the ontology alignment that uses the multi-strategies ontology alignment approach (Song et al., 2012) as shown in section 4-5. This approach will assign similarity value to each pair of ontology elements. Only the pairs with the similarity value bigger than the threshold value will be kept and aligned.

Table 5-1. Ontology alignment between new participant and existing participants

Elements of existing participants	Elements of new participants	Relationship	Similarity value	Id
ProductID	CargoID	Similar/equal	87.496%	1
ProductName	CargoName	Similar/equal	87.496%	2
ProductCategory	CargoClassification	Similar/equal	74.527%	3
Price	Price	Equal	100%	4
InventoryNumber	InventoryLevel	Similar/equal	90.831%	5

The figure 5-11 shows an example of car purchasing and car manufacturing which happen between a customer from web and car manufacturer federate with HLA federation.

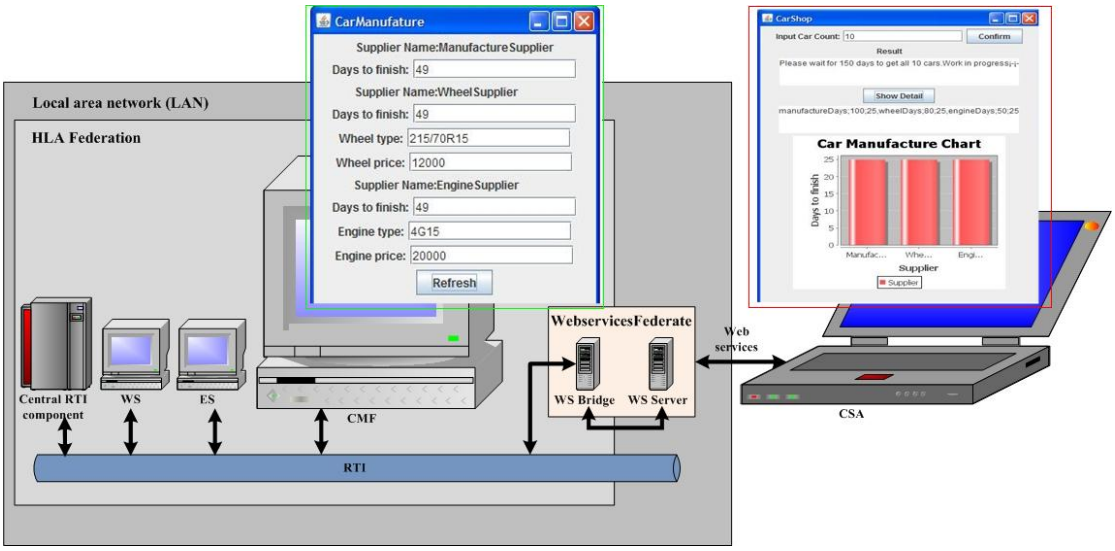

Figure 5-11. Example of car purchasing and car manufacturing

Firstly, the customer sends an order to the car manufacturer federate (CMF). And then the car manufacturer federate calculates the amount of raw materials based on the bill of material. After that it dispatches orders to the different suppliers, such as wheel supplier (WS) and engine supplier (ES), to get the parts. When suppliers finish the production, they deliver the products to the CMF who assembles them and then deliver to the customer. In this simulation, the status of the order is the concerned issue. The customer cares about when he can receive his cars and the status of this manufacturing process. As a result, the customer does not have the complex interaction with the existing federate within the HLA federation. Thus, he does not need to follow the complex procedure as the web applicants mentioned earlier. The only thing needs to know is what kinds of cars the car manufacturer can provide. The only thing

needs to follow is the HLAservice Interface shown in the figure 5-12. By implementing this interface, the customer can send the purchasing order, can know when he can receive the cars, and can trace the manufacture process.

```
public interface HlaService {

    pubilc void sendOrder(String ip, String orderMessage) throws IOException;
    public int getDaysToAllFinished(String ip,int carCount)throws IOException;
    public String getAllCurrentState(String ip) throws IOException;
}
```

Figure 5-12. HLA service for client

As the user interface within the red box on the right side of the figure 5-11 shown, the customer can input order quantity. After the confirmation of this order, the client side shows the total number of manufacturing days. During the car manufacturing, the customer can request the detail of the manufacturing process, and the federation immediately sends back the result, which has been generated as a bar chart to vividly illustrate the status of the manufacturing process.

In the CMF user interface (as the user interface within the green box on the left side of the figure 5-11 shown), the remaining days of car manufacture is presented. In order to assist the car assembling process, the information of the suppliers is presented. By clicking the 'refresh' button, CMF can receive the latest information from each supplier. The Gantt chart shown in figure 5-13 is a real time report of the car manufacturing schedule. As shown, the car assembling process is waiting for the wheel and engine, any delay caused by suppliers will postpone the schedule of car assembling process. Thus, the car manufacturer has to monitor the suppliers' manufacturing process, in order to be able to cope with the incidents by adjusting the schedule or proposing a new solution. This Gantt chart allows the car manufacturer to monitor the suppliers in real time. For example, the figure 5-13 shows that the wheel supplier has completed 20% process, and engine supplier has completed 40%, both of them are progressing under the schedule.

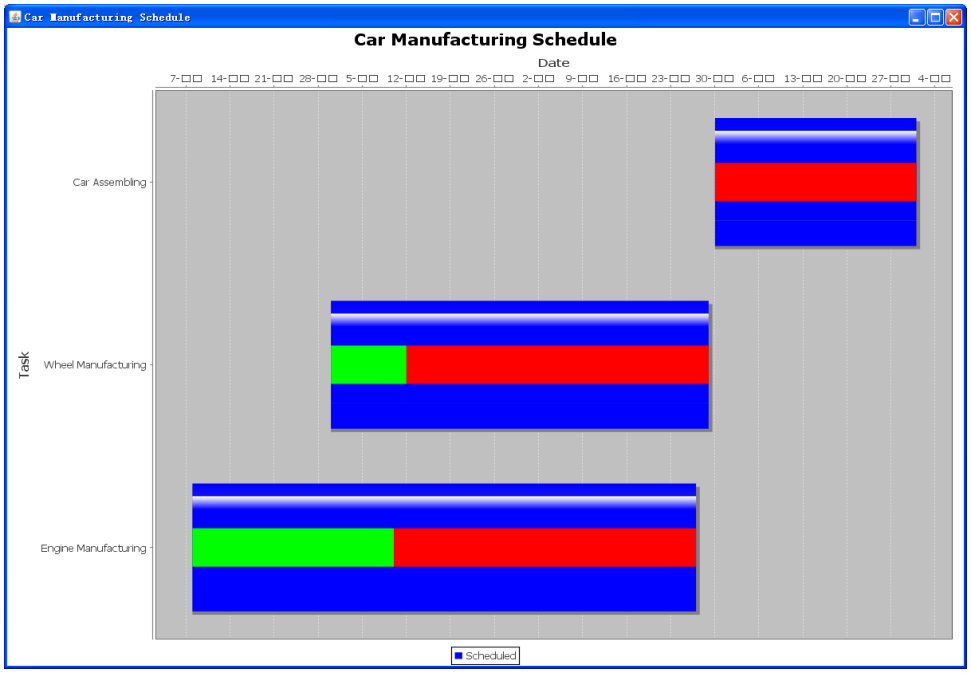

Figure 5-13. Car manufacturing schedule

Since the supplier may be a participant coming from the web, its information may lose because of network fault. Therefore, the car manufacturer cannot get the real time information to make the decision. As the failure tolerance solution introduced in section 3.4.2.4, the WebservicesFederate will play as a standby federate to give a network failure report to the car manufacturer. At the same time, the simulation will keep running, but the WebservicesFederate will request the supplier's federate for the information of the state when it disconnected.

5.3. Conclusion

The simulation of this case runs correctly with laboratory data. It has proved the feasibility of the federated approach proposed in this thesis. The reverse engineering tool can obtain system object information and system state information. So that it can help to generate the major part of federate code. This tool also provides the bridge component that passes some gaps between the HLA 1516 Evolved standard and the API provided by poRTIco.

This case study proves that the harmonized and reversible HLA based and framework and methodology can implement the federated approach of enterprise interoperability. The harmonized HLA and MDA framework guides the development. The model reverse method discovers the information from participants' legacy systems, and uses it to quickly establish HLA federation for simulating enterprise interoperability. The web-enable federate design

method provides a web-enable RTI solution to allow potential participants to easily join the existing collaboration through web as “plug-and-play”. The short-lived ontology method initiates the local ontology glossary of web participant for information analysis.

Nevertheless, the simulation runs based on the academic data, and it still needs to be validated in the industrial project. In addition, the bridge component code for implementing WebserviceFederate is performed at the application layer, which means that the Web-RTI functionality is implemented without changing any mechanism or source code of poRTIco. As a result, the link, between the WebServiceFederate and the rest of the HLA federation, is not providing all HLA functionalities as web services on the web. The role of the external component is mostly constrained to receiving data and basic actions of data sending, e.g. it cannot impose time management modifications. In addition the security issue could also been involved. Now, the authors use encapsulation and encoding method to ensure the security of the data package.

General Conclusion

This thesis has contributed to develop a Reversible Model driven and HLA based framework and methodology for implementing federated approach under the Enterprise Interoperability Framework. Firstly, a Harmonized and Reversible HLA based framework has been elaborated and its associated methodology defined. This methodology has proposed a novel way to support the development of federated approach of enterprise interoperability by reusing some existing methods, architectures, and technologies, such as MDA (Model Driven Architecture), Model Reverse Engineering, HLA (High Level Architecture), Web Services, and Ontology. More precisely, this methodology (1) utilizes MDA to formalize the system architecture and relationship among systems, (2) applies Model reverse engineering to reuse and align different systems/component to initiate enterprise IS interoperability environment, (3) uses the HLA and Web Services functionalities as technical support, and (4) uses Ontology for the information analysis. After the definition of the methodology, a Model driven and HLA based Reverse Engineering architecture has been elaborated based on which a software tool has been developed. The use of this software tool has been illustrated through an illustrative case study.

Chapter 1 identified and defined the scope and objectives of this doctoral research. Firstly, it presented the economic and industrial context, and the research background of Enterprise Interoperability. And then, it presented the definitions and conceptual explanations of Enterprise Interoperability. Afterwards, it analysed the current situation of Enterprise Interoperability and elaborated the research challenges, priorities, and tendency of the current enterprise interoperability research. Finally, it pointed out the objective and expected results of this doctoral research according to the research challenges.

Chapter 2 made a survey of the existing methods and architectures that are relevant to federated enterprise interoperability. Firstly, it reviewed the existing models for system interoperability to identify the relevant concepts, methods, and principles that can be useful suggestions for solving Enterprise Interoperability from the views of conceptual, organizational, and technological barriers. Afterwards, it describes model driven technologies, software and application distribution frameworks, and ontology into detail. It compared the methodologies or technologies in the same domain to point out their advantages for conducting to federated enterprise interoperability, and also their shortfalls that need to be complemented to satisfy with the requirement of federated enterprise interoperability.

Chapter 3 presented the main contribution of this research work. It defined a Harmonized and Reversible HLA based framework and methodology.

- The harmonized HLA&MDA engineering framework has proposed a five steps development lifecycle that adopts the strong points from both HLA FEDEP and MDA. In addition, a harmonized single federate structure has been defined as the result of this framework.
- Model reverse method has proposed a way of using UML models discovered from the existing systems to generate HLA FOM that represents system static information and HLA federate code block that represents system behaviour.
- Web-enabled HLA federate design method complies with the rules defined in HLA evolved IEEE 1516^{TM}-2010. This method can strengthen the compatibility and self-learning ability of the traditional HLA federation environment, and also strengthen the time management, environment security control, and system state management of web services.
- Short-lived ontology method has proposed a particular non persistent ontology with a very short lifetime, which is used to generate the information analysis part of each HLA federate.

Chapter 4 explained the implementation of a Model driven and HLA based Reverse Engineering tool that is based on the Harmonized and Reversible HLA based methodology. It described the implementation in three parts, build time I, run time, and build time II.

- Build time I is responsible for establishing Enterprise Interoperability environment by modelling, including model reversal, model adjustment, and target model & code generation.
- Run time is for the execution of Enterprise Interoperability, including message dispatch and management. It concerns transient information exchange and analysis.
- Build time II is for establishing Enterprise Interoperability with potential participants from web, by generating web-enable federate and connecting it with the executed HLA federation.

Chapter 5 showed a case study of the Model driven and HLA based Reverse Engineering tool with laboratory data. This case describes a scenario of car purchasing and car manufacturing.

This scenario includes customer ordering, car manufacturing, material purchasing/delivering, and product delivering. This case shows the rapid interoperability establishment of the original participants, including car manufacturing semi-manufactured goods/ automobile parts suppliers, and raw material suppliers. It also shows how participants use the interoperability simulation results to make the business decision. In addition, this case shows how potential participants, such as customer and suppliers, discover the interoperability activity, and join it.

All the research surveys and contributions are for the purpose of achieving the research objectives.

(1) Transient information exchange and analysis without common format at conceptual barrier, which are supported by web-enabled HLA federate design method and short-lived ontology.
(2) Rapid and dynamic interoperability establishment, dynamic negotiation, and agile environment compatibility at technological barrier, which are supported by harmonized HLA&MDA engineering framework, model reverse method, and web-enabled HLA federate design method.
(3) Easy connection and the collaboration environment management at organizational barrier, which are supported model reverse method, and web-enabled HLA federate design method.

The five steps development lifecycle of the Harmonized HLA & MDA engineering framework combines HLA FEDEP with MDA. MDA is responsible for standardizing the modelling process, so that the models are general and common, which can enhance the model reusability. HLA FEDEP is a standardized process for developing interoperable HLA based federations, which provides specific constraints for the model transformation towards a clear target. To sum up, this framework is a model driven enterprise interoperability framework, which proposes a standardized process for establishing federated enterprise interoperability. It provides a development environment, where model reversal is under control, model transformation (no matter forwards and backwards) is fluently carried out, and interoperability modelling is performed throughout the entire development process.

Moreover, the output of this framework - the harmonized single federate structure dissociates the business behaviour code (called “adapter” in this thesis) from RTI specific code (called “plug-in” in this thesis). This dissociation reduces the model coupling, which can enhance the

system reusability and maintainability. In addition, this dissociation conduces to the implementation of “plug and play” mechanism that can help to achieve the rapid, and dynamic interoperability establishment, and agile environment compatibility.

Model reverse method is designed to discover the enterprises’ knowledge from the legacy information systems. As mentioned earlier, the model reverse method is implemented under the Harmonized HLA & MDA engineering framework. This method uses MoDisco tool to discover UML models that are used for model evolution and model alignment. The model evolution and model alignment is the process of model adjustment and accommodation, which aims at achieving the interoperability modelling in “on-the-fly” negotiation. The output of this negotiation is a set of interoperable models that can be used to generate HLA FOM and initiate local ontology glossary of short-lived ontology. Moreover, the UML models discovered from the existing systems are also used to assist the generation of system behaviour. This method uses program tracer to collection system execution information. And then, the system execution information is represented as directed graph in which UML class is the node, function call between classes is the arc. Afterwards, the directed graph is reduced and transferred into system state diagrams that can be transformed to system simulation code. This model reverse method is responsible for implementing the harmonized single federate, in order to achieve “plug and play”. It obtains semi-automatically the knowledge from the existing systems, and generates automatically the “adapter” of the harmonized single federate by model transformation. So that it avoids completely redevelop the existing systems, and allows them to establish interoperability rapidly.

The method of web-enable HLA federate design is based on the open source RTI, portico. This method fulfils HLA evolved IEEE 1516^{TM}-2010 standard. A novel federate called WebserviceFederate is designed to bridge the gaps between HLA Evolved approach requirements and the HLA 1.3 API provided by portico. This method uses the intermediate results of model reverse process, such as the set of interoperable models and behaviour models, to generate the web services. Thus, the potential participants can discover the existing interoperability activity, and use the web services to rapidly generate their own “adapter” to join this interoperability activity. This method is responsible for implementing easy connection for potential participants, and authority management and interoperation environment management for HLA federation (interoperation environment).

The short-lived ontology method is responsible for supporting the "on-the-fly" negotiation semantically. It allows the interoperability accommodation and adjustment do not need to impose the existing models, languages and methods of work as the common format. The mechanism of this method includes the method of initiating and upgrading the local ontology glossary, and the technical schema of the short-lived ontology interpretation request/response. In order to link this method up with the model reverse method to develop an intelligent agent for achieving federated enterprise interoperability, the technical schema of the short-lived ontology interpretation request/response is designed as the state diagram. Finally, this short-lived ontology method can be implemented as the information analysis code of the "adapter".

To sum up, this thesis proposed a Reversible Model driven and HLA based framework and methodology. The methodology consists of a set of existing methods, architectures, and technologies, to support federated approach of Enterprise Interoperability. This methodology has a model driven development lifecycle to standardize the process of interoperability establishment, and a model reverse engineering process to semi-automatically collect relevant information and data of the existing systems for quickly reengineering enterprise systems. These model driven development lifecycle and model reverse engineering process reduce the complexity of EI establishment and implement the "plug-and-play" mechanism in technical layer. This methodology also has a web-enable federate design method that allows enterprises to adapt and accommodate dynamically to the potential interoperability partners. The HLA evolved platform provides an interoperability environment where enterprises can interoperate simultaneously with multiple heterogeneous partners.

The harmonized and reversible HLA based framework and methodology have been systematically described. A Model driven and HLA based Reverse Engineering Tool has been developed to implement this methodology. However, the behaviour model reverse method and the short-lived ontology method have been proposed, but not been fully implemented, because of the priority of implementation and time limitation of my doctoral research. We have only implement the method of system traversal, reduction rule definitions of model processing, and the algorithm of state diagram generation. We have implemented the generation of web participant's local ontology glossary by using a novel ontology alignment approach. This approach is proposed by another PhD candidate of our laboratory. It is an ontology alignment approach with multiple strategies and aggregated based on Method

Analytic Hierarchy Process (AHP). This approach supports the dynamic and automatic aggregation of different matching results (Song et al., 2012).

As mentioned above, we have proposed a framework and methodology for implementing federated approach of enterprise interoperability. We have validated most of the methods under the framework and methodology. We have published our contribution in many papers, such as the five step development lifecycle in I-ESA 2010 (Tu et al., 2010a), web-enable federate design method in WinterSim 2012 (Tu et al., 2011b), and the framework and methodology in research handbook (Tu et al., 2012a), IJCIM journal (Tu et al., 2012b), INSIGHT journal (Tu et al., 2011c). However, there are remaining works to be done in the future, which are considered as following:

- Behaviour model reverse method: this thesis used the behaviour model to generate system simulation code. Thus, the behaviour model only needs to represent the system logic, so that the federate knows the access of the existing systems. However, if we enrich this behaviour model with some additional elements, such as time, and rules of equivalent class definitions, then this behaviour model will not only represent the system logic, but also the business logic, so that it can be transformed into BPMN model, DEVS model, and etc. And then, these models can be used to achieve federated enterprise interoperability in process concern and even business concern.
- Short-lived ontology method: this thesis proposed a way of using short-lived ontology for data interoperability. Actually, the short-lived ontology can also be used in the model adjustment and accommodation. The ontology description can be added on the UML models, so that, we can reduce the manual operations when we determine the model similarity. However, it will increase the complexity of the initiation of the local ontology glossary. The local ontology glossary has to be initiated along with the model evolution. In addition, short-lived ontology has a self-learning mechanism for information analysis. In fact, this self-learning mechanism can also be used in the interoperability environment upgrade. It means that the interoperability environment system can analyze the short-lived ontology self-learning experience, so that it can identify some new and interesting requirements of the participants. Based on these discoveries, the interoperability environment can self-upgrade, and then inform the participants and upgrade the web services at the same time.

References

(ABILITIES, 2008) ABILITIES project, (2008). *Application Bus for InteroperabiLITy In enlarged Europe SMEs*, Available from http://www.dbis.cs.uni-frankfurt.de/index.php [accessed 07 July 2011].

(Aho et al., 1972) Aho A., Garey M., Ullman J., (1972). The Transitive Reduction of a Directed Graph. *SIAM Journal on Computing.* 1(2), 131–137.

(Ambler, 2003) Ambler S.W., (2003). Agile model-driven development is good enough. *IEEE SOFTWARE* 20(5): 71-73.

(ATHENA, 2003) ATHENA: FP6-2002-IST-1, (2003). *Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications*, DEUTSCHLAND: SAP AG.

(ATHENA, 2005) ATHENA, (2005). *Deliverable DA4.1 Requirements for Interoperability Framework, product-based and process-based Interoperability Infrastructures, Interoperability Life-cycle Services*, Available from http://www.asd-ssg.org/html/ATHENA/Deliverables/Deliverables%20provided%20to%20EC%202nd%206%20months/050321_ATHENA_DA41_V10.pdf [accessed 07 July 2011].

(ATHENA, 2007) ATHENA, (2007). *Deliverable DA4.2 Specification of Interoperability Framework and Profiles, Guidelines and Best Practices*, Available from http://www.asd-ssg.org/html/ATHENA/Deliverables/Deliverables%20provided%20to%20EC%206th%206%20Months/070322_ATHENA_DA42_V10.pdf [accessed 07 July 2011].

(Bézivin, 2005) Bézivin J., (2005). On the Unification Power of Models. *Software and Systems modeling*, 4(2), 171-188.

(Bézivin et al., 2006) Bézivin J., Brunelière H., Barbero M., (2006). *The Model Discovery (MoDisco) Component: A Proposal for a New Eclipse/GMT Component, Version 1*. INRIA (ATLAS Group). Available from http://www.eclipse.org/MoDisco/about.php.

(Biggs et al., 1986) Biggs N., Lloyd E., Wilson R., (1986). *Graph Theory*. Oxford University Press. 1736-1936.

(Bourey et al., 2007) Bourey, J.P., Grangel Seguer, R., Doumeingts, G., and Berre, A.J., (2007). *Report on Model Driven Interoperability*, Deliverable DTG 2.3, INTEROP NoE, April, pp. 91. Available from http://www.interop-vlab.eu/ [accessed 15 June 2009]

(Bruzzone et al., 2007) Bruzzone A.G., Bocca E., Longo F., Massei M., (2007). Training and Recruitment in Logistics Node Design by using Web Based Simulation. *International Journal of Internet Manufacturing and Services*, I(1), 32-50.

(Bruzzone et al., 2009) Bruzzone A.G., Fadda P., Fancello G., Bocca E., D'Errico G., Massei M., (2009). Virtual world and biometrics as strongholds for the development of innovative port interoperable simulators for supporting both training and R&D. *Int. J. Simulation and Process Modelling*, 6(1), 89 - 102.

(Bruzzone et al., 2011) Bruzzone A.G. Massei M., Tremori A., (2011). Adding Smart to the Mix. *Modeling Simulation & Training: The International Defense Training Journal*, 3, 25-27.

(Buss et al., 1998) Buss A., Jackson L., (1998). Distributed Simulation Modeling: A Comparison Of HLA, CORBA And RMI. *Proceedings of Winter Simulation Conference*. 819-825.

(C4ISR, 1998) C4ISR, Architecture Working Group (AWG), (1998). *Levels of Information Systems Interoperability (LISI)*. Washington, DC. Available from http://www.eng.auburn.edu/~hamilton/security/DODAF/LISI.pdf

(Chen et al., 1997) Chen D., Vallespir B., Doumeingts G., (1997), GRAI integrated methodology and its mapping onto generic enterprise reference architecture and methodology, *Computers in Industry*, 33(2-3).

(Chen et al., 2003) Chen, D., Doumeingts G., (2003). European initiatives to develop interoperability of enterprise applications—basic concepts, framework and roadmap. *Annual Reviews in Control*, 27(2), 153-162.

(Chen et al., 2004) Chen D., Knothe T., Zelm M., (2004). ATHENA Integrated Project and the Mapping to International Standard ISO 15704. *Proceedings of the International Conference on Enterprise Integration and Modeling Technology (ICEIMT'04)*, Canada.

(Chen et al., 2008) Chen D., Shorter D., (2008). Framework for Manufacturing Process Interoperability. *Standards for Interoperability - How, Workshop in Conjunction to I-ESA*, Berlin : Germany

(Charalabidis et al., 2008) Charalabidis Y., Gionis G., Moritz Hermann K. and Martinez C., (2008). *Enterprise Interoperability Research Roadmap, Draft Version 5.0.* Available from ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/enet/ei-roadmap-5-0-draft_en.pdf [accessed 20 December 2010]

(Chen, 2009) Chen D., (2009). Framework for enterprise interoperability. *Congrès International de Génie Industriel (CIGI2009)*, Bagnères de Bigorre: France.

(Clark et al., 1999) Clark, T., Jones, R., (1999). Organization al Interoperability Maturity Model for C2. *Proceedings of the 1999 Command and Control Research and Technology*

Symposium. United States Naval War College, Washington, DC: Command and Control Research Program (CCRP).

(Cooper, 2000) Cooper J.W., (2000), *Java™ Design Patterns: A Tutorial*. addsion wesley longman, Inc.

(Cormen et al., 2003) Cormen T.H., Leiserson C.E., Rivest R.L., Stein C., (2003). *Introductions to Algorithms*. MIT Press.

(Colan, 2004) Colan M., (2004). *Service-Oriented Architecture expands the vision of web services, Part 1: Characteristics of Service-Oriented Architecture*. Available from http://www.ibm.com/developerworks/webservices/library/ws-soaintro/index.html.

(Collins, 2005) Collins W.J., (2005). *Data Structures and the Java Collections Framework. Second Edition*. McGraw Hill.

(COIN, 2011) COIN Project, (2011). *Enterprise Collaboration & Interoperability*, Available from http://www.coin-ip.eu/ [accessed 07 July 2011].

(De Nicola et al., 2009) De Nicola A., Missikoff M., Navigli R., (2009). A Software Engineering Approach to Ontology Building. *Information Systems*. 34(2), 258-275.

(Doumeingts et al., 2001) Doumeingts G., Ducq Y., (2001), Enterprise Modelling techniques to improve efficiency of enterprises, *International Journal of Production Planning and Control - Taylor & Francis*, 12(2), 146-163

(DoD DMSO, 2006) DoD DMSO, (2006). *Introdution to VV&A*. Available from http://vva.msco.mil/Key/key.htm.

(EIF, 2004a) EIF, (2004a). *European Interoperability Framework, White Paper*. Brussels, Available from http://www.comptia.org.

(EIF, 2004b) EIF, (2004b). *European Interoperability Framework for PAN-European EGovernment Services, IDA Working Document, Version 4.2*.

(Elvesæter et al., 2007) Elvesæter B., Hahn A., Berre A., Neple T., (2007).Towards an Interoperability Framework for Model-Driven Development of Software Systems. *Interoperability of Enterprise Software and Applications*, 409-420.

(EN/ISO, 2003) EN/ISO I9439, (2003). *Enterprise Integration—Framework for Enterprise Modelling*, Technical Committee CEN/TC 310.

(ENSEMBLE, 2011) ENSEMBLE, (2011). *Deliverable 2.1 EISB State of Play Report version 1.00*. Available from http://www.fines-cluster.eu/fines/wp/d21/ [accessed 07 July 2011].

(EC, 2008) European Commission (EC), (2008). *Unleashing the Potential of the European Knowledge Economy: Value Proposition for Enterprise Interoperability (version 4.0)*, Informal Study Group on Value Proposition for EI.

(Favre et al., 2008) Favre L., Martinez L., Pereira C., (2008). MDA-Based Reverse Engineering of Object Oriented Code. *SERA'08*, 153-160.

(Fujimoto, 2000) Fujimoto R. M., (2000). *Parallel and Distributed Simulation Systems*, Wiley Interscience.

(Gehani, 1983) Gehani N., (1983). *Ada: An Advanced Introduction*. Prentice-Hall, Engelwood Cliffs, New Jersey.

(Genesereth, 1992) Genesereth M.R., Fikes R.E., (1992). *Knowledge Interchange Format, Version 3.0 Reference Manual*. Logic- 92-1. Computer Science Department, Stanford University.

(Gisolfi, 2001) Gisolfi D., (2001). *Web services architect: part 1. An introduction to dynamic e-business*, IBM developer Works Web Services articles. Available from http://www.ibm.com/developerworks/webservices/ [Accessed April 20, 2012]

(Gomez-Perez et al., 2004) Gomez-Perez A., Corcho O., Fernandez-Lopez M., (2004). *Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. (Advanced Information and Knowledge Processing)*, First Edition, Springer. 54-121.

(Gorka et al., 2007) Gorka B., Larrucea X., Elvesæter B., Neple T., Beardsmore A., Friess M., (2007), *A Platform Independent Model for Service Oriented Architectures*, Enterprise Interoperability, Ed, London, S., pp. 23-32.

(Gonçalves et al., 2012) Gonçalves R., Agostinho C., Garção A., (2012). A reference model for sustainable interoperability in networked enterprises: towards the foundation of EI science base. *Computer Integrated Manufacturing*. 25(10), 855-873.

(Gruber, 1992) Gruber T.R., (1992). *Ontolingua: A Mechanism to Support Portable Ontologies*. KSL 91-66. Stanford University, Knowledge Systems Laboratory.

(Gruber, 1993) Gruber T., (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 199–220.

(Gruber, 1995) Gruber T., (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5-6), 907 - 928.

(Granowetter, 1999) Granowetter L., (1999). Solving the FOM-Independence Problem. *Proceedings of Simulation Technology and Training Conference.*

(Granowetter, 2004) Granowetter L., (2004). Design of the Dynamic-Link-Compatible C++ RTI API for IEEE 1516. *Proceedings of Fall Simulation Interoperability Workshop*. 04F-SIW-086.

(Gustavson et al., 2005) Gustavson P., Chase T., Root L., Crosson K., (2005). Moving Towards a Service-Oriented Architecture (SOA) for Distributed Component Simulation Environments. *Proceedings of Spring SIW*, San Diego, USA.

(H.Wache et al., 2001) H.Wache T. V. o., Visser U., Stuckenschmidt H., Schuster G., Neumann H., H ¨ubner S., (2001). Ontology-Based Integration of Information — A Survey of Existing Approaches. *IJCAI-01*. Seattle, USA.

(IDEAS, 2003) IDEAS, (2003). IDEAS project deliverables (WP1-WP7) (Public reports).

(IDABC, 2008) IDABC, (2008). *European Interoperability Framework draft version 2.0*. Available from http://ec.europa.eu/idabc/servlets/Docb0db.pdf?id=31597 [accessed 07 July 2011].

(IEEE, 1990) IEEE std 610, (1990). *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, Institute of Electrical and Electronic Engineers.

(IEEE, 1995) IEEE-1278.2-1995, (1995). *Standard for Distributed Interactive Simulation - Communication Services and Profiles.* New York: Institute of Electrical and Electronic Engineers

(IEEE, 2000) IEEE std 1516.2-2000, (2000). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification*, New York: Institute of Electrical and Electronic Engineers.

(IEEE, 2003) IEEE std 1516.3-2003, (2003). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federation Development and Execution Process (FEDEP)*, The Institute of Electrical and Electronic Engineer.

(IEEE, 2010) IEEE std 1516™-2010, (2010). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) -- Framework and Rules*, New York: Institute of Electrical and Electronic Engineers

(IEEE, 2011) IEEE std 1730™-2010, (2011). *IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP)*, The Institute of Electrical and Electronic Engineer.

(INTEROP, 2003) INTEROP, (2003). *Interoperability research for networked enterprises applications and software, network of excellence*, France: UNIVERSITE DE BORDEAUX I.

(ISO, 1999) ISO 14258, (1999). *Industrial Automation Systems – Concepts and Rules for Enterprise Models*, ISO TC184/SC5/WG1

(IST, 2001) IST-2001-37368, (2001). *Thematic network, IDEAS: Interoperability development for enterprise application and software—Roadmaps, description of work*, France: UNIVERSITE DE BORDEAUX I.

(ISO, 2008) ISO 19440:2007, (2008). *Enterprise integration. Constructs for enterprise modelling*, Organisation internationale de normalisation.

(ISO, 2011) ISO 11354-1, (2011). *Advanced automation technologies and their applications -- Requirements for establishing manufacturing enterprise process interoperability -- Part 1: Framework for enterprise interoperability*, ISO TC 184/SC5

(Jaccard, 1912) Jaccard P., (1912). The distribution of the flora in the alpine zone. *New Phytologist.* 11(2), 37-50.

(JFreeChart, 2008) JFreeChart, (2008). The JFreeChart Developer Guide. Available from http://www.jfree.org/jfreechart/.

(Joint, 2000) Joint Chiefs of Staff, (2000). *Joint Vision 2020*. Washington, DC: U.S. Government Printing Office. Available from http://www.dtic.mil/dtic/tr/fulltext/u2/a510839.pdf.

(Jouault et al., 2009) Jouault F., Jean B., Mikaël B., (2009). Towards an advanced model-driven engineering toolbox. *Innovations in Systems and Software Engineering*, 5, 5-12.

(JProfiler, 2012) JProfiler, (2012). *JProfiler Help*. Available from http://resources.ej-technologies.com/jprofiler/help/doc/.

(McCarty et al., 1998) McCarty B., Cassady-Dorion L., (1998). *Java Distributed Objects*. SAMS Publishing: Indianapolis.

(Mellon et al., 1995) Mellon L., West D., (1995). Architectural optimizations to advanced distributed simulation. *Proceedings of 27th Winter Simulation Conference*. 634-641.

(Mowbray et al., 1995) Mowbray T., Zahavi R., (1995). *The essential CORBA: systems integration using distributed objects*. Wiley Publishing: New York.

(Morris et al, 2004) Morris E., Levine L., Place P., Plakosh D., Meyers B., (2004). *System of Systems Interoperability (SOSI): Final Report.* Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania. Available from http://www.sei.cmu.edu/library/abstracts/reports/04tr004.cfm [accessed 07 July 2011]

(Möller et al., 2005) Möller B, Löf S., (2005). Mixing Service Oriented and High Level Architectures in Support of the GIG. *Proceedings of the 2005 Spring Simulation Interoperability Workshop*, 05S-SIW-064.

(Möller et al., 2007) Möller B., Clarence D., Mikael K., (2007). Developing Web Centric Federates and Federations using the HLA Evolved Web Services API. *Proceedings of 2007 Spring Simulation Interoperability Workshop,* 07S-SIW-107.

(MoDisco, 2012a) MoDisco, (2012). *MoDisco/Installation.* Available from http://wiki.eclipse.org/MoDisco/Installation.

(MoDisco, 2012b) MoDisco, (2012). *MoDisco User Guide.* Available from http://help.eclipse.org/indigo/index.jsp.

(NEHTA, 2005) NEHTA, (2005). Towards an Interoperability Framework, Version 1.8

(IST, 2005) Network of Excellence - Contract no.: IST-508011, (2005). *Deliverable D6.1: Practices, principles and patterns for interoperability*, France, University Bordeaux 1.

(NisB, 2010) NisB project, (2010). *The Network is the Business*, Available from http://www.nisb-project.eu/index.php [accessed 07 July 2011].

(OMG, 2003) OMG, (2003). *MDA Guide Version 1.0.1*. Object Management Group, Document number: OMG / 20030601. Available from www.omg.org/docs/omg/03-06-01.pdf [accessed 15 June 2009].

(OMG, 2006) OMG, (2006). *Object Constraint Language*. Document number: formal/06-05-01. Available from http://www.omg.org/spec/OCL/2.0/ [accessed 15 July 2010].

(OMG, 2010) OMG, (2010). *Architecture Driven Modernization (ADM): Knowledge Discovery Meta-Model (KDM) v1.2*, OMG, Document number: formal/2010-06-03. Available from http://www.omg.org/spec/KDM/1.2 [accessed 15 July 2010].

(OMG, 2011a) OMG, (2011). *Architecture-driven Modernization: Abstract Syntax Tree Metamodel (ASTM) v1.0*, OMG, Document number: formal/2011-01-05. Available from http://www.omg.org/spec/ASTM [accessed 25 April 2011].

(OMG, 2011b) OMG, (2011). *Business Process Model and Notation (BPMN) Version 2.0*. Document number: formal/2011-01-03. Available from http://www.omg.org/spec/BPMN/2.0 [accessed 25 April 2011].

(Oracle, 2012) Oracle, (2012). *The Java EE 6 Tutorial. PartIII.* Available from http://docs.oracle.com/javaee/6/tutorial/doc/bnayl.html.

(Özsu et al., 1991) Özsu T., Valduriez P., (1991). *Principles of Distributed Database Systems.* Prentice-Hall, Eaglewood Cliffs, New Jersey.

(Parr et al., 2003) Parr S., Keith-Magee R., (2003). The Next Step Applying the Model Driven Architecture to HLA. *Proceedings of the 2003 Spring Simulation Interoperability Workshop*, 03S-SIW -123.

(Postel, 1980) Postel J., (1980). *RFC 768: User Datagram Protocol*. Internet Engineering Task Force. Available from http://tools.ietf.org/html/rfc768.

(Pokorny et al., 2006) Pokorny T., Stratton D., Smith P., (2006). AOP and the HLA: Simplified Federation Development. *Proceedings of Fall Simulation Interoperability Workshop*. 06F-SIW-034

(poRTIco, 2009) poRTIco, (2009), *Developer Documentation*. Available from http://porticoproject.org/index.php?title=Developer_Documentation [accessed 15 July 2010]

(Richardson et al., 2007) Richardson L., Ruby S., (2007), *RESTful web services*, O'Reilly Media, Inc.

(Sammet, 1978) Sammet J., (1978). The Early History of COBOL. *ACM SIGPLAN Notices - Special issue: History of programming languages conference*. 13(8), 121–161.

(Sheth et al., 1990) Sheth A. P., Larson J. A., (1990). Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3), 183-236.

(Shanks, 1997) Shanks G., (1997). The RPR-FOM. A Reference Federation Object Model to Promote Simulation Interoperability. *Proceedings of Spring Simulation Interoperability Workshop*. 97S-SIW-135.

(SISO, 2011) *"Simulation Interoperability Standards Organization"*, accessed May 10, 2011, http://www.sisostds.org/Home.aspx

(Song et al., 2012) Song F., Zacharewicz G., Chen D., (2012). Multi-strategies Ontology Alignment Aggregated by AHP. *16th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, IOS Press*, San Sebastian, 1583-1592.

(StraBburger, 2001) StraBburger S., (2001). *Distributed Simulation Based on the High Level Architecture in Civilian Application Domains*. PhD Thesis, University of Magdeburg.

(Tolk, 2001) Tolk A., (2001). Bridging the Data Gap - Recommendations for Short, Medium and Long Term Solutions. *Spring Simulation Interoperability Workshop*. U.S.: Orlando, Florida

(Tolk, 2002) Tolk A., (2002). Avoiding another Green Elephant - A Proposal for the Next Generation HLA based on the Model Driven Architecture. *Proceedings of the 2002 Fall Simulation Interoperability Workshop*, 02F-SIW-004.

(Tolk et al., 2003) Tolk A., Muguira J. A., (2003). The Levels of Conceptual Interoperability Model. *Fall Simulation Interoperability Workshop.* U.S. Orlando, Florida.

(Touzi, 2007) Touzi J., (2007). *Aide à la conception de Système d'Information Collaboratif support de l'interopérabilité des entreprises*, PHD thesis, Institut National Polytechnique de Toulouse.

(Trbovich et al., 2005) Trbovich S., Reading R., (2005). Simulation and Software Development for Capabilities Based Warfare: An Analysis of Harmonized Systems Engineering Processes. *Proceedings Spring Simulation Interoperability Workshop*, 05S-SIW-106.

(Tu et al., 2010a) Tu Z., Zacharewicz G., Chen D., (2010). Unified Reversible Life Cycle for Future Interoperable Enterprise Distributed Information Systems. *IESA 2010-Interoperability for Enterprise Software & Applications 2010*, 57-66, Coventry (UK).

(Tu et al., 2010b) Tu Z., Zacharewicz G., Chen D., (2010), Harmonized and Reversible development framework for HLA based interoperable application. *The International Conference on Modelling and Applied Simulation part of The 7th International Mediterranean and Latin American Modelling Multiconference 2010.*

(Tu et al., 2011a) Tu Z., Zacharewicz G., Chen D., (2011), Harmonized and Reversible development framework for HLA based interoperable application. *Symposium On Theory of Modelling and Simulation (DEVS/TMS'11) part of Spring Simulation Multi-Conference 2011*, Boston USA.

(Tu et al., 2011b) Tu Z., Zacharewicz G., Chen D., (2011), DEVELOPING A WEB-ENABLE HLA FEDERATE BASED ON PORTICO RTI. *Proceedings of the 2011 Winter Simulation Conference,* Orlando USA.

(Tu et al., 2011c) Tu Z., Zacharewicz G., Chen D., (2011), A Harmonized and Reversible Development Framework for HLA-Based Interoperable Application. *INSIGHT Newsletter of INCOSE.* 14(4) 16-17.

(Tu et al., 2012a) Tu Z., Zacharewicz G., Chen D., (2012), Harmonized and Reversible development framework for HLA based interoperable application. *Handbook of Research on E-Business Standards and Protocols: Documents, Data and Advanced Web Technologies 2012* chapter 3.

(Tu et al., 2012b) Tu Z., Zacharewicz G., Chen D., (2012). Building a high-level architecture federated interoperable framework from legacy information systems. *International Journal of Computer Integrated Manufacturing*. Available online.

(Ullberg et al., 2007) Ullberg J., Chen D., Johnson P., (2007). *Barriers Driven Methodology For Enterprise Interoperability*. IFIP International Federation for Information Processing, 453-460.

(Veltman, 2001) Veltman K.H., (2001). Syntactic and Semantic Interoperability: New Approaches to Knowledge and the Semantic Web. *New review of information networking*. 7, 159-183

(Vernadat, 2007) Vernadat F.B., (2007). Interoperable enterprise systems: Principles, concepts, and methods. *Annual Reviews in Control,* 31, 137-145.

(W3C, 2001a) W3C, (2001). *Web Services Description Language (WSDL) 1.1*. Available from http://www.w3.org/TR/wsdl.

(W3C, 2001b) W3C, (2001). *DAML+OIL Web Ontology Language*. Available from http://www.w3.org/Submission/2001/12/.

(W3C, 2004a) W3C, (2004). *RDF Primer*. Available from http://www.w3.org/TR/rdf-primer/.

(W3C, 2004b) W3C, (2004). *OWL Web Ontology Language Overview*. Available from http://www.w3.org/TR/owl-features/.

(W3C, 2007) W3C, (2007). *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. Available from http://www.w3.org/TR/2007/REC-soap12-part1-20070427.

(W3C, 2008) W3C, (2008). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Available from http://www.w3.org/TR/2008/REC-xml-20081126/.

(Wang, 2000) Wang Y., (2000). *Discrete Mathematics Introduction*. Harbin Institute of Technology Press.

(Wagner, 2006) Wagner F., (2006). *Modeling Software with Finite State Machines: A Practical Approach*. Auerbach Publications.

(Weatherly, 1993) Weatherly R., Wilson A., Griffin S., (1993). ALSP-theory, experience, and future Directions. *Proceedings of 25th Winter Simulation Conference.* 1068-1072.

(Wiedemann, 2007) Wiedemann T., (2007). SOA-Conform Modeling As a Highlevel standard for Discrete Modeling and Simulation, *Proceedings 21st European Conference on Modelling and Simulation,* Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni

(Zacharewicz et al., 2008) Zacharewicz G., Chen D., Vallespir B., (2008). HLA Supported Federation Oriented Enterprise Interoperability, Application to Aerospace Enterprises. *Proceeedings of 2008 International Simulation Multiconference EuroSISO*, 08E-SIW-074, Edinburgh Scotland.

(Zacharewicz et al., 2009) Zacharewicz G., Chen D., Vallespir B., (2009). Short-Lived Ontology Approach for Agent/HLA Federated Enterprise Interoperability. Proceedings IEEE of International Conference I-ESA China 2009 Interoperability for Enterprise Software and Applications. 329-335, Beijing China.

(Zacharewicz et al., 2011) Zacharewicz G., Labarthe O., Chen D., Vallespir B., (2011). A Multi Agent/HLA Platform for Enterprises Interoperability: Short-Lived Ontology Based, *Electronic Supply Network Coordination in Intelligent and Dynamic Environment: Modeling and Implementation*, 319-346.

(Zeigler, 1984) Zeigler B.P., (1984). *Multifacetted Modeling and Discrete Event Simulation*. Academic Press, London; Orlando.

Annex 1: UML file reversed by MoDisco

Below is an example of the UML file reversed by MoDisco Tool. Note that some segments of XML code have been omitted, because the complete information takes more than 50 pages to illustrate.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<uml:Model xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
xmlns:uml="http://www.eclipse.org/uml2/3.0.0/UML" xmi:id="a1" name="root model">
  <packagedElement xmi:type="uml:Model" xmi:id="a2" name="AutoPartsSupplier">
    <packagedElement xmi:type="uml:Package" xmi:id="a3" name="autopartssupplier">
      <packagedElement xmi:type="uml:Class" xmi:id="a4" name="ManufactureOrderForm">
        <ownedAttribute xmi:type="uml:Property" xmi:id="a5" name="orderId" visibility="public" type="a660">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a6" value="1"/>
          <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a7"/>
        </ownedAttribute>
        <ownedAttribute xmi:type="uml:Property" xmi:id="a8" name="productId" visibility="public" type="a660">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a9" value="1"/>
          <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a10"/>
        </ownedAttribute>
        <ownedAttribute xmi:type="uml:Property" xmi:id="a11" name="productName" visibility="public"
type="a660">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a12" value="1"/>
          <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a13"/>
        </ownedAttribute>
        <ownedAttribute xmi:type="uml:Property" xmi:id="a14" name="orderQuantity" visibility="public"
type="a646">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a15" value="1"/>
          <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a16"/>
        </ownedAttribute>
        <ownedAttribute xmi:type="uml:Property" xmi:id="a17" name="deliverTime" visibility="public"
type="a660">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a18" value="1"/>
          <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a19"/>
        </ownedAttribute>
        <ownedOperation xmi:type="uml:Operation" xmi:id="a20" name="getDeliverTime" visibility="public">
          <ownedParameter xmi:type="uml:Parameter" xmi:id="a21" visibility="public" type="a660"
direction="return">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a22" value="1"/>
            <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a23"/>
          </ownedParameter>
        </ownedOperation>
        <ownedOperation xmi:type="uml:Operation" xmi:id="a24" name="setDeliverTime" visibility="public">
          <ownedParameter xmi:type="uml:Parameter" xmi:id="a25" visibility="public" type="a648"
direction="return">
```

<upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a26" value="1"/>
<lowerValue xmi:type="uml:LiteralInteger" xmi:id="a27"/>
</ownedParameter>
<ownedParameter xmi:type="uml:Parameter" xmi:id="a28" name="deliverTime" visibility="public" type="a660">
<upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a29" value="1"/>
<lowerValue xmi:type="uml:LiteralInteger" xmi:id="a30"/>
</ownedParameter>
</ownedOperation>
.....
</ownedOperation>
</packagedElement>
<packagedElement xmi:type="uml:Class" xmi:id="a75" name="AutpPartsSupplierApp" clientDependency="a546 a547">
<generalization xmi:type="uml:Generalization" xmi:id="a76"/>
<ownedOperation xmi:type="uml:Operation" xmi:id="a77" name="startup" visibility="protected">
<ownedParameter xmi:type="uml:Parameter" xmi:id="a78" visibility="public" type="a648" direction="return">
<upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a79" value="1"/>
<lowerValue xmi:type="uml:LiteralInteger" xmi:id="a80"/>
</ownedParameter>
</ownedOperation>
.....
<packagedElement xmi:type="uml:Class" xmi:id="a127" name="CustomerOrderForm">
<ownedAttribute xmi:type="uml:Property" xmi:id="a128" name="orderId" visibility="public" type="a660">
<upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a129" value="1"/>
<lowerValue xmi:type="uml:LiteralInteger" xmi:id="a130"/>
</ownedAttribute>
<ownedAttribute xmi:type="uml:Property" xmi:id="a131" name="productId" visibility="public" type="a660">
<upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a132" value="1"/>
<lowerValue xmi:type="uml:LiteralInteger" xmi:id="a133"/>
</ownedAttribute>
<ownedAttribute xmi:type="uml:Property" xmi:id="a134" name="productName" visibility="public" type="a660">
<upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a135" value="1"/>
<lowerValue xmi:type="uml:LiteralInteger" xmi:id="a136"/>
</ownedAttribute>
.....
<ownedOperation xmi:type="uml:Operation" xmi:id="a146" name="getOrderHost" visibility="public">
<ownedParameter xmi:type="uml:Parameter" xmi:id="a147" visibility="public" type="a660" direction="return">
<upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a148" value="1"/>
<lowerValue xmi:type="uml:LiteralInteger" xmi:id="a149"/>
</ownedParameter>

```
        </ownedOperation>
        <ownedOperation xmi:type="uml:Operation" xmi:id="a150" name="setOrderHost" visibility="public">
          <ownedParameter xmi:type="uml:Parameter" xmi:id="a151" visibility="public" type="a648"
direction="return">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a152" value="1"/>
            <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a153"/>
          </ownedParameter>
          <ownedParameter xmi:type="uml:Parameter" xmi:id="a154" name="orderHost" visibility="public"
type="a660">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a155" value="1"/>
            <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a156"/>
          </ownedParameter>
        </ownedOperation>
        ......
      </packagedElement>
      ......
      <packagedElement xmi:type="uml:Class" xmi:id="a451" name="ProductInformation">
        <ownedAttribute xmi:type="uml:Property" xmi:id="a452" name="product_id" visibility="public"
type="a660">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a453" value="1"/>
          <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a454"/>
        </ownedAttribute>
        <ownedAttribute xmi:type="uml:Property" xmi:id="a455" name="product_name" visibility="public"
type="a660">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a456" value="1"/>
          <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a457"/>
        </ownedAttribute>
        <ownedAttribute xmi:type="uml:Property" xmi:id="a458" name="product_category" visibility="public"
type="a660">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a459" value="1"/>
          <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a460"/>
        </ownedAttribute>
        ......
        <ownedOperation xmi:type="uml:Operation" xmi:id="a498" name="setProduct_id" visibility="public">
          <ownedParameter xmi:type="uml:Parameter" xmi:id="a499" visibility="public" type="a648"
direction="return">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a500" value="1"/>
            <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a501"/>
          </ownedParameter>
          <ownedParameter xmi:type="uml:Parameter" xmi:id="a502" name="product_id" visibility="public"
type="a660">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a503" value="1"/>
            <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a504"/>
          </ownedParameter>
        </ownedOperation>
```

```
        <ownedOperation xmi:type="uml:Operation" xmi:id="a505" name="setProduct_name" visibility="public">
          <ownedParameter xmi:type="uml:Parameter" xmi:id="a506" visibility="public" type="a648" direction="return">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a507" value="1"/>
            <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a508"/>
          </ownedParameter>
          <ownedParameter xmi:type="uml:Parameter" xmi:id="a509" name="product_name" visibility="public" type="a660">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a510" value="1"/>
            <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a511"/>
          </ownedParameter>
        </ownedOperation>
        </ownedOperation>
        <ownedOperation xmi:type="uml:Operation" xmi:id="a528" name="getProduct_id" visibility="public">
          <ownedParameter xmi:type="uml:Parameter" xmi:id="a529" visibility="public" type="a660" direction="return">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a530" value="1"/>
            <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a531"/>
          </ownedParameter>
        </ownedOperation>
        <ownedOperation xmi:type="uml:Operation" xmi:id="a532" name="getProduct_name" visibility="public">
          <ownedParameter xmi:type="uml:Parameter" xmi:id="a533" visibility="public" type="a660" direction="return">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a534" value="1"/>
            <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a535"/>
          </ownedParameter>
        </ownedOperation>
        <ownedOperation xmi:type="uml:Operation" xmi:id="a536" name="ProductInformation" visibility="public"/>
      </packagedElement>
      <packagedElement xmi:type="uml:Dependency" xmi:id="a546" supplier="a1336" client="a75"/>
      <packagedElement xmi:type="uml:Dependency" xmi:id="a547" supplier="a1337" client="a75"/>
      <packagedElement xmi:type="uml:Dependency" xmi:id="a548" supplier="a820" client="a99"/>
      <packagedElement xmi:type="uml:Dependency" xmi:id="a549" supplier="a1338" client="a311"/>
      ......
      <packagedElement xmi:type="uml:Association" xmi:id="a568" memberEnd="a100 a569">
        <ownedEnd xmi:type="uml:Property" xmi:id="a569" type="a99" association="a568"/>
      </packagedElement>
      <packagedElement xmi:type="uml:Association" xmi:id="a570" memberEnd="a115 a571">
        <ownedEnd xmi:type="uml:Property" xmi:id="a571" type="a99" association="a570"/>
      </packagedElement>
      <packagedElement xmi:type="uml:Association" xmi:id="a572" memberEnd="a313 a573">
        <ownedEnd xmi:type="uml:Property" xmi:id="a573" type="a311" association="a572"/>
      </packagedElement>
      ......
```

```
    </packagedElement>
    <packagedElement xmi:type="uml:Package" xmi:id="a642" name="Common Java datatypes">
      <packagedElement xmi:type="uml:PrimitiveType" xmi:id="a643" name="int"/>
      <packagedElement xmi:type="uml:PrimitiveType" xmi:id="a644" name="long"/>
      <packagedElement xmi:type="uml:PrimitiveType" xmi:id="a645" name="float"/>
      ......
    </packagedElement>
  </packagedElement>
  <packagedElement xmi:type="uml:Model" xmi:id="a653" name="externals">
    <packagedElement xmi:type="uml:Package" xmi:id="a654" name="java">
      <packagedElement xmi:type="uml:Package" xmi:id="a655" name="lang">
        <packagedElement xmi:type="uml:Package" xmi:id="a656" name="reflect">
          <packagedElement xmi:type="uml:Interface" xmi:id="a657" name="GenericDeclaration"/>
          <packagedElement xmi:type="uml:Interface" xmi:id="a658" name="Type"/>
          <packagedElement xmi:type="uml:Interface" xmi:id="a659" name="AnnotatedElement"/>
        </packagedElement>
        <packagedElement xmi:type="uml:Class" xmi:id="a660" name="String" clientDependency="a661 a662
a663">
          <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="a661" supplier="a717" client="a660"
contract="a717"/>
          <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="a662" supplier="a669" client="a660"
contract="a669"/>
          <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="a663" supplier="a673" client="a660"
contract="a673"/>
          <ownedOperation xmi:type="uml:Operation" xmi:id="a664" name="equals" visibility="public">
            <ownedParameter xmi:type="uml:Parameter" xmi:id="a665" name="arg0" visibility="public"
type="a697">
              <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a666" value="1"/>
              <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a667"/>
            </ownedParameter>
          </ownedOperation>
          <ownedOperation xmi:type="uml:Operation" xmi:id="a668" name="trim" visibility="public"/>
        </packagedElement>
        ......
      </packagedElement>
      <packagedElement xmi:type="uml:Package" xmi:id="a859" name="util">
        <packagedElement xmi:type="uml:Interface" xmi:id="a860" name="EventListener"/>
        <packagedElement xmi:type="uml:Class" xmi:id="a861" name="EventObject"
clientDependency="a862">
          <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="a862" supplier="a717" client="a861"
contract="a717"/>
        </packagedElement>
        <packagedElement xmi:type="uml:Class" xmi:id="a863" name="ArrayList" clientDependency="a868
a869 a870 a871">
          <ownedTemplateSignature xmi:type="uml:RedefinableTemplateSignature" xmi:id="a864"
```

name="ArrayList<E>" parameter="a865">
<ownedParameter xmi:type="uml:ClassifierTemplateParameter" xmi:id="a865" parameteredElement="a866">
<ownedParameteredElement xmi:type="uml:Class" xmi:id="a866" name="E" templateParameter="a865"/>
</ownedParameter>
</ownedTemplateSignature>
<generalization xmi:type="uml:Generalization" xmi:id="a867" general="a882"/>
<interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="a868" supplier="a898" client="a863" contract="a898"/>
<interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="a869" supplier="a903" client="a863" contract="a903"/>
<interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="a870" supplier="a711" client="a863" contract="a711"/>
<interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="a871" supplier="a717" client="a863" contract="a717"/>
<ownedOperation xmi:type="uml:Operation" xmi:id="a872" name="size" visibility="public"/>
<ownedOperation xmi:type="uml:Operation" xmi:id="a873" name="get" visibility="public">
<ownedParameter xmi:type="uml:Parameter" xmi:id="a874" name="arg0" visibility="public" type="a643">
<upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a875" value="1"/>
<lowerValue xmi:type="uml:LiteralInteger" xmi:id="a876"/>
</ownedParameter>
</ownedOperation>
<ownedOperation xmi:type="uml:Operation" xmi:id="a877" name="ArrayList" visibility="public"/>
<ownedOperation xmi:type="uml:Operation" xmi:id="a878" name="add" visibility="public">
<ownedParameter xmi:type="uml:Parameter" xmi:id="a879" name="arg0" visibility="public" type="a697">
<upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a880" value="1"/>
<lowerValue xmi:type="uml:LiteralInteger" xmi:id="a881"/>
</ownedParameter>
</ownedOperation>
</packagedElement>
.....
<packagedElement xmi:type="uml:Package" xmi:id="a971" name="event">
<packagedElement xmi:type="uml:Interface" xmi:id="a972" name="TableModelListener">
<generalization xmi:type="uml:Generalization" xmi:id="a973" general="a860"/>
</packagedElement>
.....
</packagedElement>
<packagedElement xmi:type="uml:Class" xmi:id="a982" name="JDialog" clientDependency="a984 a985 a986 a987">
<generalization xmi:type="uml:Generalization" xmi:id="a983" general="a765"/>
<interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="a984" supplier="a998" client="a982" contract="a998"/>

```
        <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="a985" supplier="a925" client="a982" contract="a925"/>
        <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="a986" supplier="a1002" client="a982" contract="a1002"/>
        <interfaceRealization xmi:type="uml:InterfaceRealization" xmi:id="a987" supplier="a1005" client="a982" contract="a1005"/>
        <ownedOperation xmi:type="uml:Operation" xmi:id="a988" name="JDialog" visibility="public">
          <ownedParameter xmi:type="uml:Parameter" xmi:id="a989" name="arg0" visibility="public" type="a783">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a990" value="1"/>
            <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a991"/>
          </ownedParameter>
        </ownedOperation>
        <ownedOperation xmi:type="uml:Operation" xmi:id="a992" name="getRootPane" visibility="public"/>
        <ownedOperation xmi:type="uml:Operation" xmi:id="a993" name="setDefaultCloseOperation" visibility="public">
          <ownedParameter xmi:type="uml:Parameter" xmi:id="a994" name="arg0" visibility="public" type="a643">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="a995" value="1"/>
            <lowerValue xmi:type="uml:LiteralInteger" xmi:id="a996"/>
          </ownedParameter>
        </ownedOperation>
        <ownedOperation xmi:type="uml:Operation" xmi:id="a997" name="getContentPane" visibility="public"/>
      </packagedElement>
      .....
    </packagedElement>
    <packagedElement xmi:type="uml:Package" xmi:id="a1335" name="org"/>
    <packagedElement xmi:type="uml:Interface" xmi:id="a1336" name="org.jdesktop.application.Application"/>
    <packagedElement xmi:type="uml:Interface" xmi:id="a1337" name="org.jdesktop.application.SingleFrameApplication"/>
    .....
  </packagedElement>
  <packagedElement xmi:type="uml:Model" xmi:id="a1342" name="source references">
    <packagedElement xmi:type="uml:Artifact" xmi:id="a1343" name="ManufactureOrderForm.java" fileName="F:\workspace1\AutoPartsSupplier\src\autopartssupplier\ManufactureOrderForm.java"/>
    <packagedElement xmi:type="uml:Artifact" xmi:id="a1344" name="AutpPartsSupplierApp.java" fileName="F:\workspace1\AutoPartsSupplier\src\autopartssupplier\AutpPartsSupplierApp.java"/>
    <packagedElement xmi:type="uml:Artifact" xmi:id="a1345" name="DatabaseFactory.java" fileName="F:\workspace1\AutoPartsSupplier\src\autopartssupplier\DatabaseFactory.java"/>
    .....
  </packagedElement>
</uml:Model>
```

Annex 2: FOM generation

Below is a list of the code for FOM generation.

```
package objectanalysis;

import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;

/**
 * this class is used for model transformation
 * @author Zhiying Tu
 */
public class WriteToFile {

    public static boolean generateFOM (HashMap<String, ArrayList<String>> fomMap) {
        try {
            //Convenience class for writing character files. boolean:true,append
            FileWriter fw = new FileWriter("testFOM.fed",false);
            //Print formatted representations of objects to a text-output stream.
            PrintWriter out = new PrintWriter(fw);

            //load poRTIco FOM template
            String topMsg = loadTemplate();
            //Create new FOM
            out.write(topMsg);

            Iterator iterator = fomMap.keySet().iterator();
            while(iterator.hasNext()){
                String className = iterator.next().toString();

                //Find UML:class
                ArrayList attrList = fomMap.get(className);
                if (attrList.size() == 0) {
                    //Transform to RTIObjects:class
                    out.println("\n\t\t\t(class " + className + ")");
                } else {
```

```
                    //Transform to RTIObjects:class
                    out.println("\n\t\t\t(class " + className);
                    for(int i = 0; i < attrList.size(); i++){
                        //Find UML:Attributes, and transform them into RTIObjects:Attributes
                        out.println("\n\t\t\t\t(attribute " + attrList.get(i) + " reliable timestamp TestSpace)");
                    }
                    out.println("\n\t\t\t)");
                }
            }
            out.println("\n\t\t)");
            out.println("\n\t)");
            out.println("\n)");

            out.close();
            fw.close();
            return true;
        } catch (IOException e) {
            System.out.println("Write file error!");
            e.printStackTrace();
            return false;
        }
    }

    private static String loadTemplate() {
        String template = "(FED\n\t(Federation Portico-Test)\n\t\t(FEDversion v1.3)\n\t\t(spaces" +
                "\n\t\t\t(space TestSpace\n\t\t\t\t(dimension TestDimension)\n\t\t\t)\n\t\t(space OtherSpace\n\t\t\t"
+
                "(dimension OtherDimension)\n\t\t\t)\n\t\t)\n\t(objects\n\t\t(class ObjectRoot\n\t\t\t"+
                "(attribute privilegeToDelete reliable timestamp)\n\t\t\t(class RTIprivate)";

        return template;
    }
}
```

Annex 3: RTI specific code

Below is a list of RTI specific code.

```
// ////////////////////////////////////////////////////////////////////
    // ////////////////////////// Main Simulation Method ////////////////////////
    // ////////////////////////////////////////////////////////////////////

    /**
     * This is the main simulation loop. It can be thought of as the main method
     * of the federate. For a description of the basic flow of this federate,
     * see the class level comments
     */
    public void runFederate(String federateName) throws RTIexception {
        // ////////////////////////////////
        // 1. create the RTIambassador //
        // ////////////////////////////////
        rtiamb = RtiFactoryFactory.getRtiFactory().createRtiAmbassador();

        // ////////////////////////////
        // 2. create the federation //
        // ////////////////////////////
        // create
        // NOTE: some other federate may have already created the federation,
        // in that case, we'll just try and join it
        try {
            File fom = new File("testfom.fed");
            rtiamb.createFederationExecution("ExampleFederation",
                    fom.toURI().toURL());
            log("Created Federation");
        } catch (FederationExecutionAlreadyExists exists) {
            IS_FIRST = false;
            log("Didn't create federation, it already existed");
        } catch (MalformedURLException urle) {
            IS_FIRST = false;
            log("Exception processing fom: " + urle.getMessage());
            urle.printStackTrace();
            return;
        }

        // ////////////////////////
```

```
// 3. join the federation //
// ////////////////////////
// create the federate ambassador and join the federation
fedamb = new Example13FederateAmbassador();
rtiamb.joinFederationExecution(federateName, "ExampleFederation",
        fedamb);
log("Joined Federation as " + federateName);

// ////////////////////////////
// 4. announce the sync point //
// ////////////////////////////
// announce a sync point to get everyone on the same page. if the point
// has already been registered, we'll get a callback saying it failed,
// but we don't care about that, as long as someone registered it
if (IS_FIRST) {
    rtiamb.registerFederationSynchronizationPoint(federateName, null);
} else {
    rtiamb.registerFederationSynchronizationPoint(READY_TO_RUN, null);
}
// wait until the point is announced
while (fedamb.isAnnounced == false) {
    rtiamb.tick();
}

// WAIT FOR USER TO KICK US OFF
// So that there is time to add other federates, we will wait until the
// user hits enter before proceeding. That was, you have time to start
// other federates.

// ////////////////////////////////////////////////////
// 5. achieve the point and wait for synchronization //
// ////////////////////////////////////////////////////
// tell the RTI we are ready to move past the sync point and then wait
// until the federation has synchronized on
rtiamb.synchronizationPointAchieved(READY_TO_RUN);
log("Achieved sync point: " + READY_TO_RUN
        + ", waiting for federation...");
while (fedamb.isReadyToRun == false) {
    rtiamb.tick();
```

```
        }

        // ////////////////////////////
        // 6. enable time policies //
        // ////////////////////////////
        // in this section we enable/disable all time policies
        // note that this step is optional!
        enableTimePolicy();
        log("Time Policy Enabled");

        // ////////////////////////////
        // 7. publish and subscribe //
        // ////////////////////////////
        // in this section we tell the RTI of all the data we are going to
        // produce, and all the data we want to know about
        publishAndSubscribe();
        log("Published and Subscribed");

        // /////////////////////////////////////
        // 8. register an object to update //
        // /////////////////////////////////////
        objectHandle = registerObject();
        log("Registered Object, handle=" + objectHandle);

        int classHandle2 = rtiamb.getObjectClassHandle("ObjectRoot.CarWheelSupplier");
        objectHandle2 = rtiamb.registerObjectInstance(classHandle2);
        log("Registered Object, handle=" + objectHandle2);

        while(true){
            String feedback = businessAdapter.handleBusiness(this.getMsg());
            advanceTime(1.0);
            log( "Time Advanced to " + fedamb.federateTime );
            if(feedback.split(@)[0].trim.equals(“quit”))break;
        }
//                  // 9.1 update the attribute values of the instance //
//
    updateAttributeValues( objectHandle ,allCarDays-i,currentState,announceWheel,announceEngine,count);
//                  // 9.2 send an interaction
//                  sendInteraction(allCarDays-i,currentState,announceWheel,announceEngine,count);
```

```
//                      // 9.3 request a time advance and wait until we get it
//                      advanceTime( 1.0 );
//                      log( "Time Advanced to " + fedamb.federateTime );


            // ////////////////////////////////////
            // 10. delete the object we created //
            // ////////////////////////////////////
            deleteObject(objectHandle);
            log("Deleted Object, handle=" + objectHandle);

            // ////////////////////////////////////
            // 11. resign from the federation //
            // ////////////////////////////////////
            rtiamb.resignFederationExecution(ResignAction.NO_ACTION);
            log("Resigned from Federation");

            // ////////////////////////////////////////
            // 12. try and destroy the federation //
            // ////////////////////////////////////////
            // NOTE: we won't die if we can't do this because other federates
            // remain. in that case we'll leave it for them to clean up
            try {
                  rtiamb.destroyFederationExecution("ExampleFederation");
                  log("Destroyed Federation");
            } catch (FederationExecutionDoesNotExist dne) {
                  log("No need to destroy federation, it doesn't exist");
            } catch (FederatesCurrentlyJoined fcj) {
                  log("Didn't destroy federation, federates still joined");
            }
      }

      private String getMsg() {
            return fedamb.recMsg;
      }
      private void setMsg(String msg) {
            fedamb.recMsg="";
      }
      private void setLastRecMsg(String msg) {
```

```
            fedamb.lastRecMsg="";
        }
public Example13Federate(){
}

        // ////////////////////////////////////////////////////////////////////
        // ////////////////////////////// Helper Methods
        // ////////////////////////////////
        // ////////////////////////////////////////////////////////////////////
        /**
         * This method will attempt to enable the various time related properties
         * for the federate
         */
        private void enableTimePolicy() throws RTIexception {
            // NOTE: Unfortunately, the LogicalTime/LogicalTimeInterval create code
            // is
            // Portico specific. You will have to alter this if you move to a
            // different RTI implementation. As such, we've isolated it into a
            // method so that any change only needs to happen in a couple of spots
            LogicalTime currentTime = convertTime(fedamb.federateTime);
            LogicalTimeInterval lookahead = convertInterval(fedamb.federateLookahead);

            // ////////////////////////////
            // enable time regulation //
            // ////////////////////////////
            this.rtiamb.enableTimeRegulation(currentTime, lookahead);

            // tick until we get the callback
            while (fedamb.isRegulating == false) {
                rtiamb.tick();
            }

            // ////////////////////////////
            // enable time constrained //
            // ////////////////////////////
            this.rtiamb.enableTimeConstrained();

            // tick until we get the callback
            while (fedamb.isConstrained == false) {
```

```
            rtiamb.tick();
        }
}

/**
 * This method will inform the RTI about the types of data that the federate
 * will be creating, and the types of data we are interested in hearing
 * about as other federates produce it.
 */
private void publishAndSubscribe() throws RTIexception {
        int classHandle = rtiamb.getObjectClassHandle("ObjectRoot.CarManufactureSupplier");
        int aaHandle = rtiamb.getAttributeHandle("wheelMessage", classHandle);
        AttributeHandleSet attributes = RtiFactoryFactory.getRtiFactory()
                        .createAttributeHandleSet();
        attributes.add(aaHandle);
        rtiamb.publishObjectClass(classHandle, attributes);
        rtiamb.subscribeObjectClassAttributes(classHandle, attributes);
        int interactionHandle = rtiamb
                        .getInteractionClassHandle("InteractionRoot.CarManufactureSupplier");
        rtiamb.publishInteractionClass(interactionHandle);
        rtiamb.subscribeInteractionClass(interactionHandle);

        int classHandle2 = rtiamb.getObjectClassHandle("ObjectRoot.CarWheelSupplier");
        AttributeHandleSet attributes2 = RtiFactoryFactory.getRtiFactory()
        .createAttributeHandleSet();
        int abHandle = rtiamb.getAttributeHandle("dayToFinish", classHandle2);
        attributes2.add(abHandle);
        int acHandle = rtiamb.getAttributeHandle("currentState", classHandle2);
        attributes2.add(acHandle);
        int adHandle = rtiamb.getAttributeHandle("type", classHandle2);
        attributes2.add(adHandle);
        int aeHandle = rtiamb.getAttributeHandle("price", classHandle2);
        attributes2.add(aeHandle);

        rtiamb.publishObjectClass(classHandle2, attributes2);
        rtiamb.subscribeObjectClassAttributes(classHandle2, attributes2);
        int interactionHandle2 = rtiamb
                        .getInteractionClassHandle("InteractionRoot.CarWheelSupplier");
        rtiamb.publishInteractionClass(interactionHandle2);
```

```
        rtiamb.subscribeInteractionClass(interactionHandle2);

    }

    /**
     * This method will register an instance of the class ObjectRoot.A and will
     * return the federation-wide unique handle for that instance. Later in the
     * simulation, we will update the attribute values for this instance
     */
    private int registerObject() throws RTIexception {
        int classHandle = rtiamb.getObjectClassHandle("ObjectRoot.CarManufactureSupplier");
        return rtiamb.registerObjectInstance(classHandle);
    }

    /**
     * This method will update all the values of the given object instance. It
     * will set each of the values to be a string which is equal to the name of
     * the attribute plus the current time. eg "aa:10.0" if the time is 10.0.
     * <p/> Note that we don't actually have to update all the attributes at
     * once, we could update them individually, in groups or not at all!
     */
    private void updateAttributeValues(int objectHandle,int dayToFinish ,int currentState,int announceWheel, int
announceEngine,int count) throws RTIexception {
        SuppliedAttributes attributes = RtiFactoryFactory.getRtiFactory()
                .createSuppliedAttributes();
        byte[] aaValue =
EncodingHelpers.encodeString(String.valueOf("wheelDays;"+this.carWheelDays*count+";"+dayToFinish));
        int classHandle = rtiamb.getObjectClass(objectHandle);

        int aaHandle = rtiamb.getAttributeHandle("dayToFinish", classHandle);
        attributes.add(aaHandle, aaValue);

        byte[] abValue = EncodingHelpers.encodeString(String.valueOf(currentState));
        int abHandle = rtiamb.getAttributeHandle("currentState", classHandle);
        attributes.add(abHandle, abValue);

        byte[] acValue = EncodingHelpers.encodeString("wheelType;215/70R15");
        int acHandle = rtiamb.getAttributeHandle("type", classHandle);
        attributes.add(acHandle, acValue);
```

```
        byte[] adValue = EncodingHelpers.encodeString("wheelPrice;"+300*count);
        int adHandle = rtiamb.getAttributeHandle("price", classHandle);
        attributes.add(adHandle, adValue);

        rtiamb.updateAttributeValues(objectHandle, attributes, generateTag());
        LogicalTime time = convertTime(fedamb.federateTime
                + fedamb.federateLookahead);
        rtiamb.updateAttributeValues(objectHandle, attributes, generateTag(),
                time);
    }


    private void sendInteraction(int dayToFinish,int currentState,int announceWheel,int announceEngine,int
count) throws RTIexception {
        SuppliedParameters parameters = RtiFactoryFactory.getRtiFactory()
                .createSuppliedParameters();
        byte[] xaValue = EncodingHelpers.encodeString("dayToFinish:" + dayToFinish);
        int classHandle = rtiamb.getInteractionClassHandle("InteractionRoot.CarWheelSupplier");
        int xaHandle = rtiamb.getParameterHandle("dayToFinish", classHandle);
        parameters.add(xaHandle, xaValue);

        byte[] xbValue = EncodingHelpers.encodeString("currentState:" + currentState);
        int xbHandle = rtiamb.getParameterHandle("currentState", classHandle);
        parameters.add(xbHandle, xbValue);

        byte[] xcValue = EncodingHelpers.encodeString("type:" + "215/70R15");
        int xcHandle = rtiamb.getParameterHandle("type", classHandle);
        parameters.add(xcHandle, xcValue);
        byte[] xdValue = EncodingHelpers.encodeString("price:" + 300*count);
        int xdHandle = rtiamb.getParameterHandle("price", classHandle);
        parameters.add(xdHandle, xdValue);

        rtiamb.sendInteraction(classHandle, parameters, generateTag());
        LogicalTime time = convertTime(fedamb.federateTime
                + fedamb.federateLookahead);
        rtiamb.sendInteraction(classHandle, parameters, generateTag(), time);
    }
```

```
/**
 * This method will request a time advance to the current time, plus the
 * given timestep. It will then wait until a notification of the time
 * advance grant has been received.
 */
private void advanceTime(double timestep) throws RTIexception {
        // request the advance
        fedamb.isAdvancing = true;
        LogicalTime newTime = convertTime(fedamb.federateTime + timestep);
        rtiamb.timeAdvanceRequest(newTime);

        // wait for the time advance to be granted. ticking will tell the
        // LRC to start delivering callbacks to the federate
        while (fedamb.isAdvancing) {
                rtiamb.tick();
        }
}

/**
 * This method will attempt to delete the object instance of the given
 * handle. We can only delete objects we created, or for which we own the
 * privilegeToDelete attribute.
 */
private void deleteObject(int handle) throws RTIexception {
        rtiamb.deleteObjectInstance(handle, generateTag());
}
private String getLastRecMsg() {
        return fedamb.lastRecMsg;
}
private double getLbts() {
        return fedamb.federateTime + fedamb.federateLookahead;
}

private byte[] generateTag() {
        return ("" + System.currentTimeMillis()).getBytes();
}
```

Résumé: L'interopérabilité est une des caractéristiques requises pour les entreprises évoluantdans un marché globalisé à la concurrencecroissanteet complexe. Dans la dernière décennie, l'interopérabilité des entreprises a été développée et prescrite par différents types de cadres, de méthodes et de techniques. Cependant, le développement de l'interopérabilité n'est pas encore assez mature pour être considéré en tant que science à part entière. Par ailleurs, il ne cesse d'évoluer en fonction des besoins des entreprises,de leursenvironnementset desdifférents secteurs d'activité. Aujourd'hui, l'environnement s'organise en réseaux multipleet provoque d'imprévisibles situations liées à leurs dynamiques (création, modification, résilience). Ainsi l'interopérabilité durable devient une dimension nouvelle de recherche pour l'interopérabilité des systèmes d'entreprise et de leurs domaines d'applications. Dans l'interopérabilitédurable, l'interopérabilité d'entreprise dynamique est l'un des thèmes focaux. Cette approche dynamique, également appelée «fédérée», est originaire du cadre d'interopérabilité de l'Entreprise proposée dans le Réseau d'Excellence (NoE) INTEROP. Il vise à donner la capacité aux entreprises d'établir une interopérabilité à la volée sans connaissance préalable des informations à échanger. Cette thèse présente l'état actuel des travaux qui se rapprochent du développement de l'interopérabilité des entreprises «fédérés» endynamique. Ces travaux de thèse mettent tout d'abord en évidence l'intérêt de la redécouverte de modèles à partir d'un système existant avant de concevoir un futur système. Unemethodologie de réverse engineering dirigée par les modèles et basée sur la norme de simulation distribuée HLAest proposée pour concevoir et développerpar l'approche fédérée d'interopérabilité le futur système d'information de l'entreprise. La phase de mise en œuvre réutilise les concepts d'interopérabilité issusde la simulation distribuée pour faciliter et coordonner la communication entre les systèmes d'information distribués hétérogènes des entreprises en combinant avec les dernières orientation service actuelle du web. La plate-forme tend ainsi à satisfaire les attentes de la dernière version du standard de l'architecture de haut niveau HLA 1516 Evolved. Ce cadre propose donc un cycle complet de développement pour qui a l'intention de réutiliser un système d'information existant sans recoder ex-nilo, mais en l'adaptant aux nouvelles exigences de la dynamique d'interopérabilité.

Mots clés: L'interopérabilité d'entreprise; Dynamique; Approche fédérée; HLA; MDA; Ingénierie inverse

Abstract: Interoperability is one of the requisite features for existing enterprises in the increasing competitive and complex global market. In the last decade, enterprise interoperability has been developed and prescribed by various kinds of frameworks, methods, and techniques. However interoperability development is still not mature enough to become a science. Meanwhile, it keeps evolving according to different business requirement and market environment. Nowadays, networked environment causes unpredictable dynamical situations, thus sustainable interoperability becomes a new research dimension in the interoperability of enterprise systems and applications domain. In the sustainable interoperability, enterprise interoperability dynamics is one of the focal topics. This dynamic approach also called federated is originated from Enterprise Interoperability Framework of INTEROP NoE, which aims to establish interoperability on the fly. This thesis presents current state on federated approaches to develop enterprise interoperability dynamics. Based on this study, a reversible model driven and HLA based methodology is proposed for achieving federated approach for Enterprise Interoperability. It reuses distributed simulation interoperability concepts to facilitate and coordinate the communication between heterogeneous distributed information systems of the enterprises. The platform is complaint with the latest version of the High Level Architecture (HLA) that is a distributed communication standard. This framework is also proposing a development lifecycle that intends to reuse existing information systems without recoding them but by adapting them to the new requirements of interoperability dynamics.

Key words: Enterprise Interoperability; Dynamic; Federated approach; HLA; MDA; Model Reverse

Laboratoire de l'Intégration du Matériau au Système (IMS)
Université Bordeaux 1.
351, Cours de la Libération – 33405 Talence Cedex.
Tél. : (33) 05 40 00 36 25
http://www.ims-bordeaux.fr