# Integration of model driven engineering and ontology approaches for solving interoperability issues

Hui Liu

▶ **To cite this version:**

Hui Liu. Integration of model driven engineering and ontology approaches for solving interoperability issues. Other. Ecole Centrale de Lille, 2011. English. NNT : 2011ECLI0015 . tel-00662511

**HAL Id: tel-00662511**

**https://theses.hal.science/tel-00662511**

Submitted on 24 Jan 2012

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

**N° d'ordre : 165**

**ECOLE CENTRALE DE LILLE**

# THESE

présentée en vue
d'obtenir le grade de

# DOCTEUR

en

Spécialité : Génie Industriel

par

**LIU Hui**

**DOCTORAT DELIVRE PAR L'ECOLE CENTRALE DE LILLE**

Titre de la thèse :
**Integration of model driven engineering and ontology approaches for solving interoperability issues**

**Intégration des approches ontologiques et d'ingénierie dirigée par les modèles pour la résolution de problèmes d'interopérabilité**

Soutenue le 13 Octobre 2011 devant le jury d'examen :

| | |
|---|---|
| **Président** | *Henri BASSON, Professeur, Université du Littoral Côte d'Opale* |
| **Rapporteur** | *Hervé PINGAUD, Professeur, Université J.-F. Champollion* |
| **Rapporteur** | *Jean BEZIVIN, Professeur Emérite, Université de Nantes* |
| **Examinateur** | *David CHEN, Professeur, Bordeaux 1* |
| **Examinateur** | *Henri BASSON, Professeur, Université du Littoral Côte d'Opale* |
| **Examinateur** | *Michel BIGAND, Maître de Conférences HDR, EC-Lille* |
| **Directeur de thèse** | *Jean-Pierre BOUREY, Professeur, EC-Lille* |

Thèse préparée dans le Laboratoire de Modélisation et de Management des Organisations EA 4344

Ecole Doctorale SPI 287 (EC Paris, EC Lille, INT Evry)

**PRES Université Lille Nord-de-France**

# Acknowledgements

During the last three years of my PhD study, I have met many people inside or outside our lab LM²O who gave me the possibility to finish my thesis enjoyably. I would like to say THANK YOU from my heart to all of YOU.

First of all, I am sincerely indebted to my supervisor Prof. Jean-Pierre BOUREY. Thank you for offering me the opportunity to develop my doctoral study in LM²O. Thank you for introducing me to the world of “enterprise interoperability” and sharing your precious time to discuss problems encountered in my study. Besides, you have also recommended me some interesting academic activities (seminars, workshops and conferences), and, these activities have broadened my academic horizon. During the activities, I have also gotten acquainted with many researchers in our research domain. This provides possibilities for me to collaborate with them in the future. Thank you again, Prof. BOUREY.

Furthermore, I want to express my reconnaissance to my colleagues Prof. Anne-Françoise CUTTING-DECELLE and Prof. Michel BIGAND. Thank you for your help to my PhD study. I’d also like to thank Vincent MEISSNER, Karine GHESQUIER, Virginie LECLERCQ for helping me in the administrative work.

To my PhD colleagues Youness LEMRABET and David CLIN, and three masters Xinxin LI, Yinglong ZHAO and Jinzhou YANG, thank you for your valuable discussion during my doctoral study. Especially, to Youness, thank you for your help in my published papers.

My special appreciation goes to Ms. Hélène CATSIAPIS. In your French courses, I have learned much interesting French culture. Thanks to you, I have visited Paris, Loire Valley, the Champagne Region and Giverny (the Garden of Claude Monet), etc. All of the trips are extraordinary and memorable. They have enhanced my understanding of the French culture.

Besides, I gratefully thank all my friends, particularly, Xiaoting CHEN, Ling PENG, Lihui YANG, Yifan WANG, Lian LIAN, Jin ZHAO, Pengfei MU, Jian LIU, Jinlin GONG, Dapeng YANG, Jing YANG and Huarong WANG etc. All of you made my life not just filled with study but with entertainment and happiness.

Most importantly, I would like to express my deep gratitude to my family, especially to my parents. You are in China but you keep in close contact with me and encourage me all the time.

Finally, my doctoral study is based on the partnership agreement between the China Scholarship Council (CSC) and the five-French-Ecoles-Centrales Intergroup. The partnership agreement was signed on 12 September 2005. So I would like to thank all of the people who have contributed to the construction of the agreement. I am also sincerely grateful to CSC and Ecole Centrale de Lille who finance my academic activities.

In a word, the last three years in France will be my precious and unforgettable memory. Thank China. Thank France.

# Intégration des approches ontologiques et d’ingénierie dirigée par les modèles pour la résolution de problèmes d’interopérabilité

## Résumé étendu en français

La mondialisation économique et l’accélération à l’échelle mondiale des échanges de biens et services obligent les entreprises à collaborer entre elles pour améliorer leur compétitivité. L’entreprise utilise les services disponibles des autres entreprises pour construire son propre portefeuille de services. Ensuite, elle expose ses propres services aux autres entreprises. Les entreprises doivent aussi être agiles au niveau métier en intégrant leurs ressources afin de fournir une réponse rapide et efficace aux changements continus des exigences métier dictés par le marché ou dirigés par les clients, les partenaires, ou les fournisseurs. Pour résoudre ces deux défis, les entreprises considèrent leur système d'information (SI) comme un levier pour automatiser leurs collaborations. De nos jours, l’utilisation de plusieurs SIs pour supporter la collaboration entre plusieurs entreprises est un vrai défi connu sous le nom de problème d'interopérabilité d'entreprise illustrée sur la Figure 1.

Dans son environnement métier, l’entreprise réalise et améliore son métier à travers des activités métiers (Figure 1). Cet environnement contient toutes les activités et les informations de l'entreprise. L’environnement collaboratif est créé par l’interaction entre plusieurs environnements métiers. L’environnement IT a, quant à lui, la responsabilité d’automatiser ces activités métier par des dispositifs de communication, des systèmes d'information, etc. L’environnement d’interopérabilité est créé par l’interaction de plusieurs environnements IT. Certaines activés métiers ne sont pas supportées par l’environnement IT, par exemple l’activité « une société installe des téléphones pour une autre société ». Dans une collaboration, il y a aussi d’autres types d’interactions qui ne sont pas automatisables, par exemple l’interaction entre les personnes. Ainsi, d’une part la collaboration inter-entreprises doit être supportée par le biais des interactions entre les environnements métiers et ITs de chacune des entreprises. D’autre part, la collaboration d'entreprises doit être aussi supportée par des interactions entre des environnements métiers et IT (interaction entre les environnements de collaboration et d'interopérabilité). En

effet, pendant la collaboration, l'environnement IT /d'interopérabilité a généralement besoin d'interventions humaines pour saisir ou envoyer des informations.

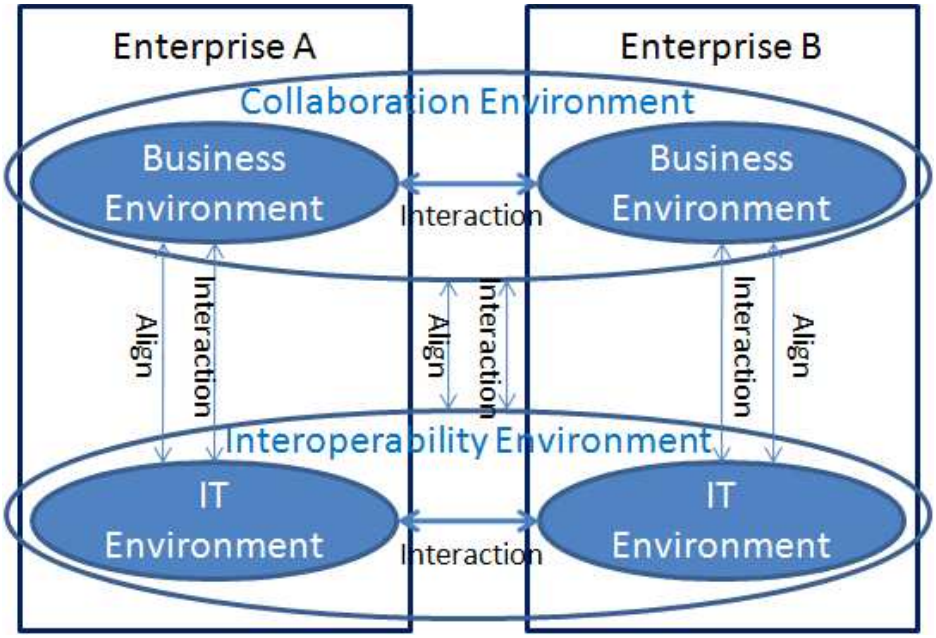



**Figure 1. Problème d'interopérabilité d'entreprise**

Dans une collaboration interentreprises, il faut aligner les environnements de collaboration et d'interopérabilité afin de réduire l'écart entre ces deux environnements. Les exigences de collaboration dans l'environnement de collaboration doivent être réalisées par les SI dans l'environnement d'interopérabilité. Cet alignement permet à l'environnement d'interopérabilité d'être agile et de s'adapter plus facilement aux changements dans l'environnement de collaboration. Cependant, les méthodes d'alignement sont influencées par les méthodes d'implémentations dans l'environnement de l'interopérabilité. Par exemple, (Touzi 2007) propose de générer un système d'information collaboratif dans un environnement d'interopérabilité à travers la transformation des modèles dans un environnement de collaboration. Le système d'information collaboratif joue un rôle de «médiateur» entre des systèmes d'information des entreprises. Cette méthode a été également adoptée par (Truptil 2011) pour résoudre un problème de gestion de crises. Mais il est possible de mettre en œuvre les exigences de collaboration sans médiateur, en utilisant uniquement le système d'information de chaque entreprise. Cette méthode d'alignement évite la dépendance au médiateur et permet aux entreprises plus de contrôle sur la collaboration. Dans cette thèse nous proposons une telle méthode d'alignement.

Notre travail est basé sur Model Driven Engineering [1] (MDE) et sur l'utilisation des ontologies. MDE est une méthodologie de développement logiciel, qui vise à élever le niveau d'abstraction dans la spécification du programme pour favoriser l'automatisation dans le développement (Batory 2006). MDE s'appuie sur deux notions fondamentales : les modèles et les transformations de modèles. Cela permet la séparation des préoccupations par niveaux d'abstraction. A chaque niveau d'abstraction des modèles sont élaborés en utilisant des domain-specific languages[2] ou des langages standardisés comme UML. Les transformations de modèles sont utilisées pour automatiser autant que possible le développement de logiciels et pour renforcer les liens entre les niveaux d'abstraction ce qui augmente la traçabilité. Cette démarche initialement prévue pour le développement de logiciels peut être adaptée dans le cadre de l'alignement. L'approche MDE est donc été retenue pour notre travail sur l'alignement métier et IT. MDA[3](Model Driven Architecture) est la vue de l'OMG (Object Management Group) qui entre dans le cadre MDE. Notre travail se focalise davantage sur MDA.

Dans les collaborations d'entreprises, les systèmes d'information, distribués et hétérogènes, s'échangent des données qui peuvent être hétérogènes. Les problèmes de l'hétérogénéité des données peuvent être divisés en deux niveaux : l'hétérogénéité syntaxique et l'hétérogénéité sémantique. L'hétérogénéité syntaxique (Goh, 1997) peut être causée par des conflits de types de données, des conflits d'étiquetage, des conflits d'agrégation, des conflits de généralisation entre des différentes bases de données/systèmes d'information. L'hétérogénéité sémantique (Goh 1997) provient principalement des conflits de noms, des conflits d'échelle et des unités et des conflits d'interprétation entre des différents systèmes. Afin de réaliser l'interopérabilité sémantique, les ontologies et les technologies basées sur la sémantique vont jouer un rôle clé (Wache, Vögele et al 2001; Uschold et Grüninger 2004). Notre travail de recherche étant basé sur l'alignement, il est nécessaire de prendre en compte également les aspects sémantiques. Notre travail est donc naturellement lié aux ontologies.

Cette thèse apporte des éléments de réponse à la question principale suivante : comment l'architecture dirigée par les modèles et l'étude des ontologies peuvent contribuer à résoudre les problèmes d'interopérabilité d'entreprise ?

[1] L'ingénierie dirigée par les modèles

[2] Langues dédiées à un domaine

[3] Architecture dirigée par les modèles

Tout d'abord, nous présentons une synthèse des travaux sur l'interopérabilité d'entreprise à partir de quatre dimensions principales : sa définition, son cadre, ses solutions et ses modèles de maturité. Ensuite, nous positionnons notre travail sur les trois dimensions suivantes : cadre, solutions et modèles de maturité. Puis, nous soulignons l'interopérabilité d'entreprise à travers l'alignement métier-IT pour soutenir des collaborations entre des entreprises.

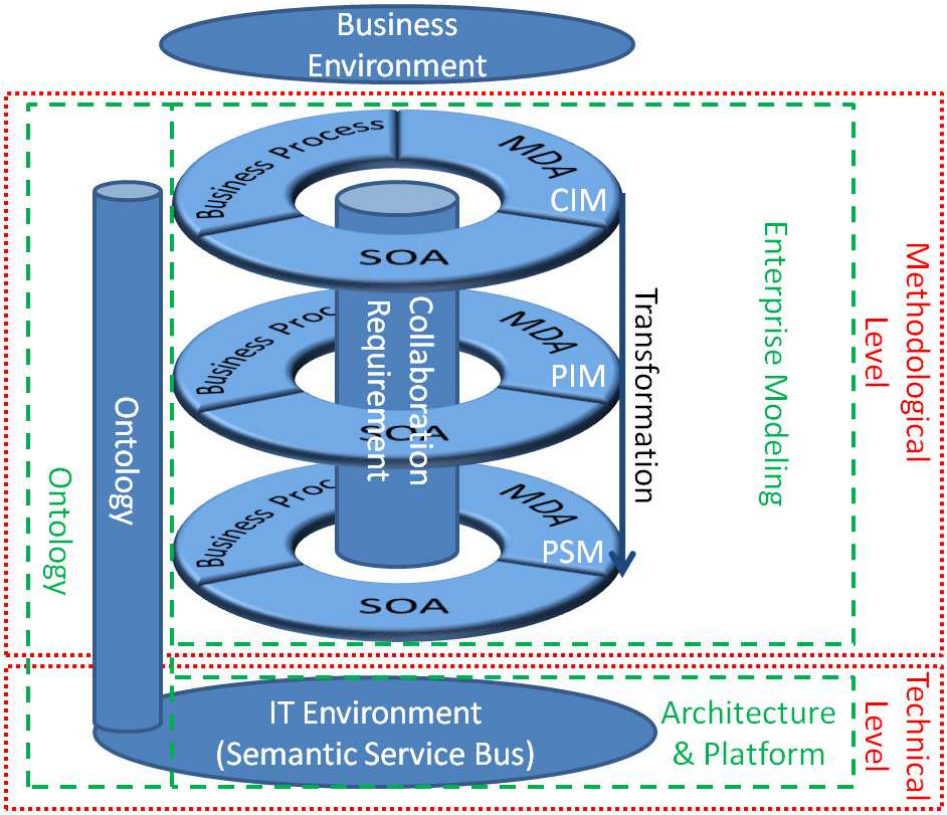



**Figure 2. Cadre pour des solutions IT aux problèmes d'interopérabilité pour une entreprise**

Afin d'aligner le métier et l'IT, nous étudions cinq domaines de recherche sur l'interopérabilité d'entreprises : les processus métier collaboratifs, MDA, SOA (Service Oriented Architecture), ESB (Enterprise Service Bus) et l'ontologie. Ensuite, nous proposons un cadre pour des solutions IT à des problèmes d'interopérabilité. Ce cadre, présenté sur la Figure 2, devra être mis en œuvre dans toutes les entreprises participant à la collaboration. Le cadre commence à partir de l'environnement métier et se termine à l'environnement IT. Au niveau méthodologique (dans le rectangle supérieur rouge), le cadre utilise des processus métiers, MDA, SOA et l'ontologie pour aligner les environnements métier et IT. Au niveau technique (dans le rectangle inférieur rouge), le cadre s'appuie sur un ESB et l'ontologie (ESB sémantique) comme la plate-

forme / infrastructure de l'environnement IT. Ce cadre couvre également trois domaines clés sur l'interopérabilité d'entreprise proposés dans (Chen et Doumeingts 2003) : la modélisation d'entreprise, l'architecture & la plate-forme et l'ontologie. Ces trois domaines clés sont identifiés dans les rectangles verts sur la Figure 2.

Afin de réaliser le cadre proposé précédemment, nous proposons une « Méthode Basée sur des Processus pour l'Interopérabilité d'Entreprise » (MBPIE) au niveau méthodologique, et une « Architecture Basée sur l'Ontologie et Dirigée par les Buts (BOGD) pour l'Interopérabilité d'Entreprise » au niveau technique.

La MBPIE est basée sur l'ontologie et constituée de cinq étapes (niveaux) principales :

**Etape 1 :** le point de départ consiste à définir un processus collaboratif ;

**Etape 2 :** les activités sont annotées avec des informations précisant notamment quels sont les collaborateurs. On obtient le processus collaboratif annoté par les collaborateurs ; ensuite, on fusionne les activités voisines qui appartiennent au même collaborateur, puis on intègre les activités qui appartiennent aux différents collaborateurs pour obtenir un processus global simplifié d'interopérabilité et plusieurs sous-processus d'interopérabilité. Cette étape est basée sur deux critères quantitatifs : le rang de processus collaboratif et le taux de coopération. Afin d'expliquer cette étape, nous introduisons un cas d'étude appelé « ShoppingDrive ».

Les deux étapes précédentes sont globales pour tous les collaborateurs alors que les étapes suivantes sont locales pour chaque collaborateur. Si, dans une collaboration d'entreprises, il n'y a pas de coopérateur principal, le processus collaboratif est créé par la négociation de tous les collaborateurs qui doivent respecter les étapes suivantes.

**Etape 3 :** chaque collaborateur transforme les (sous-)processus collaboratifs en ses propres processus collaboratifs reposant sur sa propre définition des terminologies métiers ;

**Etape 4 :** chaque collaborateur fixe les types de messages dans ses propres (sous-)processus et transforme les collaborateurs au niveau CIM (Computation Independent Model) en participants au niveau PIM/PSM (Platform Independent Model/Platform Specific Model) ;

**Etape 5 :** tous les (sous-)processus d'interopérabilité sont développés en utilisant des langages de description de processus et exécutés en respectant un même algorithme d'exécution de processus.

Les cinq étapes ci-dessus constituent la première variante de la MBPIE. Si, dans une collaboration d'entreprises, il y a un coopérateur principal, le processus collaboratif est créé par ce dernier. Après les première et deuxième étapes, le coopérateur principal exécute directement

les quatrième et cinquième étapes de la MBPIE. Les autres collaborateurs exécutent les troisième, quatrième et cinquième étapes. Ceci constitue la deuxième variante de la MBPIE. Pour les deux variantes de la MBPIE, l'usage de l'ontologie est différent.

Dans la MBPIE, les processus collaboratifs et leurs transformations (surtout les transformations entre les deuxième, troisième, quatrième et cinquième étapes) sont tous basés sur l'ontologie. Un processus collaboratif est annoté avec des informations sémantiques. Dans nos travaux, un processus collaboratif est exprimé en BPMN2.0 et nous proposons quatre méthodes basées sur l'ontologie pour ajouter des informations sémantiques dans des processus métier. Ces annotations sémantiques seront utilisées dans des transformations de processus. Durant la transformation des processus, de nouvelles informations ontologiques sont ajoutées dans les processus. Elles contribueront au processus d'exécution.

Afin de réaliser le cadre de la Figure 2 au niveau technique, une architecture Basée sur l'Ontologie et Dirigée par les Buts (BODB) est proposée. Le cœur de cette architecture est un bus de services sémantiques. Ce bus est basé sur l'ontologie et dirigé par les buts. Il s'appuie sur un mécanisme symétrique pour l'invocation de services sémantiques. Le mécanisme symétrique est conçu en étendant le protocole SOAP (Simple Object Access Protocol). Cette extension est appelée SOAP BODB. Ce protocole est constitué de trois parties : un format du message BODB, un module SOAP BODB et un modèle de traitement de SOAP BODB. Le mécanisme symétrique a trois propriétés de transparence (emplacement, sémantique et technique) qui sont essentielles à l'interopérabilité et à l'exécution des processus d'interopérabilité. Cette architecture peut déployer le bus BODB dans des styles différents pour supporter l'interopérabilité d'intra- ou d'inter-entreprises. Notamment, il peut déployer le bus BODB dans un style fédéré pour supporter interopérabilité d'inter-entreprise.

La MBPIE et l'architecture BODB ont une relation étroite. Dans la MBPIE, à la deuxième étape, les processus métiers collaboratifs et ses sous-processus seront exposés à d'autres collaborateurs. Cela dépend de la transformation horizontale du processus qui est prise en charge par l'architecture BODB. Par ailleurs, dans la MBPIE, des processus d'interopérabilité exécutables seront générés. L'exécution des processus est supportée par un moteur de processus dans l'architecture BODB.

La MBPIE et l'architecture BODB sont tous fondés sur l'ontologie. L'influence de l'ontologie sur la MBPIE et l'architecture BODB est présentée dans le tableau 1. Nous analysons

l'influence de l'ontologie à partir de trois préoccupations d'interopérabilité : l'interopérabilité de données, l'interopérabilité de services et l'interopérabilité de processus. Les trois préoccupations ont été définies dans (Chen et Daclin 2006). Selon le tableau 1, des obstacles conceptuels dans les aspects de données, services et processus peuvent être supprimés par la MBPIE (au niveau méthodologique sur la Figure 2). Les obstacles techniques aux aspects des données, services et processus peuvent être supprimés par l'architecture BODB (au niveau technique sur la Figure 2).

Par ailleurs, la MBPIE et l'architecture BODB constituent ensemble une approche fédérée à des problèmes d'interopérabilité d'entreprise. La méthode MBPIE est fédérée parce qu'à l'exception de ses deux premières étapes, elle est respectée et exécutée séparément par tous les collaborateurs. En plus, chaque collaborateur est autonome. Dans une architecture BODB le bus peut être déployé pour supporter des collaborations entre des entreprises dans un style fédéré. Donc, cette architecture peut soutenir la fédération pour résoudre des problèmes d'interopérabilité d'entreprise.

**Table 1. Influence de l'ontologie sur la MBPIE et l'architecture BODB**

| **Préoccupations d'interopérabilité** | **MBPIE** | **l'architecture BODB** |
|---|---|---|
| Processus | Base d'ontologie (description sémantique sur des processus, etc.); annotations sémantiques dans des processus; transformation verticale basée sur l'ontologie | Conteneur de composants (moteur de processus base sur l'ontologie); bus BODB (transformation horizontale basée sur l'ontologie) |
| Service | Base d'ontologie (description sémantique sur des services, etc.); annotations sémantiques dans des processus; | Bus BODB (STEP 2, et STEP OI-2) |
| Données | Base d'ontologie (terminologie métier, etc.); annotations sémantiques dans des processus; | Bus BODB (STEP 1, STEP 3, STEP OI-1 et STEP OI-3) |

En résumé, notre travail propose la conception d'une approche fédérée pour résoudre des problèmes d'interopérabilité d'entreprise. L'approche fédérée permet de réaliser l'interopérabilité au niveau conceptuel et technique en prenant en considération trois préoccupations d'interopérabilité : données, services et processus.

Notre proposition a cependant quelques limites. Tout d'abord, la MBPIE et l'architecture BODB dépendent étroitement de l'ontologie. En effet, le niveau d'interopérabilité qu'ils peuvent atteindre est déterminé par la qualité des ontologies et de la capacité de mapping entre les ontologies. Mais aussi, l'architecture BODB est basée sur SOAP BODB, donc, son protocole de transport est limité à SOAP.

Enfin, notre étude réalisée dans cette thèse constitue une première ébauche de solution. Cependant d'autres pistes restent à explorer. D'un point de vue purement technique, il faut construire des outils logiciels et des plates-formes pour supporter la MBPIE et l'architecture BODB. D'autre part, ATL devrait être étendu pour pouvoir invoquer des services externes au cours de la transformation de modèle. Cela est nécessaire dans la transformation des processus basée sur l'ontologie. D'un point de vue scientifique, la découverte de services dirigée par les buts et la découverte de fournisseurs de services doivent être étudiées. Elles sont utilisées dans le bus BODB. Dans l'architecture BODB, un moteur de processus basés sur l'ontologie devrait également être étudié pour supporter l'exécution des processus d'interopérabilité. En outre, la composition des processus métier dirigée par des buts doit être étudiée. Dans notre étude, un modèle de « buts » a été proposé, et ce modèle sera un bon début pour construire une telle approche de composition automatique de processus métier collaboratifs.

# Contents

# Figure List

# Table List

# *INTRODUCTION*

Nowadays, with the deep development of economic globalization, enterprises tend to collaborate closely with others to improve their competitiveness by using other enterprises' valuable services as its own complement and make its own services potentially used by others. In addition, as business requirements from market, customers and partners are often changed, enterprises have to integrate their resources to provide fast and efficient responses, i.e., realize business agility. In order to resolve the above two problems, enterprises usually want to take advantage of their information systems to automate their collaboration and adapt themselves to changes in the collaboration. How to support such collaboration and related changes by information systems of different enterprises is a big problem, and it is described as the enterprise interoperability problem, illustrated in Figure 1.

In Figure 1, a business environment of an enterprise is to realize and improve its own business values through business activities, and it includes all the activities and information about business. A business environment can interact with another business environment; such interaction generates a collaboration environment between enterprises. Instead, IT environment automates activities in the business environment by communication devices, computers and information systems, etc. IT environment of an enterprise can also interact with others and such interaction generates an interoperability environment. As not all of collaboration tasks between enterprises can be supported by IT environment, such as "a company installs telephones for another company", hence during collaborations, there are also interactions between enterprises (between persons). So, collaboration between enterprises must be supported by interactions between business environments and by interactions between IT environments. Furthermore, enterprise collaboration must also be supported by interactions between business and IT environments or by interactions between collaboration and interoperability environments, because during collaboration, IT/interoperability environment usually needs persons to input some information or sends out some information to persons.

In order to realize collaboration between enterprises, a gap between the collaboration environment and interoperability environment should be aligned, that is to say collaboration requirements in collaboration environment must be realized by information systems in interoperability environments. The alignment will make interoperability environment adapted more agilely to changes in collaboration environment. However, the alignment methods will be influenced by implementation methods in interoperability environment. For example, (Touzi 2007) generated a collaborative information system in interoperability environment through model transformation from collaboration environment, and the collaborative

information system plays a role "mediator" between different enterprise information systems. This method is also adopted in (Truptil 2011) to resolve a crisis management problem. Besides the above alignment method, collaboration requirements can also be implemented without mediator, only with information system of each enterprise. This alignment method avoids the dependency on mediators and makes enterprises have more control of their collaboration. Our work is motivated by the desire to find such alignment method.

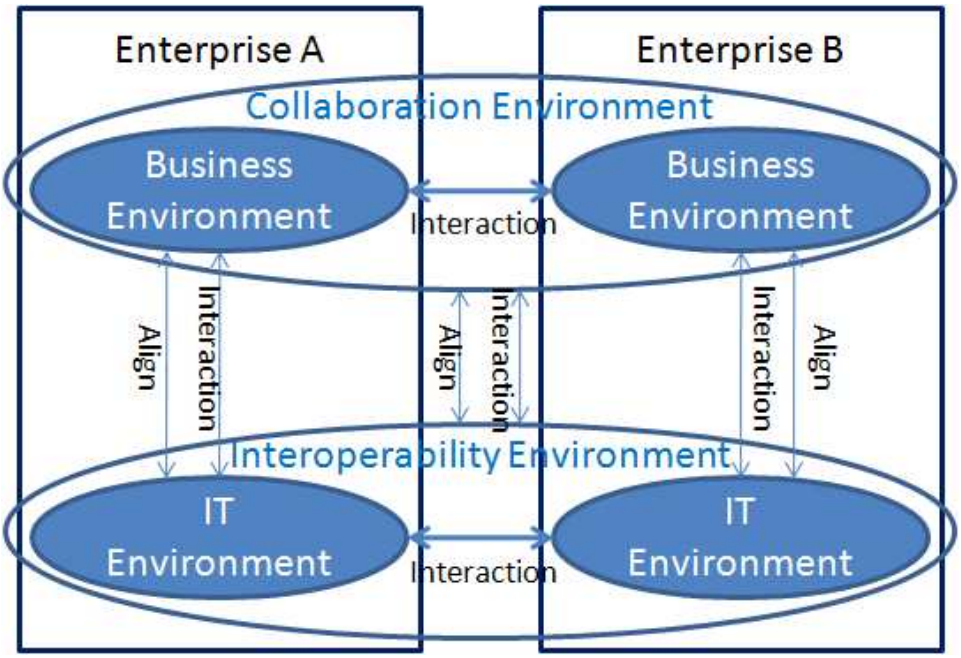



**Figure 1. Enterprise Interoperability Problem**

Our work is based on Model Driven Engineering (MDE) and Ontology. MDE is a software development methodology, which aims to raise the level of abstraction in program specification and increase automation in program development (Batory 2006). MDE has two core concepts: model and model transformation. Model in domain-specific languages focuses on higher-level specification of programs. Model transformation is used to automate software development. So MDE is beneficial to our work when aligning business and IT. MDA is Object Management Group (OMG)'s view on MDE. Our work focuses more on MDA.

In enterprise collaborations, distributed and heterogeneous information systems from different enterprises will exchange data with each other. The exchanged data may be heterogeneous. Problems caused from data heterogeneity can be divided into two levels: syntactic heterogeneity and semantic heterogeneity. Syntactic heterogeneity (Goh 1997) may be caused by data type conflicts, labeling conflicts, aggregation conflicts, generalization conflicts between different databases/information systems. Semantic heterogeneity (Goh 1997) primarily comes from naming conflicts, scaling and units conflicts and confounding

conflicts between different systems. In order to achieve semantic interoperability, ontologies and semantics-based technologies in general will play a key role to overcome the problem of semantic heterogeneity (Wache, Vögele et al. 2001; Uschold and Gruninger 2004). So our work must be also related with ontology.

How can we integrate MDE and ontology to solve enterprise interoperability problems? This thesis will respond to the question. Our work in this thesis is organized in the following structure:

1) Chapter 1 summarizes the research about enterprise interoperability from four main dimensions: its definition, framework, solutions and maturity models. During the summarization, our work is positioned in the dimensions. Finally, this chapter points out a research direction to enterprise interoperability: aligning business and IT to support collaborations between enterprises.
2) In order to align business and IT for enterprise collaborations, Chapter 2 analyzes the related research domains about enterprise interoperability: collaborative business process, MDA, SOA, ESB and ontology. Then, this chapter proposes a framework for IT solutions to interoperability problems. The framework integrates closely the above five research domains together to align business and IT and meanwhile to satisfy enterprise collaboration requirements.
3) In order to realize the framework proposed in Chapter 2, Chapter 3 will propose a "Process-Based Method for Enterprise Interoperability" (PBMEI), which employs collaborative processes to represent collaboration requirements between enterprises. PBMEI transforms a collaborative process to multiple executable interoperability processes according to two quantitative criteria: rank of collaborative process and cooperation rate. In order to explain PBMEI, a case named "ShoppingDrive" cooperation process is studied.
4) Chapter 4 presents the ontology usage and contents in PBMEI. Collaborative processes and process transformations in PBMEI are all grounded in ontology. In PBMEI, collaborative process is annotated with semantic information. As collaborative process is expressed in BPMN2.0, therefore this chapter proposes our ontology-based methods to annotate semantic information into BPMN2.0-based business processes. Such semantic annotations are used in process transformations. During process transformation, new ontology information is added into processes and such ontology information will contribute to process execution.

5) In order to support execution of interoperability processes generated in PBMEI, Chapter 5 designs an ontology-based and goal-driven (OBGD) architecture for enterprise interoperability. The core of the architecture is OBGD semantic service bus. This service bus is based on a symmetric mechanism for OBGD service invocation. The symmetric mechanism is designed according to OBGD Simple Object Access Protocol (SOAP) which is composed of OBGD message format definition, SOAP module definition and SOAP processing model definition. In collaborations, enterprises are usually independent of each other, so semantic service buses for enterprises are usually organized in a federated style. The federated deployment of semantic service buses is also discussed in this chapter. At last, this chapter analyzes the relationship between ontology-based PBMEI and OBGD architecture.

# *CHAPTER 1: Enterprise Interoperability*

Interoperability has been widely studied in many domains, such as e-Health (Stegwee and Rukanova 2003; NEHTA 2005), e-Government (EIF 2004; Gottschalk 2009), enterprise software applications (Chen and Doumeingts 2003), modeling and simulation domain (Wang, Tolk et al. 2009) and military domain (C4ISR-Interoperability-Working-Group 1998), etc. In different domains, researchers have described and defined interoperability from different viewpoints and they have not achieved a general consensus. In order to study further, researchers have also constructed different interoperability frameworks. The purpose of the frameworks is to provide an organizing mechanism so that concepts, problems and knowledge on interoperability can be represented in a more structured way (Chen, Doumeingts et al. 2008). Beside interoperability frameworks, researchers have also studied evaluation mechanisms of interoperability. The mechanisms evaluate the extent to which the interoperability can be achieved. The evaluation mechanisms are named differently in different academic papers, for example stages-of-growth (Gottschalk 2009), maturity levels (Gottschalk 2009), maturity model (C4ISR-Interoperability-Working-Group 1998), conceptual model (Tolk and Muguira 2003; Tolk, Diallo et al. 2007) (Wang, Tolk et al. 2009) or reference model (NATO 2003). This section will summarize some definitions and frameworks and maturity models about interoperability, especially about enterprise interoperability. This section will also position our work in these research domains.

## I. WHAT IS INTEROPERABILITY?

During researching enterprise interoperability, many literatures have proposed their own definitions of interoperability, some of which are listed in Table 1-1. According to the definitions in Table 1-1, the interoperability entities can be components, devices or communicating entities. All of them can be regarded as systems at different levels. So, interoperability between enterprises can be regarded as a system of systems, and enterprise interoperability will have some emergency properties (Fisher 2006), such as location transparency, semantics transparency and technique transparency and these properties will be discussed in more details in Chapter 4. In addition, except (Chen and Doumeingts 2003), all of the definitions are focused on information[1] exchange and use at the ICT (Information Communication Technology) level. That is because, in enterprise interoperability, information

[1] Information is data equipped with meaning (Schreiber et al 1999). The information can be simple, such as a message (SOAP message), or complex, such as a model (a business process model in BPMN, or data model in database). The information can be little, such as the value of a person's salary, or very large, such as the information of all the books in a library. The information can be plaintext, or can be encrypted/compressed.

systems[2] (IS) from different enterprises are usually distributed and heterogeneous, and in order to resolve interoperability problem, how to make such information systems exchange information and understand and use the exchanged information is the first encountered problem.

**Table 1-1. Definitions of Interoperability**

| Reference | Definition |
|---|---|
| (IEEE 1990) | **Interoperability** is the ability of two or more **systems or components** to exchange information and to use the information that has been exchanged. |
| (ISO-14258 1998) | **Interoperability** may occur between two (or more) **entities** that are related to one another in one of three ways:<br>• *Integrated* where there is a standard format for all constituent systems<br>• *Unified* where there is a common meta-level structure across constituent models, providing a means for establishing semantic equivalence<br>• *Federated* where models must be dynamically accommodated rather than having a predetermined meta-model. |
| (IEC-TC65/290/DC 2002) | **Interoperability** is the ability of two or more **devices**, regardless of manufacturer, to work together in one or more distributed **applications**. The application data, their semantic and application related functionality of each device is so defined that, should any device be replaced with a similar one of different manufacture, all distributed applications involving the replaced device will continue to operate as before the replacement, but with possible different dynamic responses. |
| (Chen and Doumeingts 2003) | **Interoperability** is considered as achieved only if the interaction between two **systems** can, at least, take place at the three levels: ***data, resource* and *business process*** with the semantics defined in a business context. |
| (Morris, Levine et al. 2004) | Interoperability is defined as: the ability of a set of communicating **entities** to (1) exchange specified state data and (2) operate on that state data according to specified, agreed-upon, operational semantics. |
| (Fisher 2006) | **Interoperation**, also called **interoperability**, has to do with the exchange and use of information necessary for effective operation of a system of **systems**. |

For the aspect of information exchange (see Figure 1-1), numerous network[3] devices (e.g., hub, switch, router, gateway, etc) have been constructed and deployed all over the world to connect devices (e.g., personal computers, computer servers) of different enterprises. Meanwhile, some protocols and standards for describing information transport and information format have also been proposed and widely used, for example, TCP/IP, HTTP,

[2] This article follows the definition of information system in (Alter 1999).

[3] Networks can be computer networks, wireless communication networks or TV/telephone networks, but this paper will focus more on computer networks.

JMS, XML and SOAP and so on; especially, to support the information exchange at the enterprise-level, some middleware and architecture styles have also been proposed, such as EAI, CORBA, ESB, P2P, SOA (web service, RESTful service (Fielding 2000)) and SMDA (Service Model Driven Architecture) (Xu, Mo et al. 2007). For the aspect of information understanding and using, numerous ontology languages have been proposed, for example, OKBC, OIL, OWL-S[4], WSMO[5], WSDL-S[6], SAWSDL[7], PIF (Polyak, Lee et al. 1998), some of which are XML-based, some are not and some of which are used to represent knowledge, some are used to describe Internet resources and some are used to describe business processes. Ontology will be discussed in detail in Chapter 2-Section V.

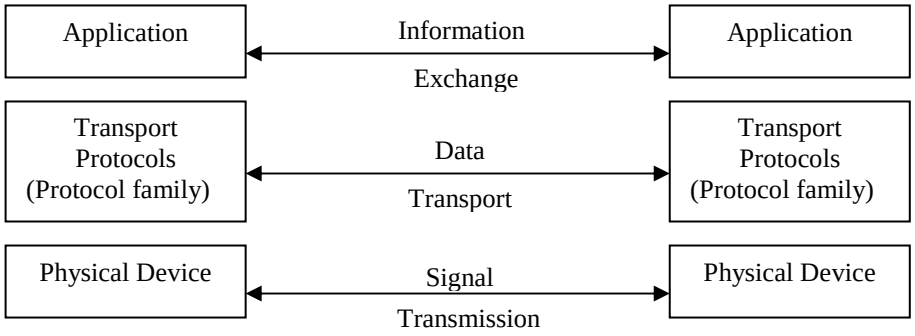



**Figure 1-1. Information Exchange between Enterprises**

## II. INTEROPERABILITY FRAMEWORK

With further study, enterprise interoperability can take place not only at ICT level but also at other levels, such as process level, business level. Therefore, relevant interoperability frameworks have been proposed, some of which are shown in Figure 1-2. In Figure 1-2, the first framework is proposed in (Chen and Doumeingts 2003). In this framework, interoperability must be achieved at three levels of an enterprise, including business environment and business processes at business level, organizational roles, skills and competencies of employees and knowledge assets at knowledge level, and applications, data and communication components at ICT level. Semantics traverse the three levels and provide necessary mutual understanding between enterprises (Chen and Doumeingts 2003). (Chen and Doumeingts 2003) also proposed a roadmap for enterprise interoperability research, which integrates three main research domains:

[4] http://www.daml.org/services/owl-s/

[5] http://www.wsmo.org/

[6] http://www.w3.org/Submission/WSDL-S/

[7] http://lsdis.cs.uga.edu/projects/meteor-s/SAWSDL/

a) **enterprise modeling** (EM) dealing with the representation of the inter-networked organization to establish interoperability requirements;
b) **architecture & platform** (A&P) defining the implementation solution to achieve interoperability;
c) **ontologies** addressing the semantics necessary to assure interoperability.

The roadmap is then supported by several European projects, such as ATHENA[8], CROSSWORK(Mehandjiev, Stalker et al. 2006), INTEROP[9], ECOLEAD[10], etc. INTEROP has enriched the work of (Chen and Doumeingts 2003) and proposed its own interoperability framework in (Kosanke 2006) (described in Figure 1-2 (b)); this framework identifies three categories of barriers at four levels. The "barrier" means "incompatibility", "mismatch" or "heterogeneity" which impede the sharing and exchange of information (Chen and Daclin 2006). The three kinds of barriers are explained as follows (Chen and Daclin 2006):

- *Conceptual barriers* include syntactic and semantic incompatibility. For example, different people or systems use different structures to represent information and knowledge; or information in models or software has no clearly defined semantics to avoid misunderstanding.
- *Technical barriers* are concerned with ICT level. They can be incompatibility of communication protocols, operating systems, infrastructures, IT architecture & platforms or techniques used to represent exchanged information, etc.
- *Organizational barriers* are related with the incompatibility of organization structures and management techniques performed in different enterprises.

The relationship among the three kinds of barriers is orthogonal: conceptual barriers are oriented to business information problems; technical barriers are oriented to machine problems; organizational barriers are oriented to human problems. Instead, the four levels in the framework: data, service, process and business have a dependency relationship between them at the functional aspect. **Data** is used by services; **services** are employed in **processes** to realize **business** objectives of enterprises. The four levels are shown in the following list (adapted from (Chen and Daclin 2006)):

[8] http://www.athena-ip.org

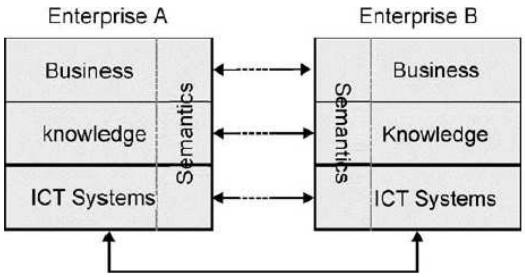
[9] http://www.interop-vlab.eu/

[10]http://ecolead.vtt.fi/

- *Data interoperability*: operate together different data models/bases using different query languages.
- *Service interoperability*: identify, compose and operate together various application services.
- *Process interoperability*: make various processes work together.
- *Business interoperability*: make organizations or companies work in a harmonized way in spite of different modes of decision-making, methods of work, legislations, company cultures and commercial approaches etc.

In the above framework, developing interoperability solutions means to remove the barriers according to enterprises' ability (Chen and Daclin 2006). But how can we develop interoperability solutions? There are three possible approaches defined in (ISO-14258 1998): integrated, unified and federated approaches. The three approaches have been narrated in Table 1-1. The integrated approach demands companies to share the same information models; the unified approach requires the same meta-model; however, the federated approach needs no common models or meta-models but it may need ontology to establish interoperability. Generally speaking, each of the three approaches makes companies increasingly more flexible to cope with interoperability problems.

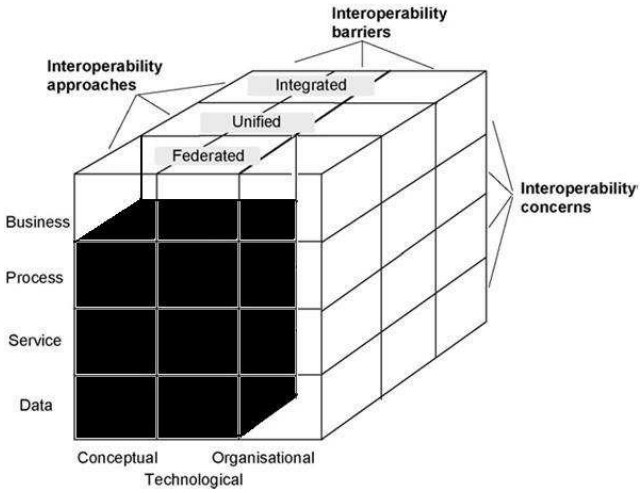
When adding the three approaches into Figure 1-2 (b), then (Chen and Daclin 2006) has generated a new enterprise interoperability framework shown in Figure 1-2 (c). Figure 1-2 (c) makes the three approaches as the third dimension. Our work is focused on the conceptual and technical barriers at the data, service process levels. For the third dimension, our work is associated with the federated approach (see black cubes in Figure 1-2 (c)).




(a) Simplified Interoperability Framework from (Chen and Doumeingts 2003)

| *Levels/barriers* | *conceptual* | *technological* | *organizational* |
|---|---|---|---|
| *Business* | Semantic incompatibilities | ITC incompatibilities | Org. structure incompatibilies |
| *Process* | Semantic and syntactical incompatibilities | | Incompatibilities of authorities and responsibilities |
| *Service* | | | |
| *Data* | Semantic incompatibilities | | Incompatibilities of responsibilities |

(b) Enterprise interoperability framework from INTEROP (Kosanke 2006)




(c) Enterprise Interoperability Framework (with three dimensions) from INTEROP (Chen and Daclin 2006; Chen, Doumeingts et al. 2008)

**Figure 1-2. Enterprise Interoperability Frameworks**

## III. FEDERATION

As our work is more about federation, this thesis will discuss more about it. In (ISO-14258 1998), three interoperability approaches: integrated, unified and federated are defined, and then they are adopted into the interoperability framework Figure 1-2 (c). However, the three approaches are defined conceptually as their definitions are based on models of exchanged information (model, meta-model, no model/meta-model). Their definitions are not practical in a real project, so (Berre, Hahn et al. 2004) defines them as follows and distinguishes them from the viewpoint of system architecture topology in Figure 1-3:

- Integrated approaches ensure interoperability by using shared execution environments and shared communication conventions.
- Unified approaches ensure interoperability by using shared meta-models and concepts and shared specification environments;
- Federated approaches establish and maintain collaboration between autonomous local services, each of which runs a local business process.

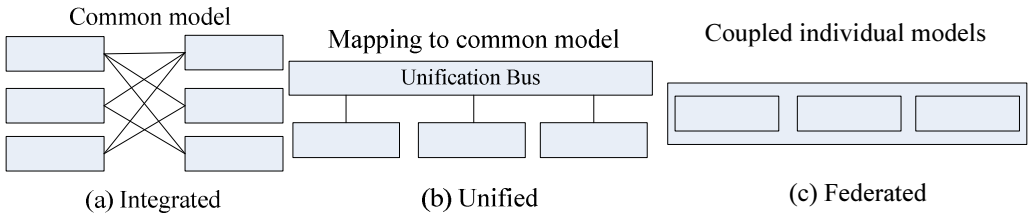



**Figure 1-3. The Integrated, Unified and Federated Approaches (Berre, Hahn et al. 2004)**

In this thesis, we propose five criteria in Table 1-2 to distinguish the three kinds of approaches. In Table 1-2, "problem scope" describes whether the scope of a problem to be treated is fixed or not. "Adaptability to changes" describes how a new system will influence original systems when it is added. "Result" describes after the integrated, unified or federated approaches are applied, which kind of system will be finally generated from original systems. "Connector" means how to connect two different collaborators/participants. "Translator" means how to do translation between different collaborators/participants as different collaborators/participants may use different models to describe their business information.

In Table 1-2, for the approaches from integrated to federated, the boundary of problem scopes becomes increasingly ambiguous. The result from the three kinds of approaches become from a monolithic system to an autonomous system. The three kinds of approaches become increasingly adaptable to changes.

In Table 1-2, as the integrated approaches use common model (same vocabulary) between collaborators/participants, hence they need no translator but they should construct connectors (a kind of software components) to establish connections between them. Connectors in integrated approaches are **technique-specific** and **vendor-specific**. Besides, information delivered by connectors is all based on common model.

In unified approaches, collaborators/participants have their own information models (different vocabularies) and their collaboration is supported by mappings from individual models to a common model. Since mappings from individual models to common model or vice versa can be done by a unification bus, collaborators/participants are not required to definitely know the common model. Therefore, one collaboration activity between two collaborators/ participants will need two connectors and two translators (see Figure 1-3). However, in unified approaches, it is not necessary to make the common model and individual models have the same meta-model[11]. This is in conflict with the definitions of unified approaches in (ISO-14258 1998) and (Berre, Hahn et al. 2004). In fact, meta-model in

[11] Meta-model means the model of a modeling language in MDA (see in Chapter 2-Section II.1).

definitions of unified approaches in (ISO-14258 1998) and (Berre, Hahn et al. 2004) is more concerned with common model, not with meta-model defined in MDA. Besides, in unified approaches, all systems to be unified must be registered in a unification bus to facilitate system management or governance. The registration information can be regarded as **logic connections** between the unification bus and collaborators/participants.

In federated approaches, connections between collaborators/participants are supported by standard transport protocols in their environment, so connectivity are no longer the focus, and connections are established when needed. Relevant connectors are technique-independent and protocol-specific. In federated systems, all constituent systems are autonomous and they can freely join or leave from the federated systems, so it is unreasonable to ask all constituent systems to use the same common model. As different collaborators/participants use different models (vocabularies), so each collaborator/participant must have a translator. In federated approaches, if N collaborators/participants want to collaborate directly with others, there are at most N*(N-1)/2 translators[12] between them.

According to Table 1-2, integrated and unified approaches are focused more on syntactic and technical problems; instead, federated approaches are focused more on semantic problems. In fact, there is no clear boundary between integrated and unified approaches or between unified and federated approaches. These approaches are proposed in different evolutionary phases of information systems in order to resolve interoperability problems. In some cases, they even share some common points. For example, connectors in integrated approaches can also be used in unified approaches to establish connections between collaborators/participants. However, this makes the whole system tightly coupled. Some traditional middlewares or EAI are examples under such situation (Bernstein 1996; IONA 2006). Since SOA and web service came out, the unification buses become more and more service-oriented, such as the Enterprise Service Bus (ESB) Celtix[13] , Petals[14] , Mule[15] or the grid computing platform Globus[16]. SOA/Web service makes the whole unification system loosely coupled. Under this situation, connectors between the unification bus and collaborators/participants can be based on standard protocols, so they are **protocol-specific**

[12] If collaborators/participants don't share a translator between each other, then at most there are N*(N-1) translators between them.

[13] http://celtix.ow2.org/

[14] http://petals.ow2.org/

[15] http://www.mulesoft.org/

[16] http://www.globus.org/

and **technique-independent**, and they do not need to be constructed specially for collaborators/ participants. Currently, some ESB have the above two kinds of connectors at the same time, such as Petals. The second kind of connectors (protocol-specific) can also be used in federated approaches because systems to be federated must be autonomous and loosely coupled. For example, (Baude, Filali et al. 2010) has proposed an ESB federation architecture for large-scale SOA. The work of Baude and his colleagues is at the conceptual level and it does not discuss in detail how to use semantic technologies to resolve semantic problems. In this thesis, we will discuss ESB federation in Chapter 5.

**Table 1-2. Difference between Integrated, Unified and Federated Approaches**

| | | **Integrated** | **Unified** | **Federated** |
|---|---|---|---|---|
| **Problem Scope** | | Scope is fixed (Systems to be integrated have been determined before) | Scope is manageable (all systems must be registered in a unification bus) | Scope is not fixed and not manageable. (There is no central manager) |
| **Adaptability to changes** | | If a new system is integrated, then all other systems will be modified. | any new system to be unified needs to be registered in a unification bus; no other systems will be influenced | If a new system is added, it can make other systems know itself gradually |
| **Result** | | Generate a monolithic system | Generate a loosely coupled system (all unified systems are managed but not controlled by unification bus) | Generate an autonomous system (all constituent systems are autonomous) |
| **Connector** | **Collaboration between two collaborators** | One connector (technique-specific) | Two connectors | No technique-specific connector |
| | **Collaboration between N collaborators** | At most N(N-1)/2 technique-specific connectors | N connectors | No technique-specific connector |
| **Translator** | **Collaboration between two collaborators** | No translator | Two translators | One translator |
| | **Collaboration between N collaborators** | No translator | N translators | At most N(N-1)/2 translators |

## IV. MATURITY MODELS FOR ENTERPRISE INTEROPERABILITY

Besides the above interoperability frameworks, there are some other interoperability frameworks which have been resumed in (Chen, Doumeingts et al. 2008). When applying enterprise interoperability frameworks to solving associated problems, it is better to evaluate *the extent to which interoperability can be achieved*. That is to say, interoperability is not a

level (degree) as defined in compatibility levels of (IEC-TC65/290/DC 2002), but it is a spectrum including several levels (degrees). There are many terms to describe such extent (degree), such as, stages of growth, measurement model, maturity model, maturity levels and reference model, etc. This thesis prefers using **maturity model** to measure interoperability. Many maturity models have been proposed during evolution of interoperability. (Ford, Colombi et al. 2007) has summarized 14 interoperability measurement (maturity) models and it has also analyzed their types, strengths and weakness. Unfortunately Ford and his colleagues have not identified mappings of maturity levels between different maturity models. But two of the 14 maturity models have been aligned in (Tolk and Muguira 2003). This chapter will discuss and align another two maturity models: interoperability maturity model in e-government (Gottschalk 2009) and LCIM (Level of Conceptual Interoperability Model) (Wang, Tolk et al. 2009). At last, this chapter will discuss the relationship between maturity model and our work.

When researching interoperability in digital government, Petter Gottschalk defined five maturity levels for interoperability (Gottschalk 2009):

1) *Computer interoperability*: based on physical connectivity and communication, different systems can directly exchange messages and meaningful, context-driven data;
2) *Process interoperability*: it aligns processes (sub-processes, complete processes and sets of processes) in inter-operating organizations. Semantic interoperability must be examined and resolved at computer and process interoperability levels;
3) *Knowledge interoperability*: knowledge about interoperating organizations is collected and stored together by following a flow strategy;
4) *Value interoperability*: it is concerned with interactions between primary activities in different value configurations (e.g., value chains, value shops and value networks) present in different interoperating organizations;
5) *Goal interoperability*: synergy among interoperating organizations is important and there are no conflicting goals.

Amongst the five maturity levels, goal interoperability is the highest and computer interoperability is the lowest. The five levels are listed in Table 1-3.

LCIM (Level of Conceptual Interoperability Model) is another maturity model of interoperability. It is originally proposed in (Tolk and Muguira 2003) and then it evolves to a more mature model which is illustrated in (Wang, Tolk et al. 2009). LCIM has seven levels:

L0-No Interoperability, L1-Technical Interoperability (defining bits and bytes), L2-Syntactic Interoperability (defining structured data), L3-Semantic Interoperability (defining meaning of data), L4-Pragmatic Interoperability (defining use of data), L5-Dynamic Interoperability (defining effect of data), and L6-Conceptual Interoperability (defining assumption, constraints, etc). LCIM concentrates on and is limited to information (data) exchange. It can be regarded as a maturity model to evaluate the interoperability and composability of existing systems; in the maturity model, higher levels of LCIM will not be achieved until lower levels are all satisfied (L6 is higher than L0). LCIM can also be used as a guidance model to prescribe and guide the interoperability and composability design and implementation of future systems. The above seven levels are listed in Table 1-3.

Table 1-3 also lists the barriers proposed in the interoperability framework Figure 1-2 (c) and establishes the mappings to levels of the above two maturity models. In fact, achieving interoperability means removing barriers. The mapping between barriers and the two maturity models is not precise and the mapping can also not be precise because, different maturity models are established from different viewpoints of interoperability. The maturity model of (Gottschalk 2009) can be regarded as proposed according to organization management (goal, value, knowledge, process and computer); instead, the maturity model of (Wang, Tolk et al. 2009) can be regarded as proposed in terms of linguistics (syntactic, semantic and pragmatic). Two examples for the imprecise mappings are as follows:

- process interoperability in (Gottschalk 2009) will deal with conceptual and technical barriers and it is also relative to L2-syntactic, L3-semantic, L4-pragmatic and L5-dynamic in (Wang, Tolk et al. 2009).
- L2-syntactic and L3-semantic in (Wang, Tolk et al. 2009) are related with conceptual and technical barriers and they are also associated with computer interoperability in (Gottschalk 2009).

Generally speaking, when collaboration/interoperability between enterprises is supported by IT[17], a higher maturity level of interoperability must be based on the achievement of all lower levels. If a certain maturity level of interoperability is achieved, the corresponding barriers will be eliminated by following interoperability approaches (see Figure 1-2 (c)). Our work is interested in the maturity level "Process Interoperability" in Table 1-3. However, our work is not limited to this level and it will also study problems at "Computer

[17] Interoperability between enterprises may be supported only by humans, not by IT. Then conceptual or organizational barriers will be coped with only by humans.

Interoperability" level. As "Goal Interoperability" is very important to collaborative enterprises, this thesis will introduce the concept "goal" into lower levels: process and computer interoperability levels.

**Table 1-3. Approximate Mapping between Two Maturity Models of Interoperability**

<table>
<tr><th colspan="2">Sources<br>Maturity levels<br>Barriers</th><th>(Gottschalk 2009)</th><th>(Wang, Tolk et al. 2009)</th></tr>
<tr><td rowspan="3">Organizational Barriers</td><td rowspan="3">Different definitions about responsibility, authorization, organizational structures and organizational goals, etc.</td><td>Goal Interoperability</td><td rowspan="3"></td></tr>
<tr><td>Value interoperability</td></tr>
<tr><td rowspan="5">Knowledge Interoperability</td></tr>
<tr><td rowspan="6">Conceptual Barriers</td><td>Different terminologies or dictionaries; different description of assumptions or constraints in business, etc.</td><td>L6-Conceptual Interoperability</td></tr>
<tr><td rowspan="2">Modeled in different languages; executed in different methods (in series or parallel)</td><td rowspan="3">Process Interoperability</td><td>L5-Dynamic Interoperability</td></tr>
<tr><td>L4-Pragmatic Interoperability</td></tr>
<tr><td rowspan="2">Different terminology definitions; different representation methods for semantics</td><td rowspan="2">L3-Semantic Interoperability</td></tr>
<tr><td rowspan="2">Computer interoperability</td></tr>
<tr><td>Data may be structured in different methods and in different modeling languages.</td><td>L2-Syntactic Interoperability</td></tr>
<tr><td rowspan="2">Technical Barriers</td><td>Incompatibility of programming languages, coding formats (e.g., UTF-8, ISO-8859-1), platforms /infrastructures, etc; Incompatibility of operating systems; Incompatibility of network devices and network protocols, etc</td><td>L1-Technical Interoperability</td></tr>
<tr><td>No connectivity device</td><td></td><td>L0-No Interoperability</td></tr>
</table>

## V. CONCLUSIONS

To resolve enterprise interoperability problems, the most important thing is to identify technical, conceptual and organizational barriers (or heterogeneity) and then to align the heterogeneity to achieve a certain maturity level of interoperability. Figure 1-4 positions the

barriers (heterogeneity) in an enterprise environment. An enterprise environment has business environment and IT environment[18]. Conceptual barriers come from business environment, and they are caused by different understanding and representation to the same world from different humans/enterprises. Technical barriers stem from IT environment. Some technical barriers are not related with business, such as incompatibility of different operating systems or communication protocols. Some technical barriers are related with business, such as incompatibility of different implementation of business concepts, and they are situated in "IT solutions". IT solutions occupy the overlap between business and IT environments and they support business activities with the help of IT software and hardware. In fact, IT solutions confront the barriers from IT and business environments. That is why some papers, such as (Tolk and Muguira 2003; Wang, Tolk et al. 2009), cannot evade the discussion about technical interoperability when talking about conceptual interoperability. Finally, organizational barriers originate from enterprise environment, such as difference of organization structures. Some of organizational barriers are related with business, such as different authorization [19] of employees, so they are located in business environment. Organizational barriers will influence all collaboration/ interoperability activities in enterprise environment.

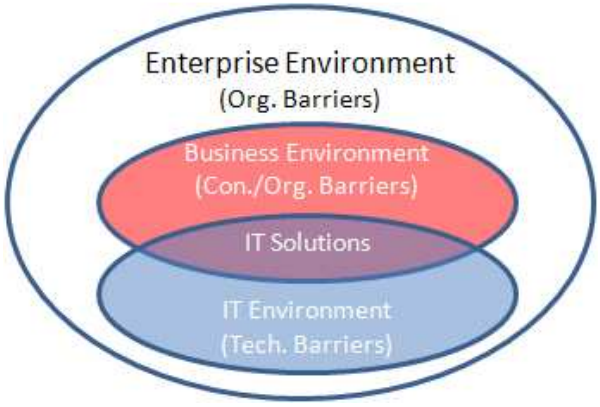



**Figure 1-4. Three Kinds of Barriers in Enterprise Environment**

Our work is to find an IT solution to enterprise interoperability problem. The IT solution will align business and IT environments to support collaboration between different enterprises. In order to construct our IT solution, this thesis will study related research

[18] Besides business and IT environments, an enterprise also has social, economic and legal environments. They are not close to my work, so they are not in Fig. 1-3.

[19] For example, some employees are authorized to launch a business process, but others may not. Different enterprises may have different authorization.

domains: collaborative business process, SOA, MDA, ontology and ESB. After their study, this chapter will conclude the landscape for IT solution to enterprise interoperability problems.

# *CHAPTER 2: State-of-the-Art for the Research Domains Related With Enterprise Interoperability*

Although (Chen and Doumeingts 2003) have proposed three main research domains about interoperability, but it also points out that the three main research domains are lack of integration and one critical task is to develop an integrated view and approach that link three domains together to find interoperability solutions. This chapter will provide an answer to the above problem. It will study the research domains related with enterprise interoperability: collaborative business process, MDA/MDI, SOA, ESB and ontology, and it will position the roles of the research domains in a solution to enterprise interoperability problems.

# I. BUSINESS PROCESS AND COLLABORATIVE BP TOOLS

In order to resolve business collaboration and business agility, collaborative business process (CBP) is widely studied in the scientific domain and industrial domain. Collaborative business process aims to define business collaboration requirements, not only between different enterprises but also between different departments of one enterprise. Evidently CBP includes inter- and intra-organizational workflows.

For business collaboration and agility, Service Oriented Architecture (SOA) is recognized as the leading architectural approach and it also facilitates technical agility and interoperability between information systems (IS) of enterprises (OMG 2008). So our researches focus on collaborative business process in the SOA environment.

So far, there have been many languages for modeling collaborative business process, for example, WSFL, XLANG, ebXML BPSS, WPDL, XPDL, BPMN, WS-CDL, BPDM and so on. In order to model and execute the CBP expressed in the above languages, there are also some CBP tools such as BizAgi, jBPM, BONITA, Oracle, MEGA and so on. This chapter will compare them. The comparison work is the base for our further study.

Section I.1 will provide a brief history of business process and it will also resume the research result of comparison between different collaborative business process languages. In Section I.2, we will propose a comparison framework. Section I.3 will list the CBP tools to be compared and Section I.4 will give the comparison result. At last, Section I.5 concludes this section.

## I.1. Literature study of Collaborative Business Process

### I.1.a. Emergence of Collaborative Business Process

In order to discuss the appearance of CBP, we will discuss the brief history of business processes. This thesis divides the history of business processes into four phases:

1) The first phase is before 1970's - the advent of information systems in computer science domain (Avgerou 2000). Business processes were written in papers and performed by human beings.
2) The second phase is from 1970's to 1990's. In this phase, before 1980's, all information and control flows about business process were hard-coded in applications; and then in late 1980's and early 1990's, business processes could be expressed by flexible scripts (Dayal, Hsu et al. 2001). In this phase, if business processes were not able to be realized by IT applications, most enterprises wrote them down into policy and procedure manuals, which were as hard to modify as business processes encoded in applications (Bauer, Roser et al. 2005). In general, implementation of business processes in this phase has improved management efficiency and productivity of companies. However, business processes were not observable or visible in information systems and they were not easy to monitor by managers in companies.
3) The third phase is from the early to mid 1990's. During this phase, business processes were realized and managed by workflow management systems (WMS), which were used to integrate applications, data and procedures. In this phase, there was another important event – business process reengineering, which was promoted by Hammer (Hammer 1990) and Davenport (Davenport 1993) to improve customer service and cut operational costs. However, WMS did not support simulation, verification and validation (V&V), analysis and optimization of business processes (van der Aalst, ter Hofstede et al. 2003).
4) The fourth phase is up to now. In this phase, the WMSs were relabeled as Business Process Management (BPM) system, suites or platforms (van der Aalst, ter Hofstede et al. 2003) (Bartonitz 2010). The first appearance of BPM on the Internet is from 1997 in an announcement about the annual report of the American firm EDS (Bartonitz 2010). BPM systems (BPMS) can design, implement, simulate, execute, manage and analyze business processes (van der Aalst, ter Hofstede et al. 2003).

The third phase focuses on the intra-organizational business processes; instead, the fourth phase focuses more on the inter-organizational business process. To realize the objective of business processes, numerous languages have been proposed and standardized in the research domain of WMS/BPMS. Most business processes description languages are derived from the traditional programming languages and they are difficult for business

analysts to learn and use (Jenz 2003). It means that there is a wide gap between business domain and IT domain. During the alignment of business and IT, BPMS and SOA go together to realize business agility (Ling and Xin 2009) – companies can be more rapidly adapted to business changes from customers, market or themselves. As web service is de-facto implementation protocol of SOA, nowadays BPMSs are more and more based on web service-based XML execution languages (OMG 2011). However, these languages such as WS-BPEL (Alve and Arkin 2007) are still oriented to IT engineers, not to business people. In order to address the interoperation of business processes at human-level (OMG), BPMN Version 1.1 was created and published out by OMG in 2008. In order to model collaborations between companies, some important concepts such as "conversation" and "choreography" were added into BPMN2.0 released in January 2011.

Section I.1.b will discuss some languages for business processes, especially for collaborative business processes. And it will also give some existent comparison results between the languages. Section I.1.c will introduce BPMN in detail and our study about enterprise interoperability will be based on BPMN.

### I.1.b. Comparison between Specification Languages of Collaborative Business Processes

In order to describe and formalize the collaborative business processes, many enterprises have proposed their own specification languages, such as Microsoft, IBM, SAP, etc. Some specification languages have been delivered to standard organizations, such as OASIS, W3C, OMG, WfMC and so on and have been published as standards. In November 2009, (Bartonitz 2010) described the relationship between some specification languages and the related organizations. (Bartonitz 2010) also illustrated the development of the past decade for these languages. This section extends the result of (Bartonitz 2010) in Figure 2-1.

In 2003, (Bernauer, Kramler et al. 2003) compared seven specification languages/approaches (WSDL, WSFL, ebXML, BPML, XLANG, BPEL, WSCL, WSCI and WPDL) according to a framework of requirements which has the seven perspectives: functional perspective, operational perspective, behavioral perspective, informational perspective, interaction perspective, organizational perspective and transactional perspective. Each perspective defines detailed criteria, for example, interaction perspective defines three criteria: interaction primitives, interaction implementation and interaction independence. The comparison shows that none of seven languages fulfill all requirements of the framework. And (Bernauer, Kramler et al. 2003) proposed two methods to solve the above problem:

extending some language to address all of the requirements or combining several languages together. It also pointed out that Model Driven Architecture (MDA) will be a research direction about CBP for generating processes automatically.

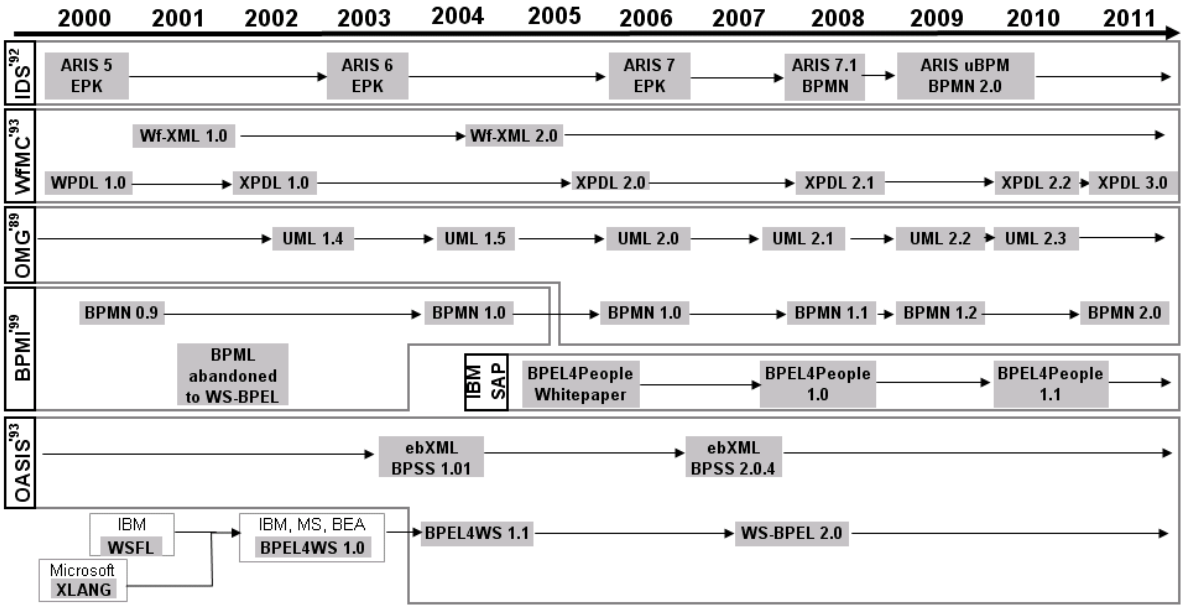



**Figure 2-1. Historical development of technical standards in CBP (adapted from (Bartonitz 2010))**

Figure 2-1 also implies that, over time, some specification languages are merged together, some are replaced with new languages and some are created for the first time, for example XLANG and WSFL were merged to WS-BPEL, WPDL was replaced officially with XPDL v1.0 by the WfMC in 2002, and BPDM came out in 2003 and was finalized by OMG in 2008. So in 2005, (Roser and Bauer 2005) compared and classified eight languages/techniques (ARIS, BPDM, BPML, BPMN, ebXML-BPSS, WS-BPEL, WS-CDL and J2EE) from five perspectives: MDA's abstraction level (CIM, PIM, PSM), modeling of business processes, notation, standardization and tool-support. Their research surrounded a methodology for developing collaborative business processes from CIM to PIM and to PSM, i.e., integrated collaborative business process languages into software engineering process.

### I.1.c. BPMN

There are many languages for CBP and each language has its own supporting tool set. However, we will focus on the CBP tools in BPMN. Why do we focus on BPMN? In Section I.1.a, the evolutionary history of business processes has offered some reasons. Besides, BPMN itself has some advantages. BPMN (OMG 2011) is a graphical notation whose aim is

to model enterprise processes. It was developed by Business Process Management Initiative (now a part of the OMG).

BPMN can be used to capture the business processes that can be shared between the stakeholders. BPMN is very expressive and provides a notation that is intuitive to business users. The latest BPMN specification (Version 2.0) adds enhancements to BPMN so that execution engines would be able to interpret and execute business process models (Buelow, Das et al. 2010). BPMN 2.0 permits business and IT alignment: Process analyst can use BPMN 2.0 for modeling and models can be refined by IT developers so that the process models can be executable. Contrariwise, IT developers can create lower processes which can be combined by business users to support faster-to-market requirements. The enhancements of BPMN 2.0 are as follows:

- BPMN 2.0 includes both diagram interchange as well as model interchange. It provides a standard XML schema for interchanging BPMN models, both executable and non-executable. Indeed, it is the same schema for both. An executable model just has more technical details. (Buelow, Das et al. 2010)
- Non-interrupting events: interrupting Event Sub-Processes and boundary Events interrupt normal execution of their parent activities and after their completion, the parent activities are immediately terminated. However for non-interrupting Events, during execution of a non-interrupting Event Sub-Process, the execution of the parent activity continues as normal. For instance, a timer event permits do specify a task deadline, but if the deadline expires we do not necessarily want to interrupt the task. We can send a reminder to the performer, while letting the task continue (Silver 2009 ).
- As BPMN2.0 defines the formal execution semantics for BPMN elements, hence it can be used to capture process models and to implement models (Buelow, Das et al. 2010).
- BPMN2.0 extends the definition of human interaction and aligns BPEL4people with itself(Buelow, Das et al. 2010).
- BPMN2.0 defines new diagrams: Conversation Diagram, Choreography Diagram.

### I.2. Comparison Framework

There are many views about BPM lifecycle. For example, (van der Aalst, ter Hofstede et al. 2003; Brahmandam 2008) has proposed their own BPM lifecycle. In (van der Aalst, ter Hofstede et al. 2003), a BPM lifecycle has four phases: process design ((re)modeling &

simulation) [20] , system configuration (implementation), process enactment (execution), diagnosis (monitoring and analysis). In (Brahmandam 2008), business process lifecycle contains six phases: business process discovery (definition & modeling), business process analysis and modeling (modeling & implementation), business process simulation, business process development (implementation), business process execution, business process monitoring and optimization (monitoring, analysis & definition).

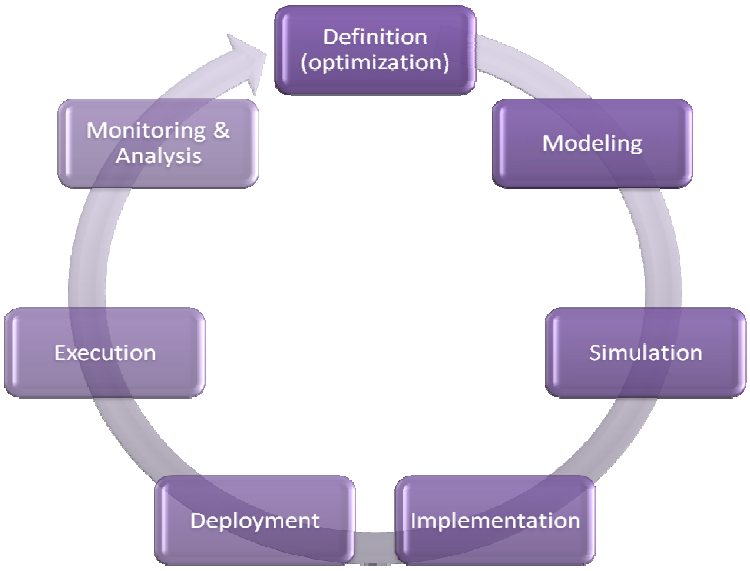



**Figure 2-2. BPM lifecycle**

This thesis prefers the BPM lifecycle depicted in Figure 2-2. The preferred BPM lifecycle includes seven phases: "definition", "modeling", "simulation", "implementation", "deployment", "execution", "monitoring and analysis" and at last back to "definition (with optimization)". In the "Definition" phase, requirements for business processes are defined, including optimization requirements. In the "Modeling" phase, business processes are modeled in a process language. In the "Simulation" phase, business processes are configured with simulation parameters, such as expected time taken by a task, and then they are simulated to find out process bottlenecks and performance. In the "Implementation" phase, all necessary information for execution is integrated into business processes, such as data models, business rules, web services, etc. In this phase, business processes will be represented in an executable process language. In the "Deployment" phase, executable business processes are deployed into a software infrastructure/platform. In the "Execution" phase, the infrastructure/platform executes deployed business processes. In the "Monitoring & Analysis" phase, the infrastructure/platform will monitor business process execution and analyze

[20] The explanation in parenthesis is the corresponding phases in Figure 2-2.

monitoring information to generate corresponding analysis reports. The reports will be used to optimize business processes in the next life cycle.

When comparing CBP tools, this thesis is concerned about the phases “modeling”, “simulation”, “implementation”, “deployment”, “execution” and “monitoring & analysis”. For each phase, some related criteria will be added. As some CBP tools (e.g., BizAgi Studio, Bonita Open Solution) combine together the modeling phase and the implementation phase, hence in the comparison framework, we combine them together.

### I.2.a. Modeling & Implementation

**Coverage of BPMN**. In BPMN v1.x, it defines four basic categories of elements: flow objects (events, activities, and gateways), connecting objects (sequence flow, message flow, and association), swimlanes (pools, lanes) and artifacts (data objects, group, and annotation). And in BPMN 2.0, it adds new elements and attributes, for example, choreography, collaboration and conversation. So to what extent the BPMN tools support these elements and attributes is a problem.

**Persistence**. Now, CBP can be stored in XPDL, BPDM, BPEL and the BPMN 2.0 XML format. The CBP files can be managed by file systems (FS) of operating systems (OS) or by database management systems (DBMS). The CBP files in JPG, DOC or HTML will be ignored in this thesis.

**Types of CBP**. In this thesis, CBPs are more about **coordination business processes** (**CrBP**) and **cooperation business processes** (**CpBP**) defined in Chapter 3-Section I. A coordination business process is composed of the activities some of which take place between enterprises, but the process execution is owned and controlled by only one enterprise. Figure 2-3 shows an example of CrBP, in which Enterprise A uses services provided by Enterprises B and C.

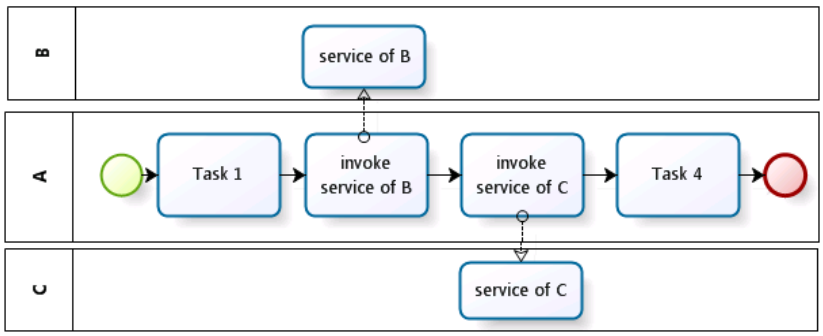



**Figure 2-3. Example of Coordination Business Process**

A cooperation business process is composed of the activities some of which take place between enterprises. The process execution is owned and controlled by enterprises, but each enterprise can only control the execution of its own activities. There are two expression views for CpBP: centralized view and distributed view. For centralized view, the activities of all enterprises in CpBP are modeled in one business process. For example, in Figure 2-4, the activities of enterprises A and B are in the same process. This view was proposed and used by (Qiming and Meichun 2001) and (Liu and Bourey 2010). For the distributed view, the activities of each enterprise are included in its own business process, and the collaboration between enterprises is expressed by their message exchange. For example, in Figure 2-4 the activities of A or B are included in their own processes, and their collaboration is the message exchange. This view has been studied by (van der Aalst and Weske 2001) and (Chebbi, Dustdar et al. 2006).

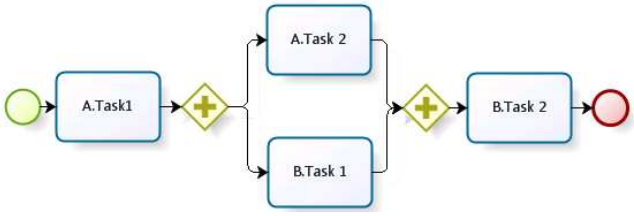



**Figure 2-4. Example for centralized view of Cooperation Business Process**

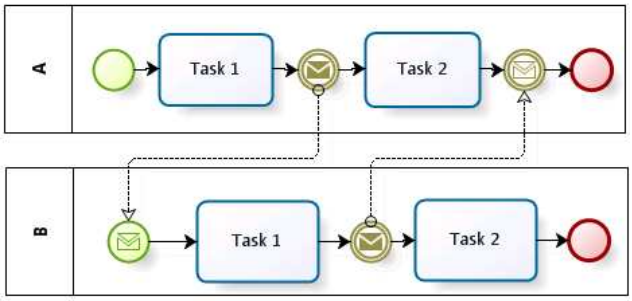



**Figure 2-5. Example for distributed view of Cooperation Business Process**

**Human Tasks**. In a collaborative business process, not all tasks can be performed automatically by an IS. Some of them may be related with human beings and even only manipulated by human beings, and they are called human tasks, for example, a task may ask a person to input some information into IS, or a task may ask a telephone technician to install a telephone at a customer location. So a CBP must model human tasks. It can provide a graphical user interface (GUI) for human beings to input some data into IS, or it can send out a notification (for example an e-mail) to a designated person.

**Business Rules**. Business Rules are necessary in CBP, especially for their gateways and script tasks. During the modeling phase, formal or informal business rules must be provided in CBP. More information about business rules can be gotten in Appendix A.

**Web Services**. As de facto implementation of SOA, web service can be used as one kind of implementations for service tasks in BPMN2.0.

#### I.2.b. Simulation

In CBP, some objects or attributes can be predefined for flow objects, such as the instance number of CBP, the execution duration of a task or an input object for a "start" event. After these configurations, the CBP can be simulated. The simulation process can be started, stopped, paused and restarted. The simulation parameters can be stored in file systems or databases.

#### I.2.c. Deployment

**Deployment Style**. If BPMN modeler can provide a "deploy" button and when the button is clicked, the CBP is deployed on process engines automatically, this is **automatic deployment** method. The automatic deployment method is usually supported by a database. If the constructed CBP and relevant configuration files must be copied manually onto process engines, this is the **manual deployment** method.

**Deployment Topology**. If a CBP is deployed only on one server which can be managed by one of collaborators (organizations participating in CBP) or by a neutral third party, such deployment is the **centralized deployment**. CrBP and CpBP can be deployed centrally.

If a CBP is deployed on servers of all collaborators (enterprises), such deployment is the distributed deployment. Only CpBP can perform the **distributed deployment**. There are two modes for distributed deployment corresponding to two views of CpBP: **uniform mode** and **discrete mode**.

- If a CBP is under the centralized view, all collaborators' servers will deploy the same copies of CBP, and this situation is defined as **uniform mode** of distributed deployment. For example, in Figure 2-4, on the servers of Enterprises A and B, there will be a copy of the whole CBP. Under uniform mode, each collaborator can have a global view of their collaboration.
- If a CBP is under the distributed view, each collaborator's server will deploy part of the CBP which belongs to the collaborator. This situation is defined as **discrete mode** of distributed deployment. For example, in Figure 2-5, the process in the rectangle A will

be deployed on Server A and the process in the rectangle B will be deployed on Server B. Under discrete mode, each collaborator can only have its local (partial) view of their collaboration.

#### I.2.d. Execution

**Coverage of BPMN**. Not all the process engines execute all of the elements and attributes defined in BPMN 2.0.

**Persistence**. During the execution of process engines, runtime states and history information of CBP instances and warning or error information can be made persistent into files managed by file systems of OS or by DBMS. So if any failure occurs in process engines, the CBP instances can be restored when process engines are restarted and the administrators of process engines can follow logs to locate problems and find corresponding solutions.

**Transaction**. Transaction in business process is derived from that in database management systems. There are three kinds of transaction models: traditional ACID (**A**tomicity, **C**onsistency, **I**solation, **D**urability), extended transaction models (Sheth, Rusinkiewicz et al. 1992; Shet and Rusinkiewicz 1993) for long-running workflows and interoperable transaction model (Weigand and Ngu 1998).

- According to (Dayal, Hsu et al. 2001) and (Bernauer, Kramler et al. 2003), traditional ACID is not suitable to treat an entire business process as a transaction;
- Extended transaction models are suitable for intra-organizational workflows;
- Only interoperable transaction model is a suitable technique for CBP.

During the execution of an interoperable transaction, steps have to be rolled back, or compensated, and sometimes different alternatives must be tried and negotiated to fulfill the given task instead of aborting the whole transaction. For example, BPMN provides Transaction sub-processes. A transaction has three outcomes (OMG 2011): (1) successful completion, (2) failed completion: the activities inside the transaction will be subjected to the cancellation actions (rollback or compensation), (3) hazard: something went terribly wrong and that a normal success or cancel is not possible (i.e. no rollback nor compensation).

**Execution Modes of CBP**. The execution mode of CBP depends on its deployment topology. If a CBP is under centralized deployment, then the execution mode of CBP will be **centralized mode**, i.e., there is only one execution engine who executes the CBP, especially if a CBP is a CpBP, then the execution of each task is only controlled by its owner, i.e., no enterprise can control the execution of the activities that do not belong to it.

If a CBP is under distributed deployment, then the execution mode of CBP will be **distributed mode**.

- Under the distributed execution mode, if a CBP is under centralized view, then the CBP will be under uniform mode of distributed deployment and the server of each collaborator (enterprise) in CBP will execute the whole CBP, but if it encompasses the activities that do not belong to it, it will just wait notifications from the activities' owners and then skip them.
- Under the distributed execution mode, if a CBP is under distributed view, then the CBP will be under discrete mode of distributed deployment and the server of each collaborator will execute its local part in CBP and the servers will be responsible for message exchange between collaborators.

**Human Task**. When a process execution engine comes across a human task in CBP, it parses and executes the associated code or scripts, and then it interacts with people.

**Business Rules**. When a gateway in CBP is executed, related business rules must be parsed and executed.

#### I.2.e. Monitoring & Analysis

The persistence of the monitoring information (for example, runtime states of CBP instances, events during the execution of CBP) is discussed in Section I.2.d "Persistence", so this section will discuss the visualization of monitoring.

During execution of a CBP, the CBP management system can monitor the execution progress of CBP visually by providing a GUI, for example a webpage or a dynamic picture. In order to make monitoring information more meaningful, the CBP management system can generate statistic reports after analyzing the monitoring information.

### I.3. Comparison of Collaborative Business Process Tools

According to the comparison framework defined in Section I.2, this section will compare six CBP tools: BizAgi Xpress, jBPM, Bonita, Oracle BPM, ADONIS and MEGA. Their graphical user interfaces (GUI) can be seen in Appendix C. The comparison result is concluded in Section I.4, which also provides the advantages and disadvantages of these tools.

#### I.3.a. BizAgi Xpress

BizAgi is one of the BPM Solutions, and it can model, execute and improve business processes through a graphic environment and without the need of programming. BizAgi is

available in multiple editions to support the varying needs of organizations. This section focuses on the Xpress edition, which consists of three main modules: Process Modeler, BizAgi Studio and BizAgi BPM Server.

Process Modeler is used to draw process flowcharts in BPMN 2.0. It supports most of the elements and attributes defined in BPMN 2.0 but it does not support "conversation", and "choreography", etc. It can construct CrBP and two views of CpBP. The processes can be managed by file systems of OS.

BizAgi Studio is used to implement modules: input all necessary information for process execution: required data types, web forms for human tasks, business rules, actors of activities, invocation information of web services (see Figure 2-6). After that, business processes are deployed automatically on the target BPM Server (under centralized deployment), and in fact, business processes are stored in a database (ENIX 2006), and then used at runtime for process execution by BizAgi BPM Server.

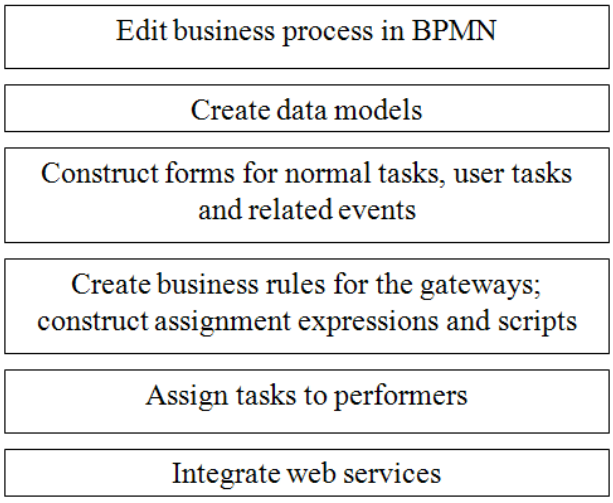



**Figure 2-6. BizAgi Method for Automatic Execution of BPMN Processes**

The BizAgi BPM Server directly executes BPMN processes stored in a database and provides a work portal for end users. When it executes human tasks in business processes, it will show out web pages for predefined actors or send out a notification to a designated person. The execution mode is centralized. All running states and historical logs of business processes are stored in a database.

The work portal provided by BizAgi BPM Server can generate a report to indicate which tasks or processes are on-time, overdue or at risk. It can also generate statistical reports to help business managers to find out bottlenecks, resource performance in business processes (BizAgi 2009).

### I.3.b. jBPM

jBPM is an open-source BPM project from JBoss Community. It is based on BPMN2.0.and it bridges business analysts and developers. This section focuses on the version jBPM5.1.0.

jBPM5.1 provides three options to model business processes: two Eclipse plug-ins and one web-based editor. One of the three modelers (BPMN Visual Editor) can support almost all of the BPMN2.0 constructs and attributes, including collaboration, conversation, choreography, etc. This is quite different from the other CBP tools. The created processes are stored in the BPMN 2.0 XML format and they can be managed by file systems of OS or by DBMS. jBPM prefers storing/deploying business processes into a knowledge database "Guvnor"; hence, the deployment of processes is centralized and automated. So far, none of the three modelers can simulate processes.

The process engine of jBPM5.1 can support a significant subset of elements and attributes defined in BPMN2.0, which have been listed in (JBPMCommunity 2011). However, the process engine does not support "message flow", so it cannot execute the distributed view of CpBP (collaboration diagrams in BPMN2.0). During its running, the process engine can store running states and historical information into files on disks or in tables of databases. It can make a task's execution as an atomic transaction. As the process engine can support compensation events, so it can support the interoperable transaction mode to a certain extent. The process engine supports human tasks during the execution of CBP. It can execute web pages for a human task, and it can also send out emails to designated persons for human tasks. Evidently, the execution mode of jBPM is centralized.

Based on the running states of processes, jBPM can illustrate execution progress of business progresses. According to historical information for execution of processes, jBPM5.1 can generate statistic reports for a process.

### I.3.c. Bonita

Bonita Open Solution is a complete open source BPM Solution. Our work focuses on Bonita Open Solution V5.5, which includes three integrated modules: Bonita Studio, Bonita User Experience and Bonita Execution Engine. Bonita Studio is used to construct business processes in BPMN 2.0. It implements most of important elements and attributes defined in BPMN 2.0 and the implemented elements and attributes are listed in (BonitaSoft 2011). Bonita Studio does not implement "conversation" or "choreography". It can construct CrBP and two views of CpBP. Bonita Studio also provides GUI to create web pages for human

tasks. After creation, a process is stored in a file on disks. Bonita Studio can also simulate a business process after configuring simulation parameters corresponding to processes, events, tasks and transitions, etc. Bonita Studio can make business processes deployed automatically, and the deployment topology is centralized.

Bonita Execution Engine is in charge of executing business processes and it can support all elements and attributed used by Bonita Studio. It can store all running states and history information of business processes into files on disks. It can also execute the webpage codes for human tasks and send out notifications to a designated person. It adopts the centralized mode to execute business processes and especially it can execute centrally the distributed view of CpBP.

Bonita User experience provides a web console for users to view the processes currently executed and the finished processes. It can also generate statistic reports for users.

#### I.3.d. Oracle BPM Suite 11g

Oracle BPM Suite 11g provides an integrated platform for SOA-enabled BPM. It delivers in the same platform a treatment of all lifecycle phases of business processes in an organization (Buelow, Das et al. 2010).

Oracle BPM 11g provides modeling tools that allow the business users to model and manage their business processes. It includes two design tools: JDeveloper-based BPM Studio and web-based Process Composer. Each tool provides different users with a different experience: BPM Studio is targeted at process analysts/architects and developers; Process Composer is targeted at process owners, business users, and business analysts. The two designer tools can support most important elements and attributes defined in BPMN 1.2/2.0.

Oracle BPM Studio enables simulation for a given process by specifying various metrics including cost, unit and time. It can also monitor and analyze interesting business indicators during process execution. Oracle BPM Suite 11g includes many out-of-the-box dashboards to analyze common business indicators such as cycle time, work distribution, work performance, and so on. Oracle BAM (Business Activity Monitoring) is a real-time monitoring product which enables modeling of various aspects of processes and their supporting environments.

Oracle BPM products leverage a Service Component Architecture (SCA) server that provides a unified service and event infrastructure. There are several service engines that provide direct execution for different model types. For example, BPMN service engine executes processes in BPMN 2.0, BPEL service engine executes processes in BPEL, business

rules service engine executes business rules, and SCA server provides optimized binding between these service engines. Moreover all aspects of BPM and related software components can be managed from a web-based console. Actually, in Oracle BPM, created models are not just business requirement documents, but part of their own implementations (Buelow, Das et al. 2010). In addition, Oracle BPM 11g provides three options for automatic deployment: by using JDeveloper, web-based console or using ant scripts.

Finally, in Oracle BPM Human Tasks are managed by the Human Task workflow engine. When the BPMN component triggers a Human Task, the Human Workflow Service is responsible for routing the task to users and notifying them. Once a last user approves or completes the task, the Human Workflow Service returns to resume the corresponding process.

#### I.3.e. ADONIS

ADONIS from BOC[21] is a BPM toolkit for the design of products/services, processes, organizational structures and information technology. ADONIS's philosophy is the continuous improvement of business processes, organizational structures as well as resources and technologies. A successful implementation of ADONIS Model is ensured by open interfaces such as XPDL, BPEL/WSDL, BPMN and XML). The professional edition of ADONIS contains the following components: acquisition, modeling, analysis, import/export, simulation, evaluation and documentation. Unfortunately, the functions "simulation" and "evaluation" are just for business process models (described by another notation language), not for BPMN models.

Graphic representations of processes lean on a reference database and the database stores the graphical elements (process, event, and actor) and their attributes. The attributes are helpful to realize the simulation and evaluation of business processes.

#### I.3.f. MEGA

MEGA[22] is a set of integrated tools (MEGA Suite) used for modeling, controlling, transforming and communicating. We will focus on the modeling tools covering from process analysis to risk and control mapping to application analysis and design, and more especially on the process modeling (MEGA Process). It makes it possible to model processes using BPMN 1.2. MEGA distinguishes four types of processes:

[21] http://www.boc-group.com/

[22] http://www.mega.com/en

a) **business processes** specifying a high-level structural view of the enterprise product and service offerings, and the breakdown of the processes producing them;

b) **organizational processes** describing the sequence of operations executed by enterprise organizational units;

c) **functional processes** describing a summary view, independent of organisational structure, to represent steps in the value chain connected to enterprise business and common to all organizational variants;

d) **system processes** describing the IT system process implemented when using an application or service.

Organizational charts and business data modeling are also available with MEGA processes. One of the main advantages of MEGA is the common multi-user repository making it possible to link each model element with another one and to bridge all the tools of the MEGA Suite. Therefore it is possible to navigate to one modeling element or view to another one. It provides also the users with powerful customisable documentation tools. Although it does not cover the process deployment phase MEGA Suite provides a simulation tool for evaluating the organizational impacts and costs of a process improvement proposal and for calculating the anticipated return on investment (MEGA Simulation BPMN Edition). It also provides a synchronization of MEGA Process models with a SAP implementation.

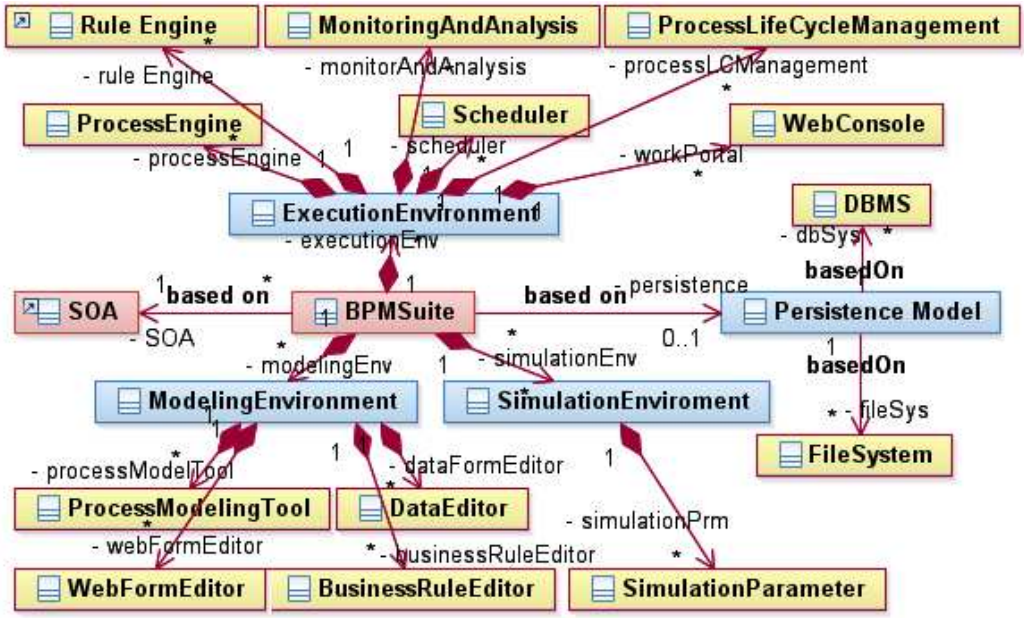



**Figure 2-7. Conceptual Model for CBP Tools**

### I.3.g. Relationship between comparison framework and conceptual model for CBP tools

According to the research about the architectures and implementations of the 6 CBP tools, we can obtain a partial conceptual model for CBP tools in Figure 2-7. In Figure 2-7, BPM Suite contains three environments: modeling, simulation and execution environments. Each environment has its own components. For example, execution environment has process engine(s), rule engine(s), monitor and analysis function, process lifecycle management, scheduler(s) and a web console. The function of each component is indicated by its name. Especially, scheduler is used to schedule execution of multiple business processes/rules. Process life cycle management means starting, suspending, restarting and stopping instances of business processes. This is the micro definition for process lifecycle management, which is different from its macro definition in Section I.2. BPM Suite is based on SOA. That is to say in its environments BPM Suite will use "service" concepts and follow SOA protocols (e.g., web service standards). BPM Suite must also depend on a persistence model in its three environments. For example, constructed processes, simulation parameters or information about monitoring must be stored in databases or in file systems. Besides, Figure 2-7 indicates that the different CBP tools become more and more homogenous but they can be implemented in different technologies.

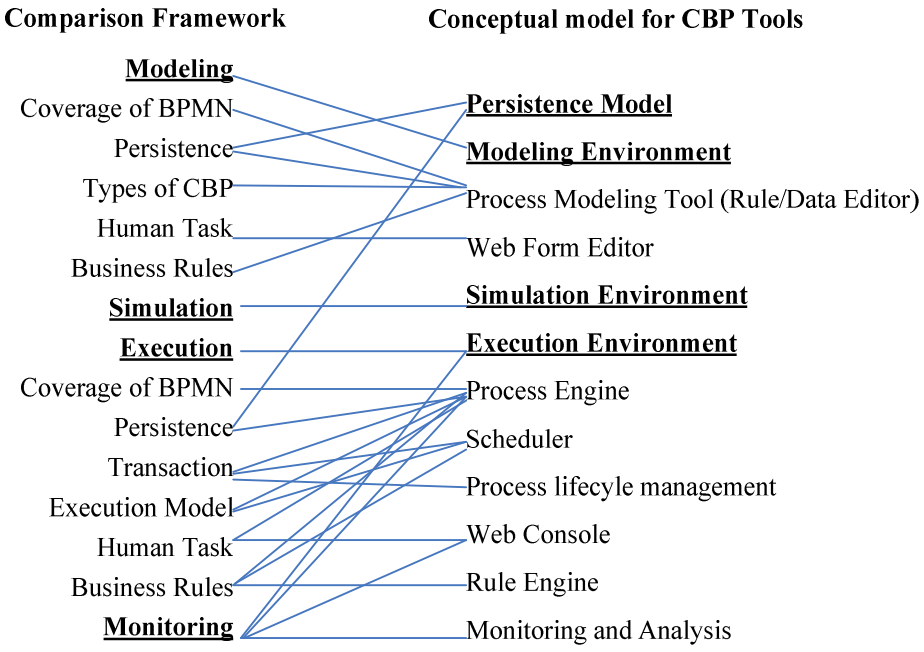



**Figure 2-8. Relation between Comparison Framework and Partial Conceptual Model for CBP Tools**

The corresponding relationship between the conceptual model and the comparison framework is provided in Figure 2-8. For example, the criterion “transaction” in the comparison framework can be related with process engine, scheduler and process lifecycle management in the conceptual model. Besides, process modeling tool in the conceptual model can be evaluated according to the criteria: coverage of BPMN, persistence, types of CBP, human task and business rules. In a word, one criterion in the comparison framework is related with one or several concepts in the conceptual model; one concept in the conceptual model can be evaluated from one or several criteria in the comparison framework.

### I.4. Comparison Result

According to the introduction to six CBP tools in Section I.3, we can summarize their comparison result, shown in Table 2-1. In Table 2-1, the CBP tools can be divided into two kinds: one kind can only model (and simulate) CBP, the other kind can model (and simulate) and also execute and monitor CBP. In the following analysis, we focus on the second kind.

For the criterion of coverage of BPMN, none of these tools can support all elements and attributes defined in any version of BPMN, but they can support the important elements and attributes in BPMN. The execution engines of these tools are almost in the same situation as their CBP modeling tools.

For the persistence criterion, these tools can store CBP in XPDL, BPDM or BPMN 2.0 XML format, which can make CBP easily exchanged between different CBP designers. Some tools, the storage of BPMN processes can be managed by DBMS, which makes CBP manageable and easily shared by other information systems. At the execution phase of CBP, persistence can also be supported by two methods: file systems of OS or DBMS.

Human tasks are supported in two phases: modeling and execution phases. They can be modeled and executed by CBP management systems. This makes BPMN suitable to create and automate human-centric CBP.

For the transaction criterion, BPMN itself can support interoperable transaction model at the modeling phase, but not all of CBP tools can support such model. When executing basic tasks in CBP, some CBP tools can make sure their atomic execution, for example jBPM and Oracle BPM.

Execution mode of CBP depends on deployment topology. All of these tools can only support centralized deployment and execution.

For the monitoring criterion, all of these tools can support to visualize execution progress of CBP instances and they can also generate statistical reports for further analysis of CBP.

Generally speaking, since these CBP tools can only adopt centralized deployment and centralized execution of CBP, in others words CBP can be deployed on and executed by only one (logic) server, hence collaborators in CBP have to select one delegate among them or a trusted third party to run their CBP. This limits the autonomy of collaborators.

In these tools, as CBP is executed on one server, this avoids technical interoperability problem between information systems of collaborators. In such situation, the semantic heterogeneity in business collaborations is resolved at the modeling phase by human being, not automatically tackled by information systems.

Instead, the deployment and execution of CBP under the distributed mode will make collaborators have more autonomy and will also make collaborators more dynamically participate in CBP. How to realize the distributed mode of deployment and execution of CBP will introduce new problems: semantic heterogeneity and negotiation mechanism between collaborators, etc. This will be a hot research point in the future.

## I.5. Conclusions

After analyzing the historical development of CBP's appearance, we summarize the existent CBP specification languages and their comparison result. This chapter focuses on comparing implementation tools of CBP which are based on BPMN. In order to compare CBP tools, we have proposed a framework inspired from CBP lifecycle. After introducing six CBP tools: BizAgi Xpress, jBPM, Bonita, Oracle BPM, ADONIS and MEGA, we have compared them and obtained a comparison result. In terms of the comparison result, we point out that how to solve semantic heterogeneity and interoperability problems during the realization of distributed deployment and execution of CBP will be a hot research point.

In order to solve the above problem, MDA will be a good framework. MDA is introduced in detail in Section II. From the viewpoint of MDA, CBP can be regarded as models, and after some transformation, they can be executable. This can improve enterprise business agility. Besides, as MDA can make CBP models portable, collaborators of one enterprise can also generate corresponding CBP which can help these enterprises collaborate with each other. A possible MDA-based methodology to the problem has been proposed in Chapter 3 and 4.

## Table 2-1. Comparison Result Between CBP Tools

| | Modeling & Implementation | | | | | | Simulation | Deployment | | Execution | | | | | | Monitoring & Analysis |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Coverage of BPMN* | *Persistence* | *Types of CBP* | *Human Tasks* | *Business Rules* | *Web Service* | | *Deployment Style* | *Deployment Topology* | *Coverage of BPMN* | *Persistence* | *Transaction* | *Execution Mode* | *Human Task* | *Business Rules* | |
| BizAgi | BPMN 2.0 (not complete) | Managed by FS of OS or DBMS | CrBP and CpBP | support | XPath | support | not support | automatic | centralized | BPMN 2.0 (not complete) | Managed by DBMS | N/A | centralized (except for distributed view of CpBP) | GUI and notifications | support | visualize execution, generate reports |
| jBPM | BPMN 2.0 (not complete) | Managed by DBMS | CrBP and CpBP | support | XPath MVEL JAVA | support | not support | automatic | centralized | BPMN v2.0 (not complete) | Managed by DBMS | Traditional and interoperable | centralized (except for distributed view of CpBP) | GUI and notifications | support | not visulalize execution, generate reports |
| Bonita | BPMN 2.0 (not complete) | Managed by FS of OS | CrBP and CpBP | support | Drools Groovy | support | support | automatic | centralized | BPMN 2.0 (not complete) | Managed by FS of OS | N/A | centralized | GUI and notifications | support | visualize execution, generate reports |
| Oracle BPM | BPMN 1.2 and some BPMN 2.0 | Managed by DBMS | CrBP and CpBP | support | XPath | support | support | (semi-) automatic | centralized | BPMN 2.0 (not complete) | N/A | traditional and interoperable | centralized | GUI and notifications | support | visualize execution, generate reports |
| ADONIS | BPMN 1. 0 (not complete) | Managed by DBMS | CrBP and CpBP | N/A | N/A | N/A | support | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| MEGA | BPMN 1.2 | Managed by DBMS | CrBP and CpBP | N/A | N/A | N/A | support | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |

# II. MODEL DRIVEN ARCHITECTURE AND MODEL DRIVEN INTEROPERABILITY

Model Driven Architecture (MDA) was firstly proposed by OMG in 2001 and the final specification was adopted in 2003. It is an approach to using models in software development. Its primary goal is to improve portability, interoperability and reusability of software solutions through architectural separation of concerns and model transformations.

MDA divides models in three levels (Miller and Mukerji 2003):

- **Computation Independent Model (CIM):** a view of a system from its environment and its business requirements.
- **Platform Independent Model (PIM):** a view of a system which focuses on the operation of the system but hides the details necessary for a particular platform.
- **Platform Specific Model (PSM):** a view of a system which combines the specifications in the PIM with the details of the use of a specific platform.

MDA uses model transformation to generate models at a lower level from models at an upper level. For example, MDA can use transformation rules to generate PSM models from PIM models. During the model transformation, additional information may be added, such as target platform description information captured in a **Platform Model**. At last, MDA will generate executable code from PSM models. So, MDA improves degree of automation of software development.

According to the above narration, model transformation is a key part of MDA, which will be discussed in Section II.1. Section II.2 will discuss how MDA supports enterprise interoperability. This problem is studied in the research domain Model-Driven Interoperability (MDI).

## II.1. Model transformation

Figure 2-9 provides an abstract architecture dedicated to “model to model” transformation. In Figure 2-9, models have three abstraction levels: Model (M1), MetaModel (M2) and MetaMetaModel (M3). Models at M1 level conform to models at M2 level and models at M2 level conform to models at M3 level ("conform" means that both the elements of a model at $M_i$ level are instances of elements of a model at $M_{i+1}$ level and the well-formed rules are satisfied). That is to say M3 and M2 defines the structure and semantics of metadata for M2 and M1. In Figure 2-9, source mode at M1 level conforms to source metamodel at M2 level and source metamodel conforms to the metametamodel at M3 level. Target model and

target metamodel are in the same situation as source model and source metamodel. The transformation from source model to target model is based on transformation rules (mappings) between source metamodel and target metamodel. The transformation rules conform to a rule language. In addition, source and target metamodels are not necessary to share the same metametamodel. In fact, there are numerous model transformation approaches which have been studied in (Czarnecki and Helsen 2003). (Czarnecki and Helsen 2003) has characterize the design features of the approaches and then classified the approaches into two kinds: model-to-model approaches and model-to-code approaches.

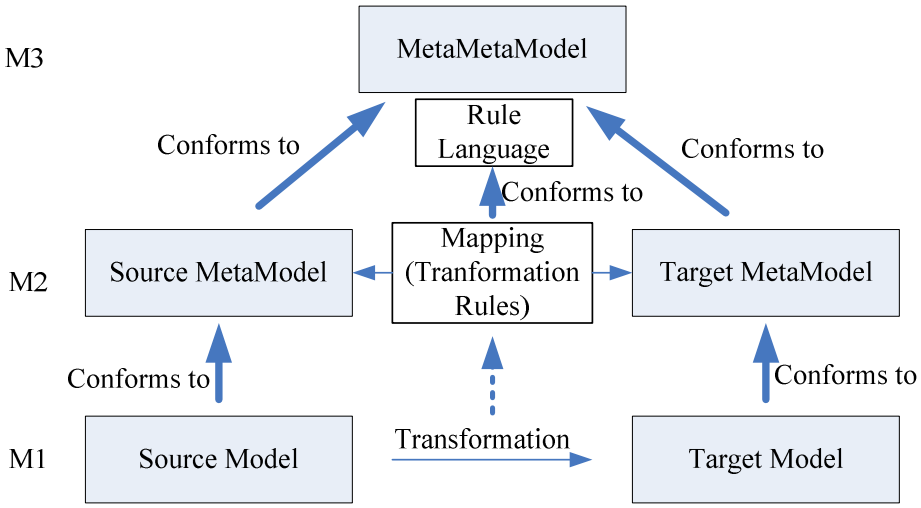



**Figure 2-9. Abstract Architecture for Model-to-Model Transformation**

Nowadays, ATL [23] language (**A**tlas **T**ransformation **L**anguage) is prevalent as the transformation rule language. ATL is supported for example by Topcased[24], an eclipse-based platform which provides a development environment and an execution environment for ATL transformation rules (see Figure 2-10). When ATL is assessed by design features of model transformation proposed in (Czarnecki and Helsen 2003), the following characteristics of ATL can be outlined:

- ATL can be used to create declarative and imperative transformation rules; rules are organized into modules;
- During execution, rules are selected by explicit conditions; and execution of rules is deterministic;
- The model transformation is unidirectional and it can only create new targets; the transformation has no traceability links.

[23] http://www.eclipse.org/m2m/atl/doc/

[24] http://www.topcased.org/

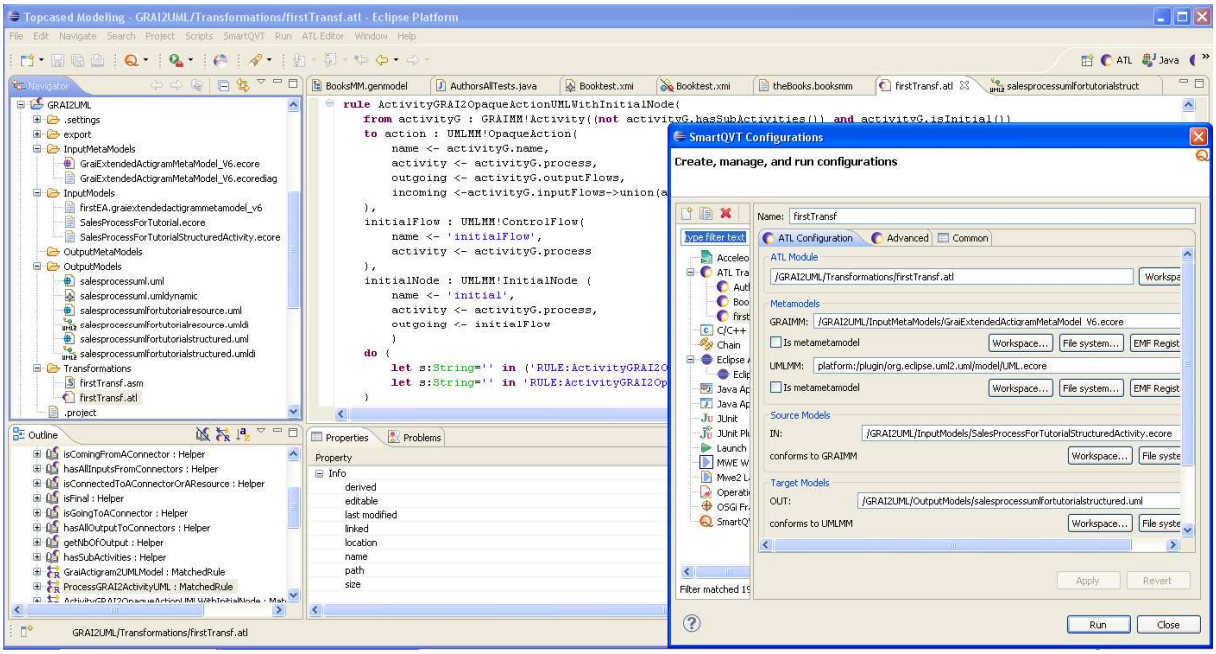



**Figure 2-10. Example for ATL transformation rules developed in Topcased v4.3.0**

## II.2. Model Driven Interoperability

**Model Driven Interoperability** Method (MDI Method) is based on MDA and it can be used for two enterprises that need to interoperate not only at the code level but also at Enterprise Modeling level with an ontological support (Bourey, Grangel et al. 2007). Jean-Pierre Bourey and his colleagues have represented this method in Figure 2-11 as Reference Model for MDI.

In this reference model, CIM level in MDA has been enriched and it is divided in two levels: Top CIM level, which represents business requirements from the viewpoint of business users, and Bottom CIM level, which represents business requirements from the viewpoint of software developers. Besides, in order to solve interoperability problems, interoperability model is established at each abstraction level. Interoperability model will be transformed from upper level to lower level with the help of common interoperability ontology constructed by two enterprises. In this reference model, the transformation from upper level to lower level is defined as **vertical transformation**. Meanwhile, the transformation at the same level between different enterprises is defined as **horizontal transformation**. Vertical transformation is primarily for code generation; instead, horizontal transformation is primarily for model exchange or consistency verification between different enterprises.

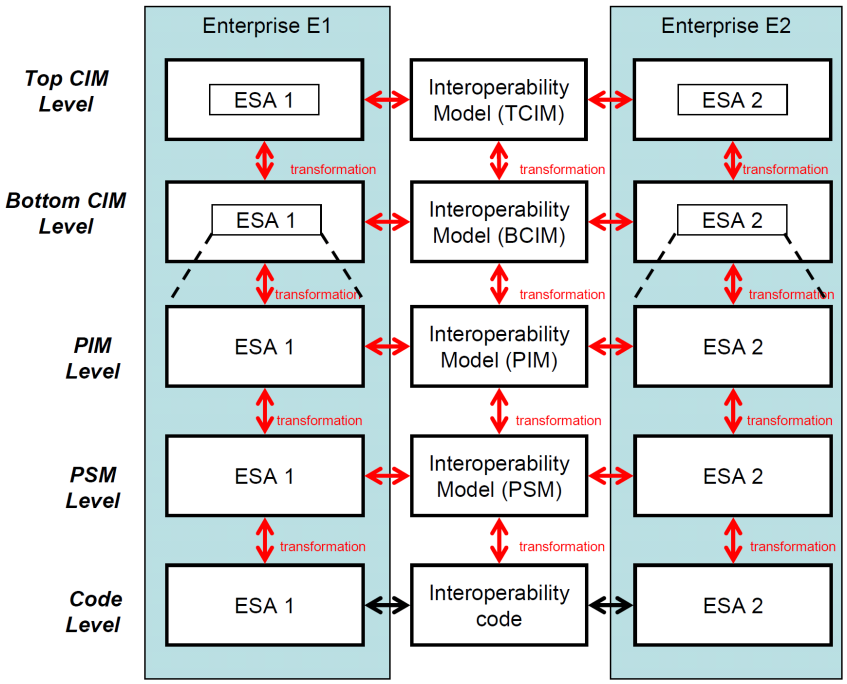



**Figure 2-11. Reference Model for MDI (Bourey, Grangel et al. 2007)**

## II.3. Conclusions

In theory, MDA can provide software developers with the benefits of reusability, portability and interoperability. However, in practice, MDA is still far from its expectation. In addition, MDA is doubted by some researchers, such as Scott W. Ambler. He is afraid of that MDA will suffer from the same problem of its predecessor "Integrated Computer-Aided Software Engineering (I-Case)" (Ambler 2003). For example, the I-Case tools generated 80 to 90 percent of the code, but the rest 10 percent required 90 percent of the efforts (Ambler 2003). So, when MDA/MDI is employed to solve enterprise interoperability problems, it is better to generate **executable models** instead of **codes**. The target models can be **executable collaborative business processes** which can be executed by business process engines. Therefore, when modeling enterprise interoperability problems, MDA can be used as the skeleton of related solutions (see Figure 2-15).

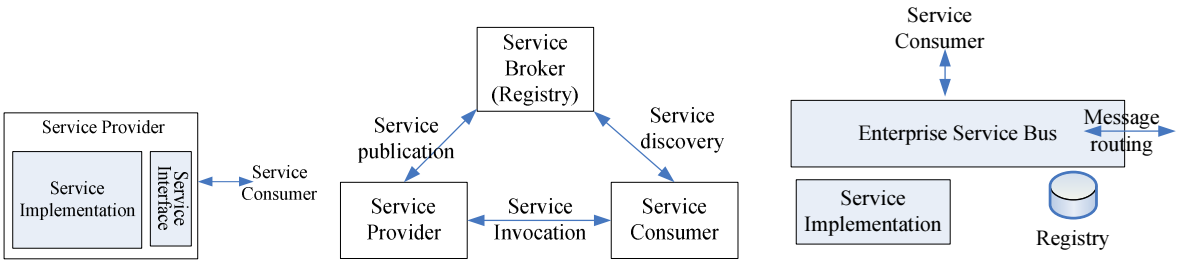
# III. SOA

**Service Oriented Architecture** (SOA) is a paradigm for organizing and utilizing distributed **capabilities** that may be under the control of different ownership domains (OASIS 2006). SOA was firstly described by Gartner in 1996 (Natis 2003) in order to make enterprises agilely adapted to business changes. However, at that time, SOA just stayed at the ideological level. With the development of XML, SOAP and WSDL, web service was widely

used in e-business domain, and many organizations were aware of that web service could not only provide capability of distributed software applications, but it could also be regarded as an architecture foundation. Therefore, web service became popular as an implementation technology of SOA, and moreover the emergence of UDDI further enriched SOA. SOAP, WSDL and UDDI are regarded as the first generation of web service protocols. These protocols can guarantee web service interoperability to a certain extend.

Consequently, with active collaboration of many software vendors, the second generation of web service protocols (WS-* protocol family) was gradually proposed, such as WS-Addressing, WS-ReliableMessaging, WS-Policy, WS-Security, WS-Eventing, WS-BPEL and WS-CDL, etc. These protocols provide some important and necessary capabilities for enterprise software applications. Especially, WS-BPEL and WS-CDL can implement intra-/inter-organizational business processes through service composition. This makes enterprises easily adapted to business changes. So, SOA/web service gains attentions from more and more enterprises, and the research about SOA/web service has been shift from registry-based pattern to enterprise service bus[25]-based pattern (see Figure 2-12).

With the development of SOA/web services, many enterprises have published their business logic as web services, but SOA is not just services. Danny Sabbah, general manager of IBM Rational, said "*SOA is 1% services and 99% governance*". So, governance is very important in SOA and some aspects of SOA governance have been implemented by enterprise service buses, such as JSSOA (Liu 2008).




(a) Client-Server Pattern (b) Registry-based Pattern (c) ESB-based Pattern

**Figure 2-12. SOA evolution**

Around our research theme in this thesis, this section will concentrate on some SOA and service characteristics which are relevant to enterprise interoperability.

[25] The research about enterprise service bus is introduced in Section IV.

### III.1. SOA and service

The core of SOA is the notion "service". A **service** is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is consistent with constraints and policies as specified by the service description (OASIS 2006). According to the above definition, a service has at least three aspects (see Figure 2-12.a): a) service implementation (capabilities, such as business functions), b) service description including service interface, constraints and policies, and c) service interaction. The separation of service implementation and service description makes services have several important characteristics: autonomy, loose coupling, interoperability and reusability.

- **Autonomy**: as service interface is the only access way provided for a service consumer, so any changes outside of services cannot influence service implementations.
- **Loose coupling**: as service interface separates service implementation from service consumers, so service consumers just need to know the address of the service interface, do not need to know the address of the service implementation. Furthermore, interaction between a service consumer and a service provider can be asynchronous, that is to say, the consumer does not need to ask the provider to immediately respond to his request. Besides, any change in a service implementation will not influence service consumers if the corresponding service interface is not changed.
- **Interoperability**: Service interfaces hide heterogeneity of service implementations and service running environments. Therefore, service consumers do not need to know how service is implemented and they just need to follow service interface to access service. As service interface is usually based on standards, such as WSDL, so the interoperability between service providers and service consumers can be guaranteed.
- **Reusability**: as business functions (including business functions in legacy systems) can be published as services and services are loosely coupled and interoperable, so services can be reused by different service consumers.

The above service characteristics bring up some important characteristics for SOA, such as **composability** and **loose coupling**.

- **Composability**: as services are loosely coupled and interoperable, they can be easily composed together to satisfy complex business requirements. Service composition can make enterprises have business agility.
- **Loose coupling**: as services are loosely coupled, therefore in an SOA-based information system, different software components do not closely depend on each other and the relationship between them is grounded in their service interfaces. So, the whole system is loosely coupled and has flexibility. This characteristic will be discussed in detail in Section III.2.

SOA use services to establish mappings between business environment and IT environment (see Figure 1-4). Services can be regarded as a modeling method for business requirements. SOA provides some principles for software design; meanwhile it can also be regarded as an architecture style to react against changes from business or IT environments.

### III.2. Loose coupling in SOA

Loose coupling for services and SOA makes original monolithic systems developed and deployed in a distributed method in computer networks. An SOA-based information system can be loosely coupled in one aspect but tightly coupled in another aspect. (Schmelzer 2007) has proposed seven levels of loose coupling. The seven levels are shown in Table 2-2. For each level of loose coupling, Table 2-2 lists corresponding objective, encountered difficulties and possible solutions. As the implementation technique of SOA, web service can realize loose coupling at the implementation and process levels, but it cannot achieve loose coupling at service contract/policy levels and semantic level. Because in web service, any service invocation will ask service consumers to know WSDL definition (including service interface and related data types) of the target web service. Any change of service interface or related data types (for input or output messages) will influence service invocation of service consumers. Hence, this tightly couples service consumers and service providers. In Chapter 5, we will propose an SOA-based semantic service bus to realize loose coupling at the service contract and semantic levels.

### III.1. Conclusions

SOA and service are important to enterprise interoperability. On one hand, SOA and service can provide loose coupling and interoperability between information systems of different enterprises; on the other hand, SOA and service can react rapidly against changes of collaboration requirements between different enterprises.

**Table 2-2. Seven Levels of Loose Coupling (adapted from (Schmelzer 2007))**

| Levels | Objective | Difficulties | solutions |
|---|---|---|---|
| **Implementation** | Service consumers are blind to the implementation technology used by service providers and vice-versa | Different implementations: Java, .NET, PHP, C++, or Basic. Service implementation may be changed | Service contract (standard-based, interoperable specifications or protocols) (XML, Web service, REST service) |
| **Service contract** | Contract changes do not cause service consumer breakage | A simple change to acceptable inputs or functional behavior of the system can have profound impact on service consumers. | Service contract change management (late-binding, intermediary-enabled, registry-based systems) |
| **Service Policy**[26] | Policy changes do not cause service consumer breakage | A small change to a Service policy can have tremendous repercussions | Service policy versioning and deprecation |
| **Process** | Service consumer should not have to know at all when a process is reconfigured. | | Expose a composite service as a service |
| **Data Schema** | Organizations need to further their loose coupling goals by enabling dynamic and heterogeneous change to the data schema shared between Service consumers and providers. | If a service consumer and provider need to have a common understanding about data schema, we have tight coupling as defined above. | Exception management, transformations, service intermediaries, and Data Services. |
| **Infrastructure** | Service implementation is infrastructure neutral | If loose coupled systems move their implementation from one ESB or Service infrastructure to another, then all hell will break loose. | Many vendors promise this sort of interchangeability, but few deliver. |
| **Semantic Layer** | Provide the promise of seamless data integration | If the data structures of service providers are imposed on service consumers, the result is as tightly coupled as previous architectural approaches. | **Dynamic** service definitions (the definition of a service interface must change based on the context of a service consumer.) |

## IV. ENTERPRISE SERVICE BUS (ESB)

In order to effectively integrate systems distributed in computer networks, middlewares emerged (Schantz and Schmidt 2001). However, traditional middlewares focus primarily on the technical interoperability level, so in order to resolve syntactic, semantic and pragmatic

[26] A **policy** is a form of metadata, as are contracts, and in fact, the only difference between a service policy and a contract is that a policy can apply to any number of Services. Because policies control many aspects of the non-functional parts of a Service. (Schmelzer 2007)

interoperability problems, Enterprise Application Integration (EAI) was proposed (Puschmann and Alt 2001). EAI can be used to integrate distributed and heterogeneous information systems in an enterprise or across boundaries of enterprises. EAI has been researched, implemented and used in many enterprises (Puschmann and Alt 2001; Losavio, Ortega et al. 2002; Reiersgaard, Salvesen et al. 2005). However, EAI has at least two inherent limitations (IONA 2006):

- Central control feature of its architecture causes its performance bottleneck. Although central controller can be deployed in a computer cluster, the improved performance is limited and a computer cluster is quite expensive.
- Continuous addition of new features makes EAI large, inflexible and hard to manage.

The limitations of EAI have hampered its own development. As the next generation of enterprise integration technology, Enterprise Service Bus (ESB) came out (see Figure 2-12.c). ESB has some advantages over traditional EAI (IONA 2006):

- ESB is service-oriented and grounded in many open standards; all business functionalities are published as services.
- ESB is light-weight and it is easy to be deployed in a distributed method. Hence, there is no performance bottleneck caused by system architectures. Business functionalities can be deployed into multiple ESBs; unlike in EAI, all of them can only be deployed into one centralized hub.

So far, there is no precise definition about ESB, and many enterprises or researchers have their own viewpoints about ESB. Generally speaking, ESB usually has the following features:

- Support service creation, registration, discovery, invocation and composition. (ESB examples: Celtix[13], Mule[15], JSSOA (Liu 2008), Petals[14])
- Support message transformation from three aspects: data types in messages, message formats (such as SOAP, XML) or message transport protocols (such as HTTP, JMS); support message routing. (ESB examples: Mule and Petals)
- Support ESB governance (authorization/authentication, access control, service deployment, monitoring service running, monitoring ESB nodes, etc). (ESB examples: JSSOA)
- Support event-driven architecture (asynchronous production and consumption of messages). (ESB examples: Celtix and Mule)

The above features make ESB more loosely coupled and easier to integrate legacy systems and they also make enterprises more agile to be adapted to business changes. Most importantly, an ESB is an open system, that is to say, the ESB can invoke services provided by other ESBs and the ESB can also make its own services invoked by other ESBs. So collaborations between different enterprises can be supported by their ESBs. However, the above ESBs (Celtix, Mule, Petals or JSSOA) cannot deal with problems of semantic heterogeneity when they are used to integrate information systems or realize collaborations between enterprises. Therefore, new ESB: semantic service bus has come out. Semantic service bus has been studied in (Karastoyanova, Wetzstein et al. 2007; BEDNÁR, FURDÍK et al. 2009). In (Karastoyanova, Wetzstein et al. 2007), semantic service bus is generated through integration of a conventional ESB and two semantic web service platforms. This semantic service bus uses semantic descriptions of services and exchanged messages to automate service discovery, selection and invocation, and message transformation and routing. The semantic service bus also annotates composite services with semantic information and executes them according to the semantic information. (BEDNÁR, FURDÍK et al. 2009) has proposed a design of a semantic service bus (named SPIKE) for networked enterprises. SPIKE platform uses a semantic manager to do semantic information manipulation, such as semantic search, matching, mediation, mapping and reasoning. In Chapter 5 of this thesis, after analysis of semantic web services and goals, we will propose our own semantic service bus for enterprise interoperability.

## V. ONTOLOGY

The word “ontology” comes from the Greek “ontos” (for being) + “logos” (for word) (Gasevic, Djuric et al. 2006). In philosophy, it refers to the subject of existence, i.e., the study of being as such (Gasevic, Djuric et al. 2006). Ontology emerged in the domain of computer science and technology some time ago as a means for sharing knowledge in Artificial Intelligence (Gruber 1993). In (Gruber 1993), it defines ontology as a specification of a conceptualization. This definition is the most widely cited one. There are also other definitions, for example:

- Ontology can be seen as the study of the organization and the nature of the world independently of the form of our knowledge about it (Guarino 1995).
- Ontology is the basic structure or armature around which a knowledge base can be built (Swartout and Tate 1999).

- Ontology is a set of knowledge *terms*, including the vocabulary, the semantic interconnections, and some simple *rules* of inference and logic for some particular topic (Hendler 2001).
- An ontology is an explicit representation of shared understanding of the important concepts in some domain of interest (Kalfoglou and Schorlemmer 2003).

### V.1. Why do we need ontology?

Research on ontologies has turned into an interdisciplinary subject including *Philosophy, Linguistics, Logics, and Computer Science* (Jarrar and Meersman 2008). In computer science, *ontologies* are becoming increasingly essential for nearly all applications. Particularly, the Internet and other open connectivity environments create a strong demand for sharing semantics of data (Jarrar and Meersman 2008).

Within computer science, the research on ontologies emerged "mainly" within two sub-communities: **artificial intelligence** and **database** (Jarrar and Meersman 2008). For example, in the AI community, ontologies have gained popularity as a means for establishing explicit formal vocabulary to share between applications (Noy 2004). In the fields of **databases** and **information integration**, researchers and practitioners have produced a large body of research to facilitate interoperability between different systems. Among these studies, **ontology** is one discipline that deals with **semantic heterogeneity** in structured data (Noy 2004).

Nowadays, the distributed and heterogeneous **information systems** between enterprises also depend on "ontology" to make them *meaningfully* communicate to exchange data and thus make their transactions *interoperate* independently of their internal technologies (Jarrar and Meersman 2008). This is the problem of enterprise semantic interoperability. In order to achieve semantic interoperability between heterogeneous information systems, the meaning of interchanged information has to be understood across systems (Wache, Vögele et al. 2001). The use of ontologies for the explication of implicit and hidden knowledge is a possible approach to overcome the problem of semantic heterogeneity.

### V.2. Research domains of ontology

When evaluating the approaches to ontology-based information integration, (Wache, Vögele et al. 2001) has proposed four main criteria: role/architecture of ontologies, ontology representation, ontology mapping and ontology engineering. Appendix B elaborates each criterion in detail. The following sections will discuss relationship between ontology and

other research domains, such as information integration/interoperability and MDA. The relationship between ontology and business processes will be discussed in Chapter 4. The relationship between ontology and SOA/ESB will be discussed in Chapter 5.

#### V.2.a. Relationship between Ontology and information integration

The problem of bringing together heterogeneous and distributed computer systems is known as *interoperability problem* (Wache, Vögele et al. 2001). Interoperability has to be at least provided at technical and informational levels. In short, information sharing not only needs to provide full accessibility to data, but it also requires that the accessed data may be processed and interpreted by remote systems. Problems that might arise owing to data heterogeneity can be divided into two levels: schematic heterogeneity and semantic heterogeneity. Schematic heterogeneity (Goh 1997) may be caused by data type conflicts, labeling conflicts, aggregation conflicts, generalization conflicts between different databases/information systems. Semantic heterogeneity (Goh 1997) primarily comes from the naming conflicts, scaling and units conflicts and confounding conflicts between different systems. In order to achieve semantic interoperability, ontologies and semantics-based technologies in general will play a key role to overcome the problem of semantic heterogeneity (Wache, Vögele et al. 2001; Uschold and Gruninger 2004). Uschold and Grüninger mention **interoperability as a key application of ontologies**, and many ontology-based approaches (Uschold and Grüninger 1996) to information integration in order to achieve interoperability have been developed. (Wache, Vögele et al. 2001) has reviewed the use on ontologies for the integration of heterogeneous information sources. Based on the results of its analysis, it summarizes that: in a typical system, integration should be done at the ontology level using either a common ontology that all source ontologies are related to or fixed mappings between different ontologies.

#### V.2.b. Relationship between Ontology and Models

##### V.2.b.i. Ontology modeling with modeling language

RDFS, OWL, and Topic Maps (TM) are commonly used in the semantic web community for expressing vocabularies, ontologies, and topics, respectively. The Ontology Definition Meta-models (ODM) (OMG-ODM 2009), standardized by OMG, defines the meta-models and UML profiles of the above three ontology languages in the modeling space MOF. These meta-models and profiles enable the use of UML notation (and tools) for ontology modeling and facilitate generation of corresponding ontology descriptions in RDF

(Resource Description Framework), OWL, and TM, respectively. In addition, to support the use of legacy models as a starting point for ontology development, and to enable ODM users to make design trade-offs in expressivity based on application requirements, mappings among a number of the meta-models are provided, such as the mappings from UML and TM to OWL and from RDFS/OWL to Common Logic (CL).

In order to develop ontology, besides the above method, (Héon, Paquette et al. 2008) also provides another methodology. Appendix B has explained it in detail. In general, the above two methods are all based on MDA.

#### V.2.b.ii. Model transformation based on ontology

(Roser and Bauer 2006) has proposed an ontology-based approach to model transformation, depicted in Figure 2-13 and Figure 2-14. This approach needs the following parts to achieve ontology-based model transformation:

- ***Semantic Transformation*:** A semantic transformation is a transformation specification describing a transformation between two ontologies. A semantic transformation is specified between a source ontology and a target ontology (see Figure 2-13), but it can also be bidirectional.
- ***Syntax-semantic Binding*:** The syntax-semantic binding specifies the connection between syntax (metamodels) and semantics (ontologies).
    - ***MO-Binding*:** *(Metamodel-ontology)* MO-Bindings specify how semantic information can be derived from model elements.
    - ***OM-Binding*:** *(Ontology-metamodel)* OM-Bindings specify how ontology elements are expressed in models.

In Figure 2-14, a combination of one semantic transformation, one MO-Binding and one OM-Binding form a *transformation configuration*. A generator for model transformations takes a transformation configuration as well as appropriate metamodel- and ontology-definitions as input. The generator outputs a model transformation specified in an *intermediate model transformation language*. Such model transformation will be translated into a specific transformation language.

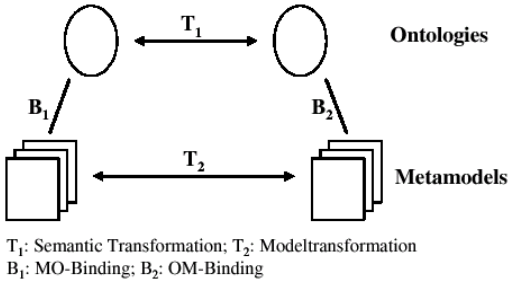



**Figure 2-13. Concept of ontology- based model transformation (Roser and Bauer 2006)**

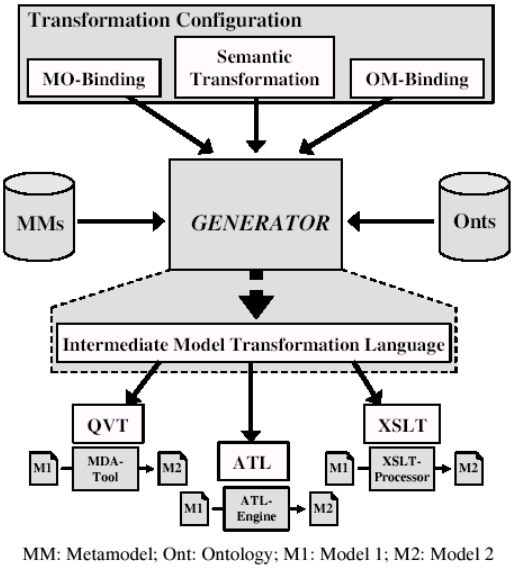



**Figure 2-14. Overall approach of ontology-based model transformation (Roser and Bauer 2006)**

## V.3. Conclusions

In (Wache, Vögele et al. 2001), after the analysis of 25 approaches to intelligent information integration, it finds that there are still two important problems that should be solved. For the first problem, there is a need to investigate mappings on a theoretical and an empirical basis. That is because most of approaches in integration systems still use ad-hoc or arbitrary mappings especially for the connection of different ontologies. There are approaches that try to provide well-founded mappings, but they either rely on assumptions that cannot always be guaranteed or they face technical problems. For the second problem, (Wache, Vögele et al. 2001) finds a striking lack of sophisticated methodologies supporting the development and use of ontologies. Such methodology has to be language-independent and includes an analysis of the integration task. It also has to support the process of defining roles of ontologies with respect to the requirements.

In order to achieve fully automatic semantic interoperability among independently developed and heterogeneous agents, individual researchers and practitioners will have to initially make many assumptions, and then relax them one by one as technology progresses (Uschold and Gruninger 2004).

Many semantic mapping, integration and/or interoperability projects take place more or less in a vacuum because of *lacking some general infrastructure* in place where one can easily register, access and use various things such as: ontologies, mappings between ontologies, mapping languages, and translation engines.

It is also prevalent that ontology is closely connected with models. Nowadays, MDA has been researched by numerous enterprises and universities, and how to add semantic information into models and how to transform models with the help of ontology will be very interesting and they are important in order to realize enterprise semantic interoperability. In Chapter 4, we will provide a way to add ontology information into collaborative business processes (models) and to transform collaborative business processes with the help of the added ontology information.

## VI. CONCLUSIONS

According to the study of business processes, MDA, SOA, ESB and ontology, we propose a framework for IT solutions to enterprise interoperability problems in Figure 2-15. Figure 2-16 provides an individual view of the framework. In Figure 2-16, for only one enterprise, business requirement[27] (replaced with collaboration requirement in Figure 2-15) is proposed from business environments.

The realization of business requirement is the alignment between business environment and IT environment. During the alignment,

- **MDA** is adopted as the **skeleton** of the IT solution framework. There are three levels: CIM, PIM and PSM. Models at lower level will be transformed from models at upper level. Model transformation will depend on information from ontology. All results generated in different levels must respect and conform to business collaboration requirement.
- **Business process** is adopted as the **representation** method for business requirement. Business process will be annotated by ontology information.

[27] Figure 2-16 uses "business requirement" instead of "collaboration requirement". This makes Figure 2-16 more general, not only for collaboration but also for general business purpose.

- **SOA** is adopted as the modeling **principles**. Information about services used in business processes will be stored in ontology.

Ontology used in the above three techniques is constructed from business and IT environments by following an ontology engineering methodology. **Ontology** is employed to solve **semantic** problems about data, service and process.

After modeling business requirements according to the above three techniques, we obtain executable collaborative business processes. The processes will be executed on a platform or an infrastructure. **Semantic service bus** will be a good choice for the **platform/infrastructure** as discussed in Section IV. Meanwhile, ontology will be also used in IT environment when semantic service bus deals with exchanged messages, discovers services or executes processes.

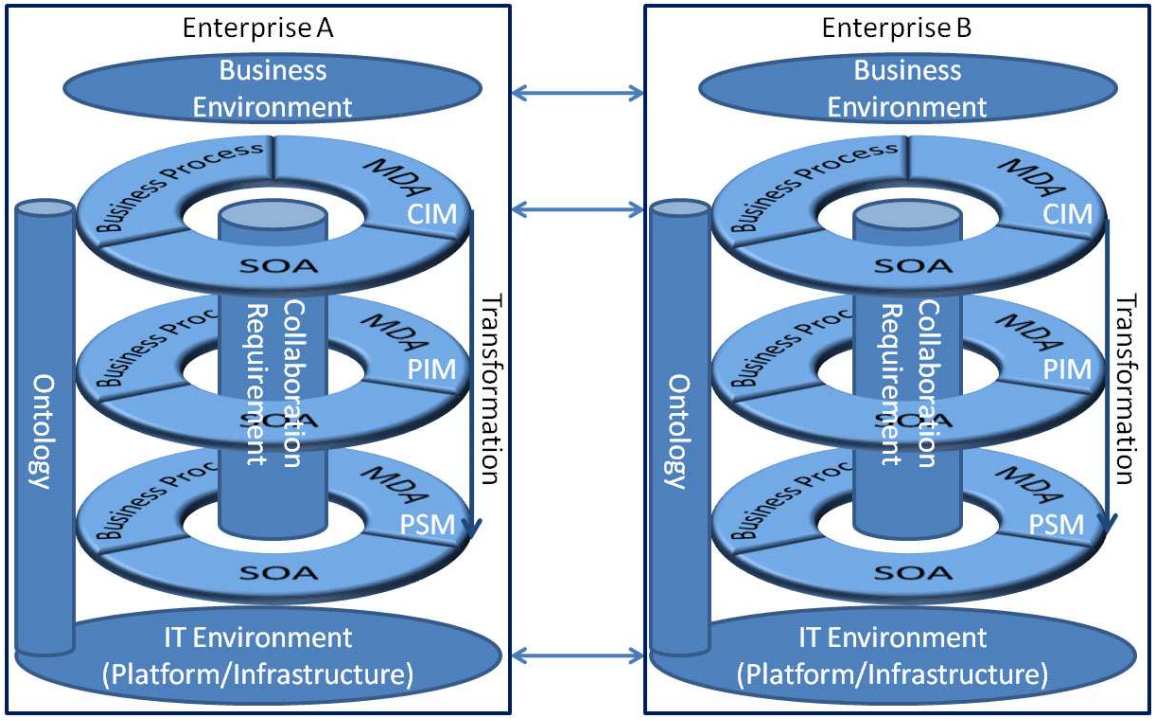



**Figure 2-15. Framework for IT Solutions to Enterprise Interoperability Problems**

In Figure 2-15, collaboration requirement (replacing business requirement in Figure 2-16) is generated from business environments of two enterprises. The interaction between the two business environments means that their collaboration needs some human interactive activities. Collaboration requirement is the motivation and the core of enterprise interoperability because if there is no such requirement, there is no need to realize enterprise interoperability.

In Figure 2-15, once collaboration requirement is modeled at CIM level in one enterprise, the result models (collaborative business processes) will be delivered to its

collaborator(s). The processes will be the blueprint for their collaboration and then each collaborator will generate their own collaborative business processes. At last, each collaborator will transform the processes to executable processes for their own platform/infrastructure.

During execution of the target collaborative business processes, the platforms/infrastructures from different enterprises will interact with each other and the business environment (human) and IT environment (information system) may also interact with each other. The interaction between business and IT environments is not indicated in Figure 2-15 for the sake of clarity.

The process of modeling collaboration requirements according to business process, MDA and SOA will be discussed in detail in Chapter 3 and 4. The platform/infrastructure for running collaborative business processes will be discussed in Chapter 5.

During modeling and execution of collaborative business processes, ontology will be employed and its usage will be discussed in Chapter 3, 4 and 5.

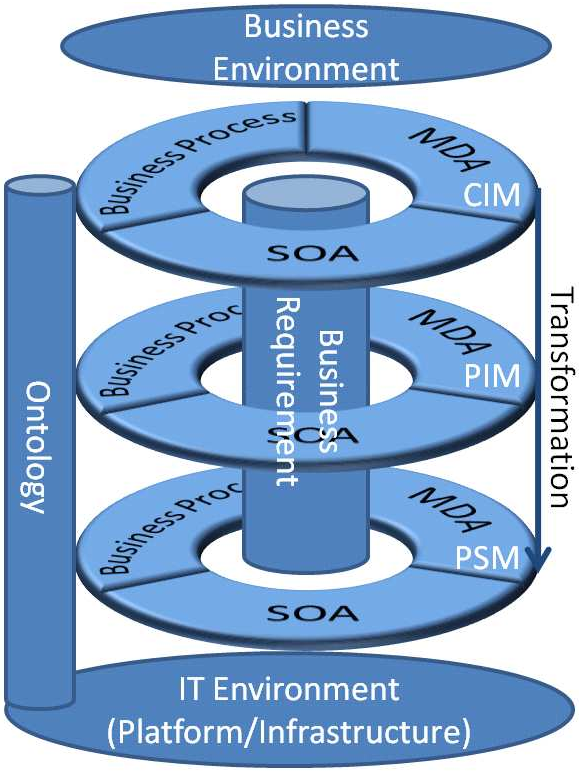



**Figure 2-16. Individual View of the Framework for IT Solutions to Enterprise Interoperability Problems**

# CHAPTER 3: Process -Based Method for Enterprise Interoperability

In order to realize the framework proposed in Chapter 2 for IT solutions to enterprise interoperability problems, this chapter will propose a "Process-Based Method for Enterprise Interoperability" (PBMEI) in Section II. PBMEI employs collaborative processes to represent collaboration requirements between enterprises. PBMEI transforms a collaborative process to multiple executable interoperability processes. The generated interoperability process will be deployed and executed in an infrastructure. In order to explain PBMEI, a case named "ShoppingDrive" cooperation process will be studied in Section II.2.

Before further discussion, some important terminologies widely used in the thesis will be defined in Section I in order to avoid unnecessary misunderstanding.

# I. TERMINOLOGY DEFINITION

## I.1. Key concepts about enterprise collaboration/interoperability

**Collaborative business process** is used to model collaboration requirements between enterprises and it is implemented by interoperability processes after some steps of transformation. According to the MDA framework, we can see that collaborative business process belongs to the level CIM and interoperability process belongs to PIM and PSM. The **actors** in collaborative business process can be enterprises or departments, and such actors are defined as **collaborators**; the actors in interoperability process can be information systems, sub-systems, components or services, they are defined as **participants**. The above concepts are positioned in MDA framework in Figure 3-1.

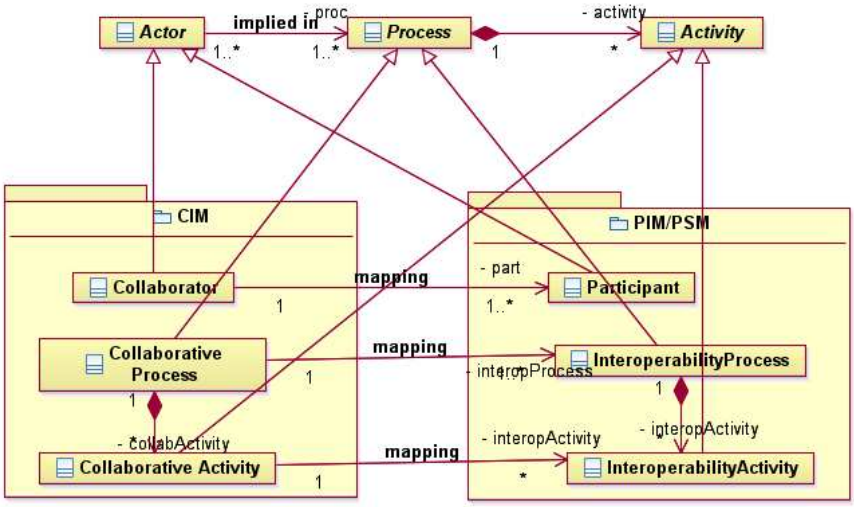



**Figure 3-1. Position of collaboration/interoperability concepts in the MDA framework**

## I.2. Classification of business processes

To further analyze enterprise interoperability problems, we analyze the classification of business processes. (Dumas, van der Aalst et al. 2005) has proposed several criteria to classify business processes. We will classify business processes according to the following criterion: the **quantitative relationship** between the **owners** and **controllers** of business activities. The relationship between owners and controllers is indicated in Figure 3-2.

- The owner of business activity is the actor who is responsible for implementing and performing this activity;
- The controller is the actor who starts the activity.

Following the above criterion, there are three kinds of processes in or between enterprise information systems:

1) The **internal process**: it is composed of the activities which belong to the same IS of an enterprise;
2) The **coordination process**: it is composed of the activities some of which take place between several IS and/or enterprises, but the process execution is owned and controlled by only one IS and/or enterprise;
3) The **cooperation process**: it is composed of the activities some of which happen between several IS and/or enterprises and the process execution is owned and controlled by several IS and/or enterprises, but each IS and/or enterprise can only control the execution of its own activities.

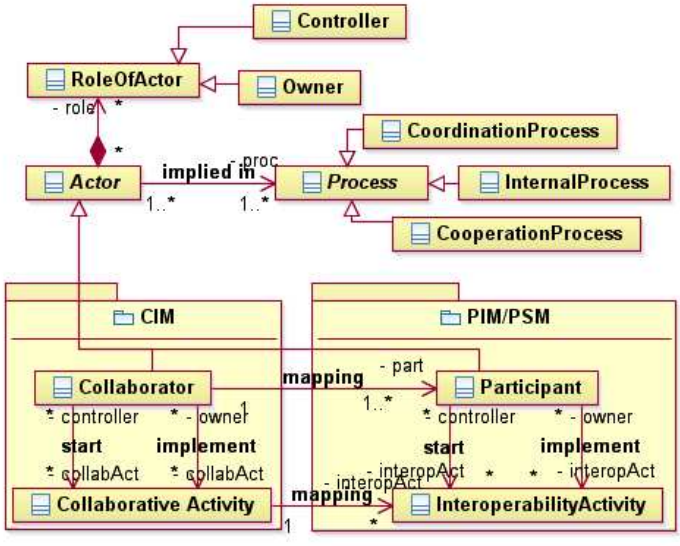



**Figure 3-2. Position of the concepts: owner, controller and three types of processes**

According to Figure 3-1, Figure 3-2 and the definition of collaborative business process, a collaborative process in the CIM package can be represented as each of the three types of business processes. If the collaborative process is the internal process, i.e., the process is across different departments in one enterprise, the collaborator is just the enterprise itself; if it is the coordination process, the collaborator who controls the process execution is named the **coordinator (or mediator)**, other collaborators are named **passive collaborator**s; if it is the cooperation process, the collaborator who controls the process is named **principal cooperator** and the collaborator who controls its own activities but does not control the process is named **secondary cooperator**. The relationship between these concepts is described in Table 3-1.

**Table 3-1. Relationship between roles of collaborators in Collaborative Business Processes**

| **Collaborator** | | **Controls its own activity in the process** | | **Has no activity in the process** |
|---|---|---|---|---|
| | | **no** | **yes** | |
| **Controls process** | **no** | Passive collaborator | Secondary cooperator | X |
| | **yes** | X | Coordinator, Principal cooperator[28] | Coordinator |

Note: "X" means no definition.

**Table 3-2. Roles of Actors in Business Processes**

| **Business process** | **Collaborator**[29] | **Participant** |
|---|---|---|
| Internal process | X | X |
| Coordination process | Coordinator (mediator), Passive collaborator | Requester, Provider |
| Cooperation process | Principal cooperator, Secondary cooperator | Requester, Provider, Subscriber, Publisher |

Note: "X" means no definition.

According to Figure 3-1and Figure 3-2, in the PIM/PSM package, for each of the last two types of business processes, if **one of its activities is the interoperability activity, then the process is an interoperability process**. In addition, with regard to an internal process, it will be necessary to implement information exchanges between some modules of enterprises' IS. These exchanges are carried out by the activities that can be considered as internal interoperability activities. Hence, the three types of processes can be implemented as

[28] If in a business process, there is only one collaborator who not only controls its own activity in the process but also controls the process' execution, such collaborator is coordinator; otherwise, it is principal cooperator. So we can see that the quantity of principal cooperators must be more than or equal to 2.

[29] In a collaborative business process, roles chosen for collaborators are determined by collaboration requirements between enterprises.

interoperability processes. For an interoperability process, if it is the coordination process, its participants can play the roles "requester" and "provider"; if it is the cooperation process, its participants can play the roles "requester" and "provider", "subscriber" and "publisher"; if it is the internal process, it can be executed as coordination process or cooperation process and its participants can be that of coordination or cooperation process. So we can get Table 3-2.

## I.3. Rank of collaborative process, NCA and NCP

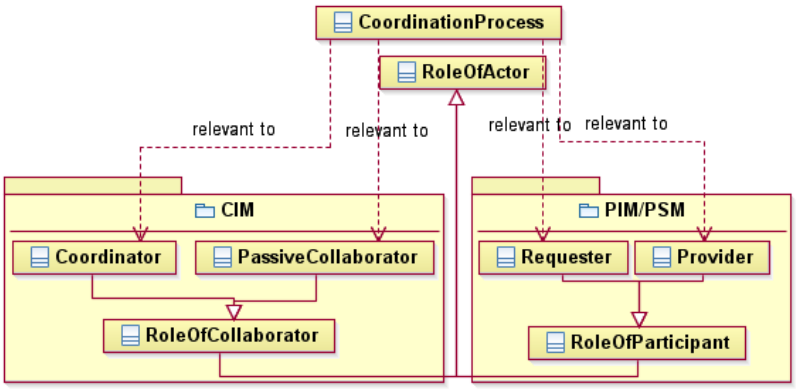
To characterize the complexity of enterprise collaboration/interoperability, we distinguish two essential characteristics: the number of owners for collaborative activities (called the **Rank of collaborative process**, noted as **R**) and the **Number of Controllers for collaborative Activities (**noted as **NCA**). As each activity in a collaboration process must belong to a collaborator, however, the execution of the activity may not necessarily be controlled by its owner, therefore the number of controllers for a collaborative activity will be less than or equal to R. If the number of controllers is 1, i.e., if only one enterprise is responsible for controlling execution of all the activities in a collaborative process, the process is actually an internal process or a coordination process. If the rank is greater than 1 and the NCA is 1, the process is a coordination process. In a coordination process, the collaborator corresponding to the controller is defined as an **active collaborator** (or **coordinator**, or **mediator**), and the other collaborators are defined as the **passive collaborators**. Table 3-3 shows the above relationship.

**Table 3-3. Relationship between three kinds of processes and their rank**

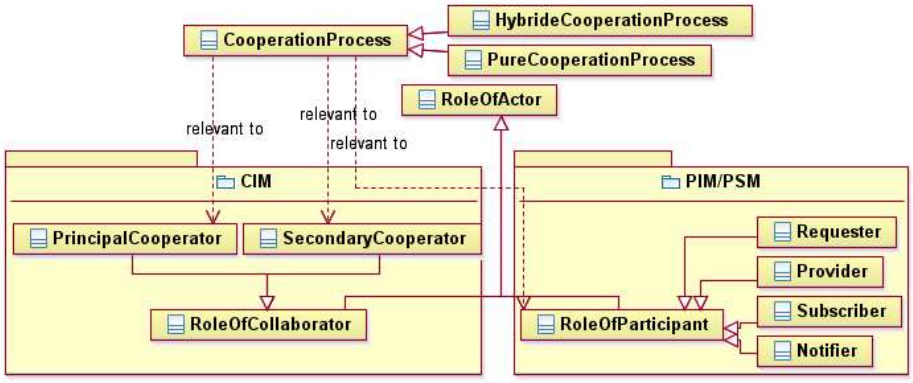
| Business Process | Rank | NCA |
|---|---|---|
| Internal Processus | R=1 | NCA=1 |
| Coordination Processus | R>1 | NCA=1 |
| Cooperation Processus | R>1 | NCA<=R but NCA>1 |

For a cooperation process, if NCA is also equal to R, the process is defined as a **pure cooperation process**. In fact, the pure cooperation process is the initial definition of cooperation process defined in Section I.2. If NCA is less than R, the cooperation process is defined as a **hybrid cooperation process**, where there are passive collaborators. In a pure cooperation process, if a cooperator not only controls its own activities but also controls the execution of the process, the cooperator is defined as **principal cooperator**. If a cooperator does not control the execution of the process but it controls its own activities, it is defined as a **secondary cooperator.** A secondary cooperator is different from a passive collaborator

because the passive collaborator controls neither the execution of the process nor its own activities: a passive collaborator only provides services for others. The **N**umber of **C**ontrollers for **P**rocesses is noted as **NCP** (evidently **NCP <= NCA**). In order to make the above concepts easily understood, Figure 3-3 provides the conceptual models for coordination and cooperation processes. In addition, the requester, provider, subscriber and notifier are defined in (Berre, Hahn et al. 2004) and (OMG 2006).




(a) Conceptual model for coordination processes




(b) Conceptual model for cooperation processes

**Figure 3-3. Conceptual model for coordination and cooperation processes**

### I.4. Cooperation rate

In a cooperation process, we define a concept "cooperation rate" as:

$$\text{Cooperation rate} = \frac{\text{occurrence number for activities of a cooperator}}{\text{number of all the activities}}$$

The concept indicates to which extent a cooperator participates in a cooperation process. It will be used during decomposition of a collaborative process in Section II.1.

## II. PROCESS BASED METHOD FOR ENTERPRISE INTEROPERABILITY

In order to realize the framework for IT solutions to enterprise interoperability framework, we propose the following method to solve interoperability problems, illustrated in Figure 3-4. PBMEI describes a transformation method from a collaborative process to a set of interoperability processes. This method starts from modeling collaboration requirements between enterprises with collaborative process. After several steps of transformation, it ends up with executable interoperability processes. It is a method in the modeling environment.

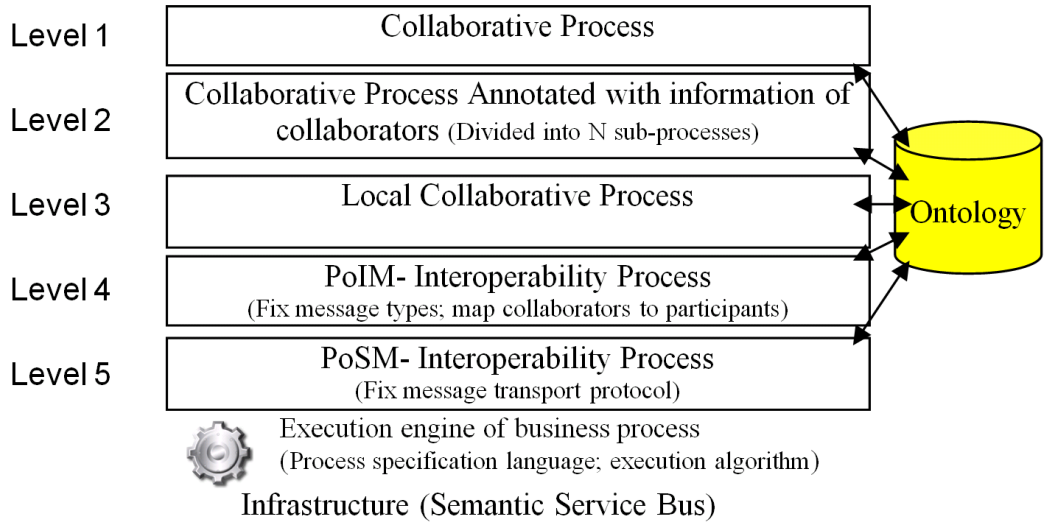



**Figure 3-4. Process-Based Method for Enterprise Interoperability**

There are 5 levels (steps) in this method:

1) The first level defines a collaborative process;
2) At the second level, the activities in the process are annotated with information about collaborators. After that, we merge the adjacent activities which belong to the same collaborators, and then we integrate the activities which belong to different collaborators to generate a simplified global process and several sub-processes;
3) At the third level, each collaborator transforms the (sub-)collaborative processes to its own collaborative processes based on its own business terminology definition;
4) At the fourth level PoIM (**Pro**tocol **I**ndependent **M**odel), each collaborator fixes data types for all the messages in its own processes and transforms collaborators at CIM level to participants at PIM/PSM level; at last, each collaborator will obtain its own interoperability processes.

5) At the fifth level PoSM (**Pro**tocol **S**pecific **M**odel), the interoperability processes are implemented in an executable specification language of business processes. At this level, message transport protocol must be fixed. When executing interoperability processes, all the collaborators must respect the same execution algorithm.

The above five levels all depend closely on ontology. Model transformations between the adjacent levels, except the model transformation between Level 1 and Level 2, will also depend on ontology. The two kinds of dependency relationships will be discussed in detail in Chapter 4. Instead, in this chapter, we will focus on the transformation between Level 1 and Level 2. The transformation between Level 1 and Level 2 is about decomposition of a collaborative process. The decomposition is very important because after the decomposition, the generated sub-processes will be reusable in other collaborations. In addition, the decomposition will reduce the number of messages delivered between enterprises.

### II.1. Decomposition of collaborative business process

Before continuing our study, we encounter the first problem: how to express collaborative business processes. As the development mode of information systems has been shifted from “programming” to “assembly” and from “data-centric” to “process-oriented” (Dumas, van der Aalst et al. 2005), hence, business process will be a trend for information (software) system development. In addition, according to the evolutionary history of business processes in Chapter 2-Section I.1.a, BPMN is the trend in business process research domain. Meanwhile BPMN itself has some advantages (elaborated in Chapter 2-Section I.1.c), so in our study about enterprise interoperability, collaborated business process will be expressed in BPMN. Furthermore, as described in Chapter 2-Section I, BPMN can also be regarded as an executable specification language for business processes. That is to say, interoperability processes in PBMEI will also be expressed in BPMN.

The second problem encountered is how to construct collaborative process at the first level in PBMEI. Rajsiri and his colleagues have proposed a semi-automated way to construct collaborative process at CIM level (Rajsiri, Lorré et al. 2008; Rajsiri, Lorre et al. 2009). The semi-automated way starts from a collaboration knowledge base and at last it will generate a BPMN collaborative process. This method is supported by a prototype. However, this method has some limitations: 1) the method needs human’s efforts, such as creation of collaboration knowledge base and verification of generated collaborative process; 2) the generated

collaborative process is possibly not valid. So in this thesis, we prefer manual creation of collaborative processes at the first level of PBMEI.

In the following subsections, a method to decompose a collaborative process is proposed. This method is based on annotated BPMN diagrams.

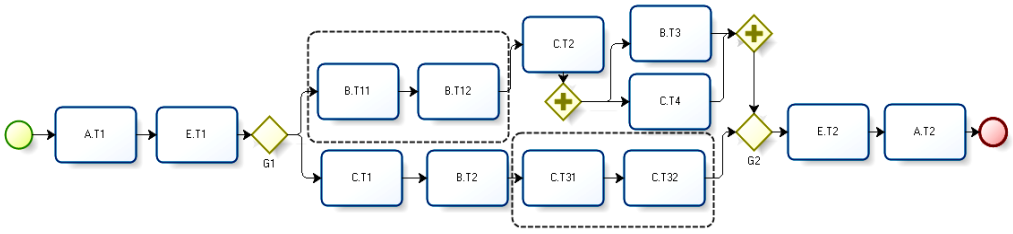
As the execution of internal business processes or coordination business processes is controlled by only one enterprise, and it is only related with a series of information exchanges between their partners, so such processes can be implemented with the help of WS-BPEL or workflow models. Instead, in the following subsections, collaborative processes will focus more on cooperation processes. In addition, in order to simplify our discussion, we assume that: collaborative processes in the following subsections only contain their business flow (data flow is omitted).

### II.1.a. Decomposition of a collaborative business process

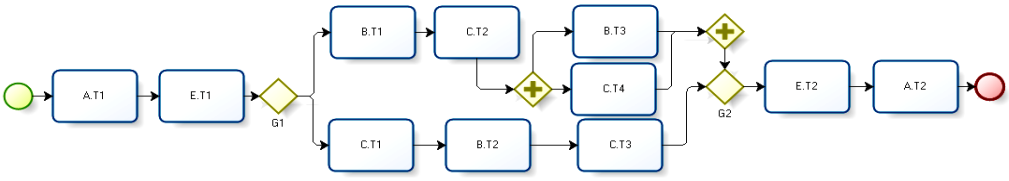
In a cooperation process (a collaborative business process), it is assumed that

1) there are N cooperators in the process, N>=2;
2) if the process is launched, all the cooperators will follow the process to carry out corresponding activities, so all cooperators must know clearly the state of the process execution;
3) the adjacent activities that belong to the same cooperator can be merged into one activity node who delegates a sub-process for the corresponding cooperator.

Following the above assumptions, the cooperation process can be changed into a new process with “sub-process” nodes and the new process has the following property: in the process, **each two neighboring activity nodes belong to different cooperators**. A transformation example is given in Figure 3-5.b which is obtained from Figure 3-5.a by merging B.T11 and B.T12 into B.T1 and merging C.T31 and C.T32 into C.T3. In Figure 3-5, the name of each activity has the format X.YZ, where X indicates the owner of the activity, Y indicates the activity type and Z is the identifier of the activity.




(a) A collaborative process with the adjacent activities that belong to the same cooperator




(b) An collaborative process without any two adjacent activities that belong to the same cooperator

**Figure 3-5. Cooperation Processes in BPMN**

Then the following question will be more interesting: how to reduce the rank of the interoperability process? In Figure 3-5.b, its R is equal to 4 and the designer has defined A and B as principal cooperators, and C and E as secondary cooperators. The two branches of the gateway G1 are related with cooperators B and C. They can be replaced with two sub-processes (cf. Figure 3-6), and the 4 interoperability sub-processes of Figure 3-5.b (i.e. B.T1→C.T2→B.T3/C.T4, C.T1→B.T2→C.T3, A.T1→E.T1 and E.T2→A.T2) are replaced with B.P1, B.P2, A.P1 and A.P2, so the rank of the obtained process is 2 (cf. Figure 3-6). Finally, the cooperators A and E have two sub-processes corresponding to (A.T1→E.T1) and (E.T2→A.T2). Meanwhile, the cooperators B and C have two sub-processes (B.T1→C.T2→B.T3/C.T4) and (C.T1→B.T2→C.T3). The cooperators B and C have one internal process separately: (B.T11→B.T12) and (C.T31→C.T32). The cooperators A and B also have one cooperation process illustrated in Figure 3-6.

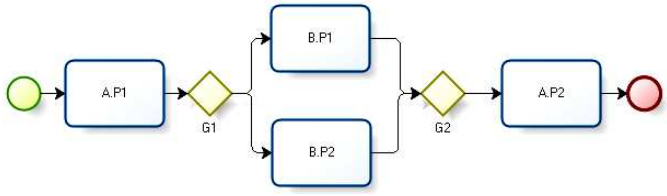



**Figure 3-6. Simplified cooperation process in BPMN**

The process transformation from Figure 3-5.a to Figure 3-5.b and then to Figure 3-6 must respect the following principles:

1) The rank of generated (target) collaborative processes must be less than that of the source collaborative process;
2) The rank of any new generated collaborative sub-process must be less than or equal to that of the target collaborative process;
3) The rank of the target collaborative process must be more or equal to 2.

4) If in a generated sub-process, there are several collaborators who influence the execution of the sub-process, the following criteria must be respected to choose one collaborator as the representative in the sub-process:
   - If the collaborators are already defined as a principal cooperators or secondary cooperators, the principal cooperator is selected as the representative ;
   - If there are several principal cooperators, the representative can be chosen by the sub-process.
   - If all the collaborators are the secondary cooperators, their cooperation rates will be compared. The cooperator whose cooperation rate is the greatest will be selected as the representative; if the cooperators have the same cooperation rate, the representative can be selected arbitrarily from the cooperators.

According to the transformation described previously, we can see that the global cooperation process (Figure 3-6) becomes simpler, and meanwhile new sub-processes are generated. The transformation simplifies the implementation of cooperation process, but meanwhile, it will increase the management complexity of the interoperability process because more collaborative processes will be transformed to interoperability processes. Are there other benefits brought from the transformation? To answer this question, we will analyze the execution of cooperation process between enterprises in the following section.

### II.1.b. Execution of interoperability process

To illustrate the execution of the cooperation process, consider the cooperation sub-process B.P1 in Figure 3-6, whose detail is given in Figure 3-7.

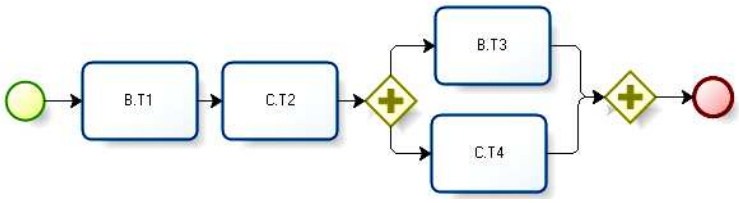



**Figure 3-7. Cooperation sub-process - B.P1**

As B.P1 is owned and controlled by cooperators B and C, this thesis offers the following execution process of the process B.P1:

1. When B.P1 is invoked, a participant, for example B, will create the instance of B.P1, and meanwhile it informs all the other cooperators (cooperator C) to create the instance of P1 in their own IS;

2. After all the cooperators have completed the instantiation of P1, then execute the following steps;
3. Each cooperator will check which cooperator executes the next activity, if B finds that it charge the execution of B.T1, then it will execute it and all the other cooperators (C) will wait for the notification from B;
4. If all the other cooperators (C) receive the notification from B, then each cooperator will check which cooperator executes the next activity, if C finds that it charges the execution of C.T2, then it will execute it and all other cooperators (B) will wait for a notification from C;
5. If all the other cooperators (B) have received the notification from C, then each cooperator will check which cooperator will execute the next activities (B.T3 and C.T4), if B finds that it will execute B.T3 and C finds that it will execute C.T4, then all the other cooperators (C and B) will wait for notifications from B and C;
6. If all the other cooperators (C and B) have received notifications from B and C, then the execution process ends.

As described earlier, the executions of the cooperation process at different cooperators are synchronized and collaborative. All the relevant cooperators follow the same method to execute the cooperation process, but in the IS of each cooperator, the execution behaviour is different. If any cooperator retreats from cooperation process or if any notification is not received by a target cooperator, the execution of the process will be blocked or abort. If any kind of failures comes out during the process execution, some measures must be taken to make the process execution recover from the failure or make the process execution stop elegantly. So, the execution engine of cooperation process should be based on distributed computing and message-oriented computing (Berre, Hahn et al. 2004).

In addition, as all cooperators have been determined before the design and implementation of cooperation process, the cooperation process can satisfy the requirements of "static" collaboration between enterprises, i.e., all the collaborators have the fixed relationship. If the collaboration is dynamic, i.e., some collaborators can often be replaced by other candidates, the cooperation process in this thesis is not able to meet such requirement directly. However, the cooperation process can be extended to support dynamic collaboration. Firstly, at the level of business modeling, the cooperation activity belongs to a role, not to a cooperator, and a role can have several cooperator candidates; secondly, during the execution

of the cooperation process, if a member quits, the execution promoter will choose another candidate whose role is the same as that of the quitting cooperator.

After the introduction of the interoperability process execution, we can see, the transformation which reduces the rank of the cooperation process can reduce many notifications between cooperators. This will be verified in a case study in Section II.2.

## II.2. Case study for decomposition of collaborative business process

The decomposition and the execution of cooperation process will be explained further by the example ShoppingDrive. In this example, we will model a cooperation process and then apply onto it the transformation (decomposition) method whose principles have been presented in Section II.1.a.

ShoppingDrive is an online shopping solution. Access to its website, and then choose products with the same prices as those offered in the real shop “Shopping”. We can then fetch the chosen products from ShoppingDrive very quickly. The concrete collaborative business process is shown in Figure 3-8. In the process, there are three cooperators: the Client, the CS (central server of “Shopping”) and Drive (ShoppingDrive). The “Client” can be considered as a cooperator (who logins the website (CS), submit its order, and pay by its bank card). The Drive is also a cooperator. In the process, we focus on cooperators. However, bank is a passive collaborator (just providing financial service), so it is omitted.

Before analyzing the process, we define the CS and the Drive as the principal cooperators and the client as a secondary cooperator. By traversing the process, we can calculate the rank of the process which is equal to 3. If we reduce the rank, the rank of each new generated sub-process and the rank of the target (global) process must be 2 according to the third principle in Section II.1.a.

In the process, before the “fork parallel gateway”, the rank is 2 and after the “join parallel gateway”, the rank is 3. Hence, our method regards the part before the “fork parallel gateway” as a sub-process: it is the sub-process {CS, Client}.P1 (Figure 3-9.a). Here, $\{x_i\}$.PID means that each element $x_i$ influences the execution of the process PID.

For the part between the two “parallel gateways”, the rank is 3, but for each branch, the rank is 2. Therefore, our method generates another two sub-processes: {CS, Drive}.P2 (Figure 3-9.b) and {CS, Client}.P3 (Figure 3-9.c).

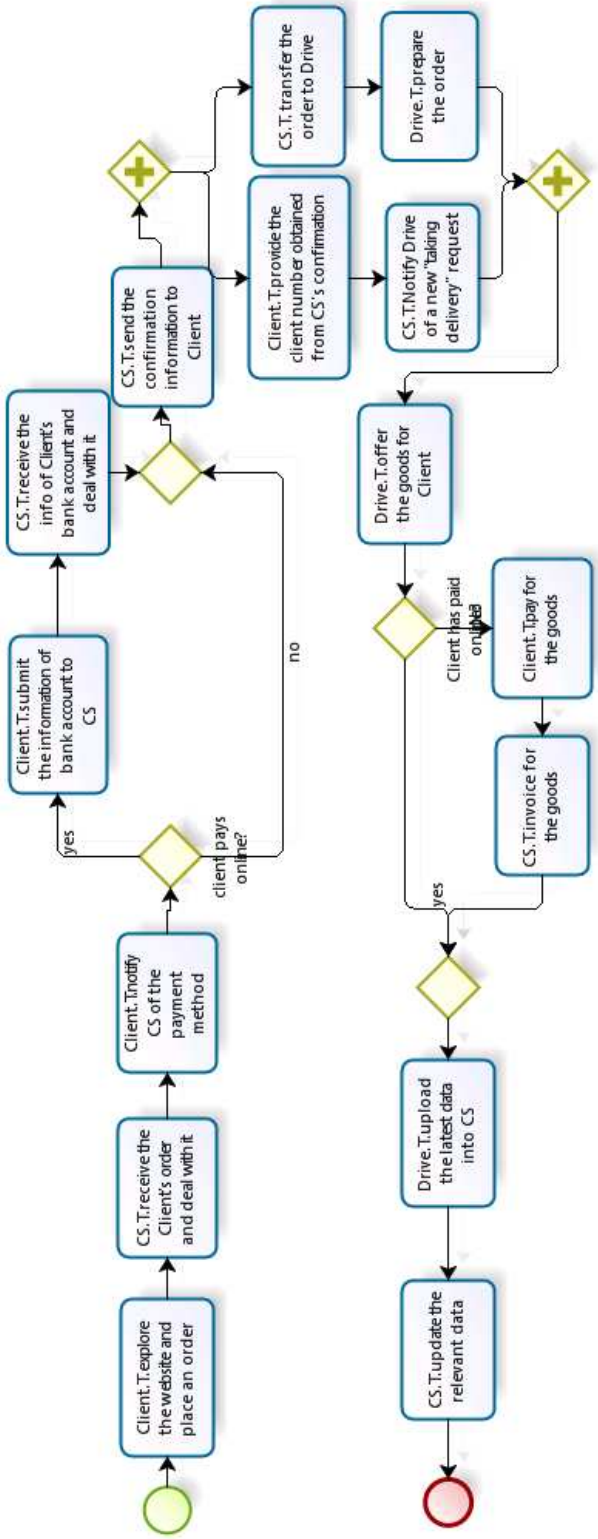
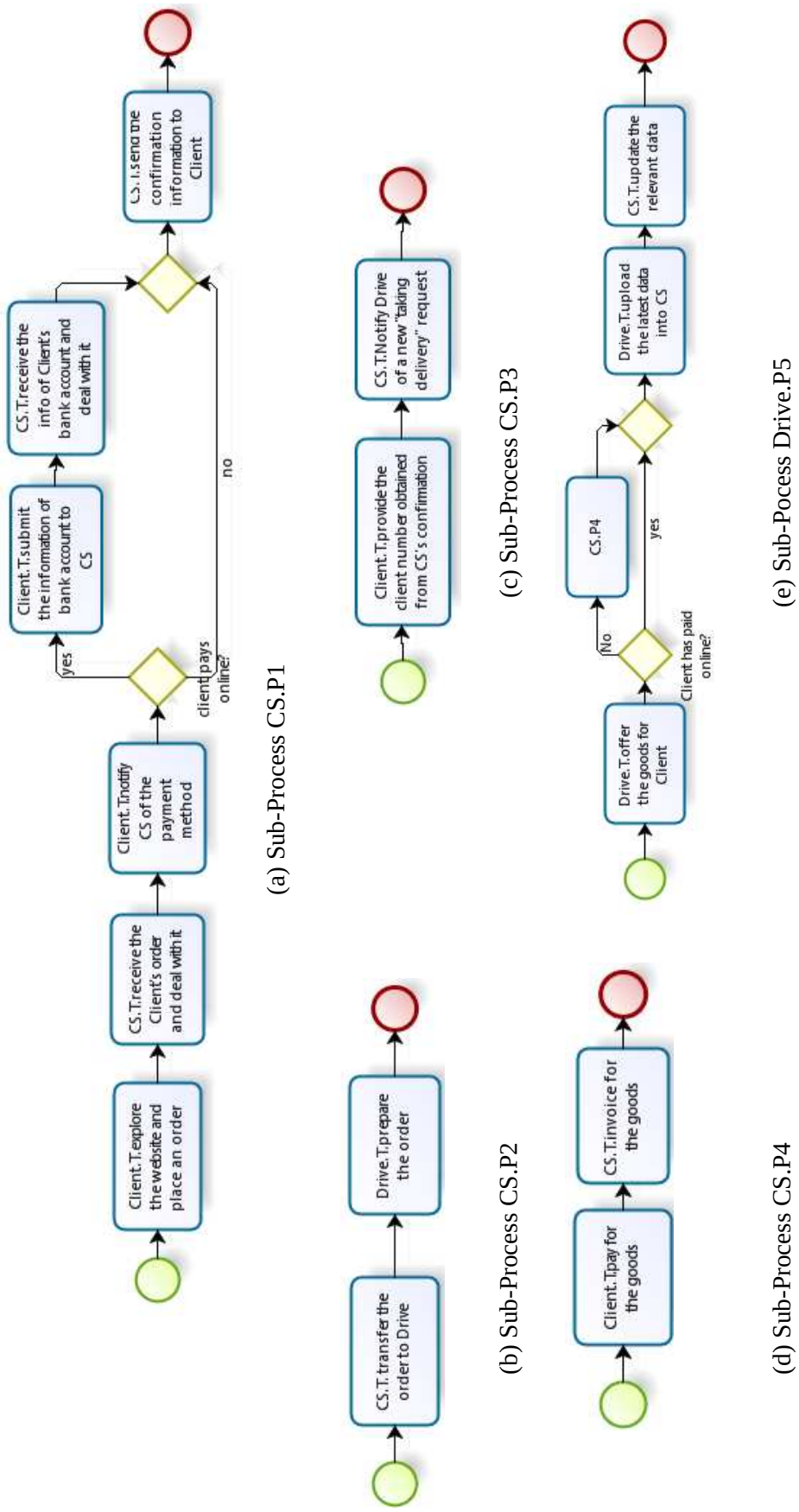



**Figure 3-8. Cooperation process for ShoppingDrive**




(a) Sub-Process CS.P1

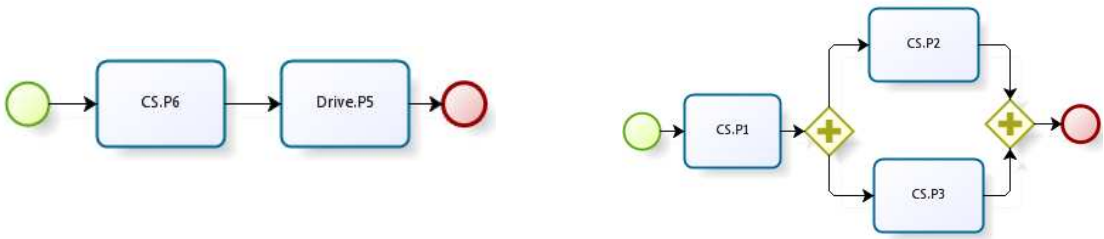
(b) Sub-Process CS.P2

(c) Sub-Process CS.P3

(d) Sub-Process CS.P4

(e) Sub-Pocess Drive.P5

**Figure 3-9. Cooperation Sub-Processes for ShoppingDrive**




(a) Global Process (b) Internal Process CS.P6

**Figure 3-10. Simplified Cooperation Process for ShoppingDrive**

For the rest of the process in Figure 3-8, the rank is equal to 3. According to the definition of the cooperation rate, the rates for Client, CS and Drive are 20%, 40% and 40%. So, it must integrate the Client's activities with the others' during generating new processes. At the same time, the "Client" is defined as a secondary cooperator. So, in the process, the activity "Client.Tpay for the goods" is integrated with "CS.Tinvoice for the goods". Before the first nearest gateway and after the second nearest gateway, the two activities belong to "Drive", not to CS. Therefore, the sub-process {CS, Client}.P4 (Figure 3-9.d) is generated. Then, the sub-process {CS, Drive}.P5 (Figure 3-9.e) is generated. The representatives for the five sub-processes are indicated in Figure 3-9 according to the fourth principle in Section II.1.a.

After generating the 5 sub-processes, the original process is transformed into the target global process illustrated in Figure 3-10.a by generating an internal process CS.P6 (Figure 3-10.b). Why do we generate the internal process CS.P6? Because the transformation must respect the third assumption at the beginning of Section II.1.a. Finally, the process in Figure 3-8 is transformed into a simple global process represented in Figure 3-10.a, whose rank is equal to 2, and for all the sub-processes their ranks are all equal to 2.

We analyze the quantity of messages sent between cooperators. In Figure 3-8, the rank of the process is 3, and each activity sends a message to the other two cooperators, so a total of 15 * 2 = 30 messages are sent out. To calculate the quantity of messages sent out by the simplified process, we must consider all sub-processes. In Figure 3-9.a, Figure 3-9.b and Figure 3-9.c, there are a total of 6 +2 +2 = 10 messages sent out, and in Figure 3-9.d, Figure 3-9.e, there are 2+4 = 6 messages sent out. In Figure 3-10.b, there is no message sent out (internal process in the enterprise), and in Figure 3-10.a there are 2 messages sent out.

In all, for the simplified process Figure 3-10.a, there are 18 messages sent during its execution. The above calculation method does not consider messages that are irrelevant to business transactions. We can see that after the process transformation from Figure 3-8 to Figure 3-10.a, the number of business messages has already been reduced.

## III. RELATED WORK

Different methods have been proposed to solve interoperability problems, especially for enterprise collaboration, such as, WISE (Alonso, Fiedler et al. 1999), ebXML[30], XLANG (Thatte 2001), WSCL[31] etc. However, many existing approaches are not flexible (Chebbi, Dustdar et al. 2006). In addition, almost all solutions suppose the homogeneity of data structures and business logic between different participants (Chebbi, Dustdar et al. 2006). In these solutions, the inter-visibility of internal processes of organizations is not well controlled: internal processes of organizations are either completely hidden or completely open (Chebbi, Dustdar et al. 2006). (Chebbi, Dustdar et al. 2006) has proposed an approach for dynamic inter-organizational workflow cooperation, and this approach can resolve the above three problems: flexibility, heterogeneity and inter-visibility.

In this section, we compare ebXML and the approach of Chebbi with our method.

### III.1. ebXML

The goal of ebXML is to provide an XML-based framework to enable XML to be utilized in a coherent and uniform manner for exchange of electronic business data in order to create a single global electronic market (ebXML 2001). The framework of ebXML is covered by a set of specifications, which are “core component technical specification”, “Registry Service” (OASIS-ebXML 2002b), “Business Process Specification Schema” (BPSS) (OASIS-ebXML 2006) and “Collaboration Protocol Profile and Agreement” (CPP&CPA) (OASIS-ebXML 2002a) etc.

The cooperation process in our method PBMEI is similar with ebXML BPSS but different. The similarity is that in collaborative process, all cooperators execute one identical copy of cooperation process. The primary difference is located at the execution level of cooperation process: all cooperators know execution state of the other cooperators but in the execution of BPSS, just the directly associated cooperators know mutually their execution state (OASIS-ebXML 2001) and the direct collaboration relationship is determined by the CPA which is generated from the CPPs of cooperators.

### III.2. Approach of Chebbi

The objective of the approach of Chebbi is to provide support for organizations which are involved in a shared but not pre-modeled cooperative workflow across organizational

[30] http://ebxml.org

[31] http://www.w3.org/TR/wscl10/

boundaries (Chebbi, Dustdar et al. 2006). The approach is inspired by SOA and it contains three steps: workflow advertisement, workflow interconnection and workflow cooperation. The approach depends on the transformation from an internal process to a cooperative process and then to public processes.

This approach supposes that interactions between workflows in virtual organizations cannot be specified before. However, in our method, enterprise collaboration must be described at the beginning of modeling phase, and this situation is also supported by (Van der Aalst 1999)[32]. In addition, in this approach, all enterprises execute their own workflows and send messages if necessary; however, in our method PBMEI, all enterprises execute the same interoperability process and send out messages.

## IV. CONCLUSIONS

In this chapter, we have proposed and elaborated the Process-Based Method for Enterprise Interoperability (PBMEI). As PBMEI is process-based, it makes relevant enterprises more responsive to collaboration requirement changes.

In this chapter, before elaboration of PBMEI, we have defined some key concepts widely used in the thesis and meanwhile we have also positioned the concepts in the MDA framework. Consequently, we discussed why we select BPMN as the representation language for collaborative business processes. With the help of BPMN, we have presented the decomposition method for a collaborative process. The decomposition method is based on two quantitative criteria: the rank of collaborative process and the cooperation rate. This method allows making information exchanges between hierarchical processes. To explain the transformation method, we have taken a collaborative process of the enterprise "ShoppingDrive" as an example. The case study about "ShoppingDrive" has indicated that the decomposition of a collaborative business process will reduce the number of exchanged messages during the execution of the process.

In the next chapter, we will research how ontology influences our proposed method. We will analyze the ontology content and then we will study how to do business process transformation with the help of ontology. The ontology-based business process transformation will be used between Level 2, Level 3, Level 4 and Level 5.

[32] (van der Aalst 1999) says that "there are numerous situations where the organizations participating in a shared workflow process feel the need to specify the coordination structure explicitly".

# CHAPTER 4: Ontology-based PBMEI and its Model Transformation

The "Process-Based Method for Enterprise Interoperability" (PBMEI) proposed in Chapter 3 is a method in modeling environment and it is inspired by MDA and ontology. However, in Chapter 3, the use of ontology in PBMEI hasn't been studied. The dependent relationship between PBMEI and ontology will be discussed in this chapter. Section I shows how ontology influences each level in PBMEI. It will discuss two variants of PBMEI thanks to different uses of ontology. This section will also present the concrete content in the dependent ontology of PBMEI. Section II will study the relationship between ontology and collaborative processes in PBMEI. This section will propose four ontology-based methods for semantic annotations in BPMN-based collaborative processes. Semantic annotations in collaborative processes will be beneficial to process transformations in PBMEI, which will be discussed in Section III.

# I. ONTOLOGY-BASED PBMEI

In order to solve enterprise interoperability problems, a "Process-Based Method for Enterprise Interoperability" (PBMEI) has been proposed in Chapter 3. In PBMEI, business requirements about enterprise interoperability are represented in collaborative processes among which the enterprises involved. As elaborated in Chapter 3-Section II.1, our method employs the service-related process specification language BPMN (OMG 2011) to describe collaborative processes. The collaborative process in PBMEI will finally be achieved through interoperability processes which are still expressed in BPMN. In this section, we will discuss how to apply ontology to PBMEI: ontology-based PBMEI.

## I.1. Ontology-based PBMEI

The ontology-based PBMEI is illustrated in Figure 4-1. This section will study how ontology influences each level of PBMEI.

At the first level, collaborative process must be defined from two aspects: business flow and data/message flow, which is inspired by the article (Hamilton and Catania 2003). According to (Hamilton and Catania 2003), US Army proposed an expansion of the system architecture into three further sub-architectures: software architecture, data architecture and network architecture. Software architecture defines the functionality of each modular, and data architecture is related to data definition, and network architecture is related to software deployment requirement. Furthermore, all business requirements must be mapped into system architecture to be implemented. Collaborative process is one kind of business requirements, so collaborative process must also be mapped into the above three sub-architectures, that is to

say collaborative process must have some aspects that can be mapped into the above three sub-architectures. However, the network sub-architecture is determined by concrete business requirements and related to the whole system, so this thesis will consider this problem in Section I.3 from overall point of view, not in collaborative processes. Finally, collaborative process will be constructed from two aspects: functionality and data. The business flow describes the functionality of the collaborative process; and the data flow describes the data exchanged in the process.

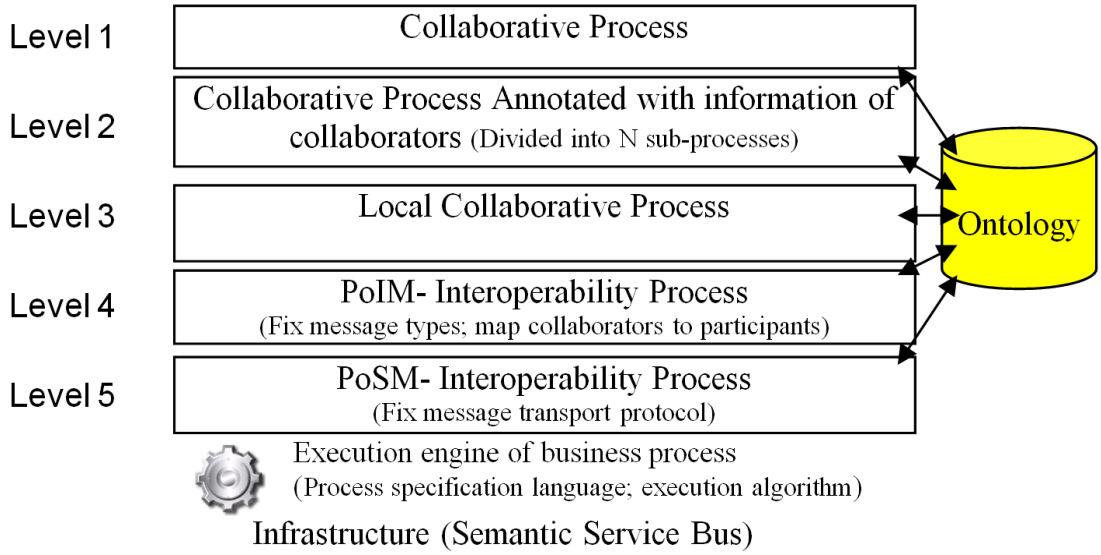



**Figure 4-1. Ontology-based and Process-Based Method for Enterprise Interoperability**

At the second level, collaborative process will be annotated with collaborators' information, i.e., each activity in collaborative process must be charged by one collaborator. This task depends on ontology. When searching the relevant collaborators for an activity, ontology will be inspected to determine which collaborator can do such activity. If several candidates are selected, the target candidate will be selected according to collaboration policy/requirements, or according to predefined conditions, such as QoS, trust rank/belief value etc. So ontology must contain such information about all collaborators (such as, collaborator's name, historical information about service running, responsibility, etc). After being annotated with information of collaborators, the collaborative process will be transformed into a set of collaborative sub-processes according to a transformation method.

At the third level, collaborative processes (including the generated collaborative sub-processes) will be transformed into local collaborative processes by each collaborator. During this transformation, the business terminologies will be transformed from global to local terminologies and the process specification language will also be transformed from global to local if necessary.

At the fourth level "PoIM" (**Pro**tocol **I**ndependent **M**odel), message types in collaborative process must be determined according to messages context (messages sender and receiver, and relevant business context). Some messages types may also be partially declared in collaboration requirement. The above two cases of message type determination are ontology-based. So, the ontology must contain business messages definitions, which may have some business context specifications.

After the determination of messages types, collaborators in a collaborative process will be mapped into participants. The key of the mapping focuses on functionality and context of an activity. After the mapping from collaborators to participants, the collaborator's information in the process must also be kept, because such information has semantics that is not implied in participants. For example, semantics for roles of collaborators cannot be represented by participant roles. The above tasks at this level also rely on ontology. As a participant is an element of system architecture, ontology must also contain information about each collaborator's system architecture. At last, after message types are fixed and the mapping from collaborator to participant is done, collaborative processes become interoperability processes.

At the fifth level "PoSM" (**Pro**tocol **S**pecific **M**odel), interoperability process will be implemented in an executable process specification language: BPMN. All message transport protocols are explicitly specified at this level.

According to the above description, PBMEI closely depends on ontology and SOA. It also has one assumption: interoperability process totally depends on original functions of each collaborator's information system. Of course, PBMEI also relies on a process execution engine and a given infrastructure, such as semantic service bus. Semantic service bus will be studied in Chapter 5.

## I.2. Two variants of PBMEI

In Chapter 3-Section I.2, collaborative business processes are classified into three types: internal process, coordination process, and cooperation process. As internal process and coordination process can be easy to implement with the help of WS-BPEL or workflow model, this thesis will focus on cooperation process in PBMEI.

When using PBMEI to solve interoperability problems, the first encountered problem is: who will create cooperation process and in which style? In practice, if there is a core cooperator, the cooperation (collaborative) process is created by the core cooperator. It will not negotiate with any others. If there is no core cooperator, the cooperation (collaborative)

process is created through negotiation of all cooperators. When applying PBMEI into the above two cases, two variants of PBMEI are generated and they are described below.

I.2.a. Ontology-based PBMEI for collaboration without core cooperator

If a collaborative process has no core cooperator, PBMEI becomes the following variant, see Figure 4-2. At Level 1, all collaborators negotiate to create a collaborative process. At Level 2, the collaborative process is annotated with collaborators' information and divided into several sub-processes. The first and second levels and the transformation between them are global, which depends on the global ontology. The third, fourth and fifth levels and the transformations between them are done locally by each separate collaborator, which depends on the local ontology. After all collaborators generate their own interoperability processes, they can execute them through an identical execution algorithm.

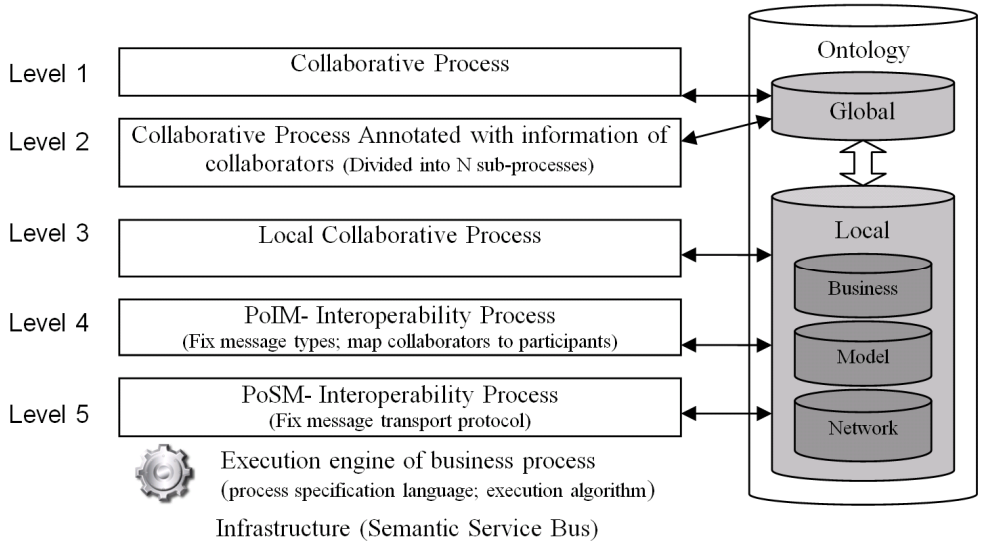



**Figure 4-2. Ontology-based PBMEI for collaboration without core cooperator**

The global ontology includes common sense necessary when collaborators negotiate with each other to create collaborative process. The global ontology will also define the syntax and semantics of collaborative process. The business expressions in all elements of collaborative process must also respect business terminology definitions in the global ontology. The global ontology must also contain collaborators' information because which is needed when collaborative process is annotated.

The local ontology contains all information about enterprise architecture for a corresponding collaborator. It includes three basic ontologies: business ontology, model ontology and network ontology. Business ontology contains all terminologies related with local business requirements. Model ontology contains all models (architecture models and

data models) in different software development phases. Network ontology contains information about software deployment.

Of course, there must be mappings between the global ontology and the local ontologies for all collaborators and such mappings will be used by each collaborator when they transform the global collaborative process into their own collaborative process. The mapping between the global and local ontology will be stored and maintained in local ontologies.

### I.2.b. Ontology-based PBMEI for collaboration with core cooperator

If a collaborative process in PBMEI has a core cooperator, once the core cooperator finishes Level 1 and Level 2 in Figure 4-3, it will deliver them to its collaborators, and the other collaborators will transform the received processes into processes described in their own languages. For the core cooperator, it will follow Level 3 and Level 4 in the PBMEI described in Figure 4-3. For all other collaborators, they will follow Level 3 to Level 5 in the PBMEI described in Figure 4-2.

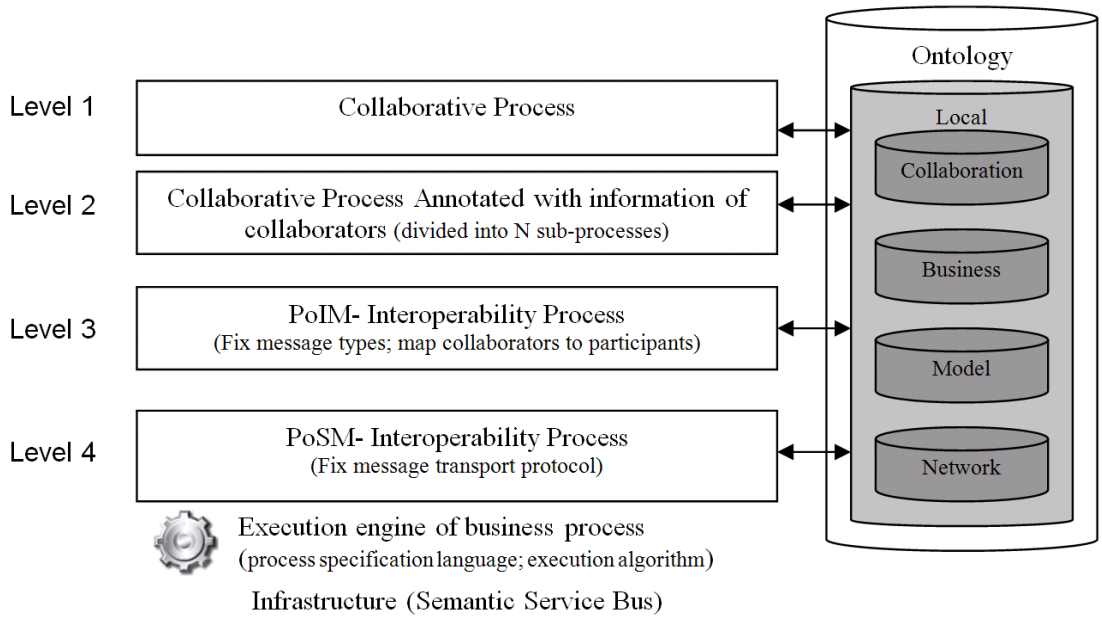



**Figure 4-3. Ontology-based PBMEI for collaboration with core cooperator**

Since collaborative process is created by the core cooperator, the process is only based on the core collaborator's ontology, and there is no need to transform the global collaborative process into local collaborative process, which is why the variant depicted in Figure 4-3 does not have the level "Local collaborative process".

Note that besides the business ontology, model ontology and network ontology, the ontology of the core cooperator also contains the collaboration ontology which offers information about collaborators and their services.

## I.3. Content of ontologies in PBMEI

As Section I.1 proposes suggestions about content of ontologies in PBMEI, and Section I.2 provides the categories of ontologies in PBMEI, this section will present which ontology should contain what. Table 4-1 gives a proposal. The construction of Table 4-1 is also based on the study in Chapter 2-Section I (about business processes and relevant tools) and in Appendix A (about business rules).

**Table 4-1. Content of ontologies in PBMEI**

| Ontology | | Content | Mapping |
|---|---|---|---|
| **Global** | | • information about all the collaborators: name, business roles, and postal address, email address, network address, offered business services, published web services and related statistical information about their offered services (e.g., QoS, trust rank), etc<br>• common business object model<br>• specification language for collaborative processes and business policies<br>• collaborative process, collaboration policies | |
| **Local** | **Collaboration** | • information about all the collaborators: name, business roles, and postal address, email address, network address, offered business services, published web services and related statistical information about their offered services (e.g., QoS, trust rank), etc<br>• common business object model<br>• collaborative process, collaboration policies | |
| | **Business** | • all business concepts in its own domain and relationships<br>• local business policies<br>• local collaborative process<br>• organizational information | Mapping to global or collaboration ontology;<br>Mapping to model ontology; |
| | **Model** | • formal descriptions of business concepts, their relationships<br>• formal descriptions of business rules and technical rules<br>• formal descriptions at different levels about the architecture of an enterprise information system<br>• formal descriptions about all physical components of enterprise software systems<br>• Interoperability processes at “PoIM” and “PoSM” levels | Mapping to business ontology;<br>Mapping to network ontology; |
| | **Network** | • deployment information of all software components of an enterprise information system | Mapping to model ontology |

In Table 4-1, there are mappings between global ontology/collaboration ontology and business ontology, between business ontology and model ontology and between model ontology and network ontology. These mappings should be maintained and managed and they will be used when transforming business process (see Section III).

According to Table 4-1, the ontology contains information about collaboration, business, model and deployment. In fact, data storage in ontology can be real or virtual. That is to say, data can be directly stored in the ontology, but they can also be stored in remote professional servers. In the latter case, the ontology only stores ontology-based description about remote data. For example, business rules can be stored in Business Rule Management System (BRMS) (Graham 2005) and the ontology only contains ontology-based descriptions about business rules.

### I.4. Conclusions

This section has presented how ontology influences each step of PBMEI. Because of different uses of ontology, two variants of PBMEI have been analyzed. During analysis of the two variants, categorization of ontology in PBMEI has been proposed: global ontology, local ontology, collaboration ontology, business ontology, model ontology and network ontology. At last, content for each ontology has been concluded according to studies about PBMEI, business processes and business rules.

This section has primarily studied the relationship between ontology and PBMEI. In the following two sections, we will discuss the relationship between ontology and collaborative processes in PBMEI and study how to use the ontology-based relationship to do process transformation.

## II. ONTOLOGY-BASED ANNOTATION FOR COLLABORATIVE BUSINESS PROCESS

### II.1. Literature study

In order to research semantic information of business processes, two aspects should be considered: which kind of information should be ontologized and how to represent the information. For the first aspect, the work of (Filipowska, Hepp et al. ; Filipowska, Kaczmarek et al.), which is based on European SUPER project (Semantics Utilised for Process management within and between EnteRprise) [33], has proposed three kinds of

[33] http://www.ip-super.org/

ontologies: process ontology, organisational ontology and domain ontology. Process ontology describes the structure of business processes whereas organisation ontology describes the artifacts involved in business processes (such as actors, resources etc), and domain ontology provides information specific to a company. (Filipowska, Hepp et al. 2009) also shows that the three kinds of ontologies have different contents in different phases of BPM lifecycle.

For the second aspect, we have two choices: firstly, represent whole business processes as ontologies including structures and contents of business processes; secondly, add semantic annotations for contents of business processes. (Lin and Ding 2005) has proposed a General Process Ontology and an application domain ontology to ontologize the structure and content of business processes. In order to do the experiments of semantic process retrieval, (Kiefer, Bernstein et al.) has transformed approximately 5000 business processes into OWL described by the concepts of MIT Process Handbook[34]. (SUPER-Project) has proposed semantic BPMN which constructs BPMN concepts in OWL and uses these definitions to instantiate BPMN processes. (SUPER-Project) has also proposed semantic BPEL (sBPEL), semantic Event Process Chain (sEPC) to describe business processes and it wants to transform business processes based on these ontologies into that based on BPMO and at last it hopes BPMO can bridge sBPMN, sEPC and sBPEL together. To achieve the goal, (Norton, Cabral et al.) has done the ontology-based translation of business process models from Business Process Modeling Ontology (BPMO) to sBPEL and from sBPEL to BPMO. This thesis will discuss the second choice, like SAWSDL[35] realized by WSMO Studio[36].

## II.2. Semantic Annotations for Business Processes in BPMN

In this thesis, semantic annotations for business processes are based on ontologies, i.e., the annotations will refer to concepts, properties or instances in ontologies (shown in Figure 4-4). However, the construction and distribution of ontologies are beyond the scope of this thesis, so this thesis will just focus on how to associate ontology with BPMN2.0-based business processes. Before that, we provide a concrete example that indicates why a semantic annotation is necessary to business processes: in a company, for the preparation of an anniversary celebration, there are numerous tasks to do, one of which is to buy 5 beautiful notebooks as awards. To the organisers of the preparation activity, “notebook” maybe means

[34] http://ccs.mit.edu/ph/

[35] http://www.w3.org/TR/sawsdl/

[36] http://www.wsmostudio.org

"book with blank pages for recording notes or memoranda"[37]. However, if the preparation process of the celebration is supported by information systems and the task, "buy 5 notebooks", is implemented by IT engineers as "find a notebook provider on the Internet and send electronic request", to IT engineers, "notebook" may be "notebook computer (a small compact portable computer)"[37]. That is to say "notebook" has ambiguity in the "preparation" process. However, this is just one case for semantic heterogeneity (Xu and Lee 2002; Wang and Liu 2009). So contents in business processes must be annotated with semantic information for disambiguation between different people. The following will explain how to realize semantic annotations for BPMN2.0-based business processes.

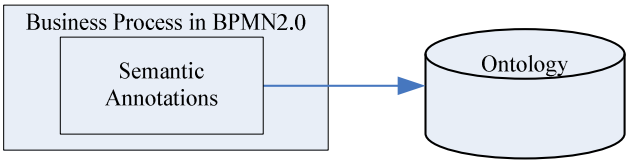



**Figure 4-4. Ontology-based Semantic Annotations for Business Processes**

BPMN2.0 metamodel provides an **extension mechanism** (see Figure 4-5). This allows business process metamodel to be extended but to be still BPMN-compliant. In BPMN2.0 metamodel, such extensibility is implied in the definitions of "baseElement" (Page 64 of (OMG 2011)), "rootElement" (Page 65 of (OMG 2011)), "documentation" (Page 64 of (OMG 2011)), and "extension" (Page 60 of (OMG 2011)). This section proposes the following four ontology-based methods of semantic annotations. Before the elaboration of the four methods, the outline of BPMN2.0 files is provided in Figure 4-6 (a). BPMN2.0 files are based on XML, and their root element is "definitions" (Page 54 of (OMG 2011)), and normally it contains two scopes: one for the structure of collaborations/choreographies/processes and the other for the visualization of all graphical notations in business collaborations/choreographies/ processes.

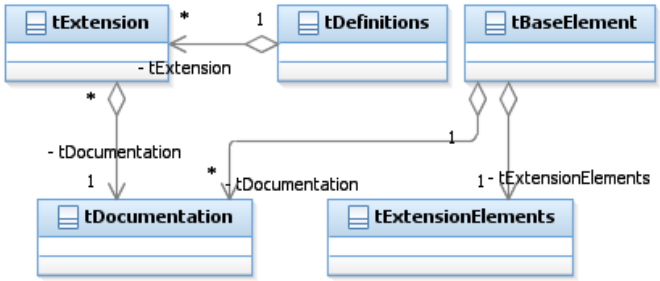



**Figure 4-5. Extensibility Model of BPMN2.0**

[37] http://wordnetweb.princeton.edu/perl/webwn

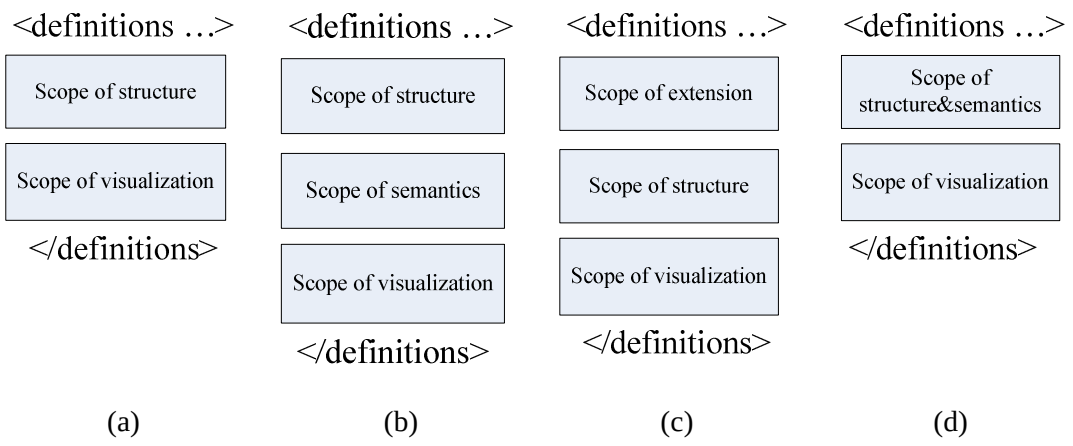



**Figure 4-6. Structures of BPMN2.0 Files**

### II.2.a. "rootElement"-based Semantic Annotation

According to BPMN2.0 metamodel, "rootElement" is a child element of "definitions" and it can be replaced by its subclasses, so we can define a subclass of rootElement's data type and create a corresponding element to replace "rootElement". Part of the schema definition for the scope of semantics is as follows. The complete schema definition is provided in Appendix D.

*Schema definition for semantic annotations of BPMN2.0*

```
<xs:complexType name="tSemanticAnnotation">
<xs:complexContent>
<xs:extension base="bpmn20:tRootElement">
<xs:sequence>
<xs:element name="detail" type="tSemanticDetail" minOccurs="0"
maxOccurs="1"/></xs:sequence>
<xs:attribute name="bpmnElement" type="xs:QName"/>
<xs:attribute name="ontologyRef" type="xs:anyURI"/>
<xs:attribute name="level" type="tMDALevel"/>
</xs:extension></xs:complexContent>
</xs:complexType>
<xs:complexType name="tSemanticAnnotationList">
<xs:complexContent>
<xs:extension base="bpmn20:tRootElement">
<xs:sequence>
<xs:element name="semanticAnnotation" type="tSemanticAnnotation"
minOccurs="0" maxOccurs="unbounded"/> </xs:sequence></xs:extension>
</xs:complexContent>
</xs:complexType>
```

```
    <xs:element name="semanticAnnotationList"
type="tSemanticAnnotationList"
substitutionGroup="bpmn20:rootElement"/>
```

In the above code, the type "**tSemanticAnnotation**" defines which attributes should be included in the semantic annotation for an element in BPMN2.0-based business processes. Its attribute "**bpmnElement**" points to a corresponding element in the scope of structure. The attribute "**ontologyRef**" points to a concept defined in an ontology and the concept explains what the above "bpmnElement" means. The attribute "**level**" means an MDA level at which the semantic annotation is. The sub-element "**detail**" contains the detailed semantic information of the annotated element and it can appear at most one time in a semantic annotation, for example, for a certain task in a business process, there is not any corresponding concept/instance in the dependent ontology, then the task can be described by its actors, action, resources and other conditions which may have corresponding concepts/instances. Besides, the list "**semanticAnnotationList**" contains all required semantic annotations for elements in business processes.

After applying the above schema into a business process in BPMN2.0, the BPMN2.0 files will be like Figure 4-6 (b). The following gives an example of the scope of semantics (the *namespace* in Italic is the namespace of the dependent ontology).

*Example of "rootElement"-based semantic annotation*

```
    <bpmnsa:semanticAnnotationList id="sid-1">
    <bpmnsa:semanticAnnotation id="sid-2_s" bpmnElement="sid-2"
ontologyRef="{namespace}/logisticOnto.owl#Notebook"
    level="CIM"/>
    </bpmnsa:semanticAnnotationList>
```

However, this method requires that the new schema and the original BPMN2.0 schema (metamodel) share the same "targetNamespace", and BPMN2.0 schema must include the new schema. That is to say the original BPMN2.0 schema will be modified, and this is the drawback of the method.

### II.2.b. "extension"-based Semantic Annotation

According to BPMN2.0 metamodel, "extension" is a sub-element of "definitions", and it can be extended. So this method is to add semantic annotations into the "extension" element. The definition of semantic annotations is the same as that in the first method. After applying this method into a business process, the BPMN2.0 files will be like Figure 4-6 (c).

The following gives an example of the scope of semantics. In this method, the scope of semantics is included in the scope of "extension", not directly stored as the sibling scope of business processes' structures like "rootElement"-based method, so the representation style of semantic annotations in this method is less clear than that in "rootElement"-based method.

*Example of "extension"-based semantic annotation*

```
<extension definition="semanticAnnotation">
<documentation>
<bpmnsa:semanticAnnotationList id="sid-1">
<bpmnsa:semanticAnnotation id="sid-2_s" bpmnElement="sid-2"
ontologyRef="{namespace}/logisticOnto.owl#Notebook"/>
</bpmnsa:semanticAnnotationList>
</documentation></extension>
```

### II.2.c. Attribute/Element-based Semantic Annotation

In BPMN2.0 metamodel, the type of "baseElement" makes it possible to add new attributes or new elements into it, and fortunately collaboration, choreography, process, task, artefact, event, message, gateway, participant and expression are extended based on the type of "baseElement", so all of the above concepts can add a new attribute to point to a concept defined in an ontology. The attribute can be defined as follows.

*Attribute definition for semantic annotation*

```
<xs:attribute name="ontologyRef" type="xs:anyURI" />
```

So after apply such annotation method, the extended business process is obtained and the following shows one fragment:

*Example of attribute-based semantic annotation*

```
<dataObject id="sid-2" isCollection="false" name="ticket"
bpmnsa:ontologyRef= "{namespace}/logisticOnto.owl#Notebook"/>
```

In this method, all semantic annotations are scattered in BPMN2.0 files, not like the above two methods where all semantic annotations are collected in one scope. The structure of BPMN2.0 files is like Figure 4-6 (d). This method is similar to SAWSDL.

### II.2.d. "textAnnotation"-based Semantic Annotation

As "textAnnotation" is extended from the type of "baseElement", so it has extensibility inherently. And "textAnnotation" can be associated with activities, events, gateways, message flows, sequence flows and other objects whose type is derived from the type of "baseElement". So "textAnnotation" can be used as a method of semantic annotations.

However, “textAnnotation” is associated with an annotated element by an association, not like the above three methods which associate semantic annotations and annotated elements by ID mappings (“bpmnElement” in Section II.2.a). In a business process, the usage of “textAnnotation”-based semantic annotation is as follows and the structure of BPMN2.0 files will be like Figure 4-6 (d).

*Example of “textAnnotation”-based semantic annotation*

```
<textAnnotation id="sid-3" textFormat="text/plain">
<text>
<bpmnsa:semanticAnnotation id="sid-2_s"
ontologyRef="{namespace}/logisticOnto.owl#Notebook"/>
</text>
</textAnnotation>
```

In terms of the above elaboration of four methods, the first two methods collect all semantic annotations together in the scope of semantics, instead the second two methods merge semantic annotations within the scope of business processes’ structures. Table 4-2 compares the four methods in detail.

**Table 4-2. Comparison between four semantic annotation methods of business processes**

| Semantic Annotation | advantages | disadvantages |
|---|---|---|
| “rootElement”-based | Keep all semantic annotations together; | Modify the meta-model of BPMN2.0; |
| “extension”-based | Keep all semantic annotations together; | Less clear than “rootElement”-based SA |
| attribute/element-based | Semantic annotations are attached directly to designated BPMN elements; | All semantic annotations are scattered in the structure scope of BPMN files; |
| “textAnnotation”-based | Semantic annotations are attached to designated; BPMN elements | Not directly mapped; “textAnnotation” appears everywhere in BPMN graphical diagrams; |

The above four ontology-based semantic annotation methods can be adopted by BPMN2.0 Tools such as BizAgi Xpress[38], Oracle BPM Suite[39], Bonita Open Solution[40] etc. According to Table 4-2, the second method is preferable. If BPMN2.0 tools want to add semantic annotations into business processes, they must provide a graphical user interface

[38] http://www.bizagi.com/index.php?option=com_content&view=article&id=19&Itemid=100

[39] http://www.oracle.com/us/corporate/press/079865

[40] http://www.bonitasoft.com/

(GUI), which could show all concepts/instances in dependent ontologies and which should also easily associate them with graphical elements in business processes. Of course, these tools should also provide a GUI for IT engineers to create detailed semantic annotations -- "detail" in Section II.2.a, which can help generate new concepts/instances in dependent ontologies.

### II.1. Conclusions

Business processes need semantic information during the alignment between business and IT. In order to supplement semantic information in BPMN2.0-based business processes, this section has presented four methods of ontology-based semantic annotations and these methods are all built on the existent extensibility mechanism of BPMN2.0. After the comparison of the four methods, the "extension"-based semantic annotation method will be preferable to the other three methods.

Apart from bringing benefits to BPMN2.0-based business processes, semantic annotations are also beneficial to ontologies. This section has indicated that the detailed semantic annotations will help to generate new concepts/instances to enhance contents of ontologies. Furthermore, semantic annotations imply the reversible associations between business processes and ontologies, hence some concepts/instances in ontologies have corresponding structural elements in business processes and they can find their preconditions/post-conditions through business processes. In fact, business processes can be regarded as contexts for some concepts/instances in ontologies. So, BPMN2.0-based business processes are one kind of structural annotations for ontologies.

## III. SEMANTIC ANNOTATIONS AND MODEL TRANSFORMATION

Besides facilitating process (or process fragment) discovery and reuse, semantic information in business processes can also help model transformation in MDA research domain. In Chapter 3, PBMEI has been proposed and it is integrated with MDA. At the CIM level, this method uses collaborative business processes to describe collaboration requirements between enterprises and after several model (business process) transformations, it is expected to generate several executable business processes. The business process transformations, especially between Level 2, Level 3, Level 4 and Level 5, will need semantic information retrieved from ontologies and add new information into generated business

processes. Semantic Annotations for business processes can be a suitable method to support such business process transformation.

Figure 4-7 shows a general model transformation in PBMEI and the transformation takes advantage of semantic annotations. In Figure 4-7, Business Process i has the existing semantic annotations which point to ontology, especially point to Ontology i, and the newly generated Business Process j contains new semantic annotations besides the originals. The new semantic annotations also point to ontology, especially points to Ontology j. During the transformation from Business Process i to j (from MDA high level to MDA low level), the mapping between Ontology i and j will be needed. With the help of the mapping, the transformation will find the concepts/instances in Ontology j corresponding to concepts/instances in existing semantic annotations of Business Process i. The new semantics will be added into Business Process j.

From the above description, semantic annotations of business processes are very useful for vertical model transformation (from MDA high level to low level).

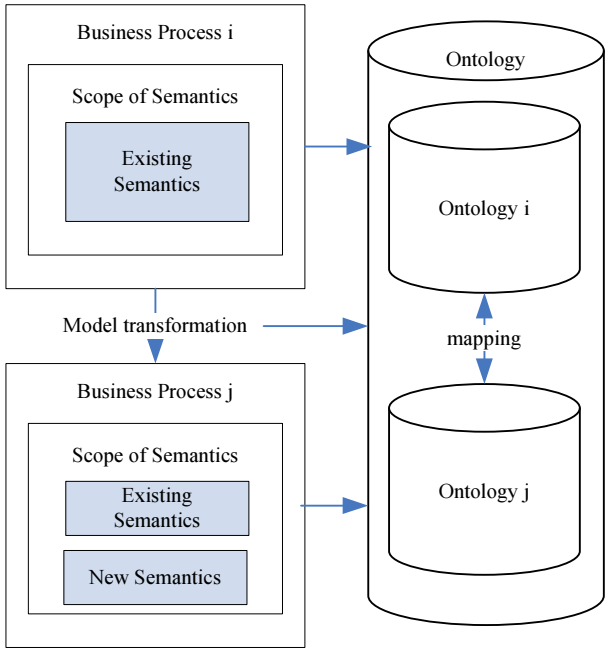



**Figure 4-7. Semantic Annotations in Business Process Transformation**

## IV. CONCLUSIONS

This chapter has presented some first developments about the method ontology-based PBMEI, which uses ontology in modeling environment to solve enterprise interoperability problems. This method also makes collaborators in collaborative process easily adapted to

collaboration requirement changes. After analyzing the dependant information in PBMEI and its two variants, the global ontology, local ontology, business ontology, model ontology and network ontology are introduced and their contents are presented.

In order to associate collaborative business processes in PBMEI with ontologies, four methods of ontology-based semantic annotations for business processes in BPMN2.0 have been proposed in this chapter. These methods are all built on the existent extensibility mechanism of BPMN2.0. After the comparison of the four methods, the "extension"-based semantic annotation method will be preferable to the other three methods. This paper has also shown that, in ontology-based PBMEI, semantic annotations are helpful to vertical transformation of collaborative business processes (a business process is regarded as a model).

According to the above discussion, ontology influences PBMEI from two aspects: representation of collaboration process and process transformation. In fact, ontology can not only influence vertical process transformation in PBMEI, but it can also influence horizontal transformation in PBMEI. In Section I.2, when we discuss two variants of PBMEI, the generated collaborative processes will be delivered to other collaborators. During this procedure, horizontal transformation of collaborative processes is necessary. How to support ontology-based horizontal transformation will be discussed in Chapter 5.

The research in this chapter is the foundation for further research on the way of using ontology-based PBMEI in a concrete application case. In a word, our method ontology-based PBMEI is ontology-based, process-based and model-driven and it is also ontology-language-independent.

# *CHAPTER 5: Goal-driven and Ontology-based architecture for enterprise interoperability*

Ontology-based PBMEI studied in Chapter 3 and 4, is designed to realize the modeling space of the framework illustrated in Figure 2-15 (Chapter 2-Section VI). In this chapter, we will propose an **O**ntology-**B**ased and **G**oal-**D**riven (**OBGD**) architecture for enterprise interoperability. This architecture is designed to realize the platform/infrastructure in the framework.

In order to design the OBGD architecture, Section I will analyze problems existing in semantic web services and in goal-based researches. Then in Section II, we will propose a goal model. Based on the goal model, we will design an OBGD SOAP and OBGD service invocation. Based on the research result in Section II, we will propose a symmetric mechanism for OBGD service invocation in Section III. The symmetric mechanism is implemented by OBGD semantic service bus, and the bus is the core of our proposed OBGD architecture. In Section III, we will study deployment methods of OBGD semantic service bus for intra- or inter- enterprise interoperability. In Section IV, we will study the relationship between this chapter and the above two chapters.

# I. LITERATURE STUDY

## I.1. Semantic Web Service

Semantic web service comes out from two domains: semantic web and web services (McIlraith, Son et al. 2001; McIlraith and Martin 2003). Originally, on the World Wide Web, there were only static contents (web pages). Most of the web contents were designed for humans to read, not for computer programs to manipulate meaningfully. So in 2001, semantic web was proposed by Tim Berners-Lee to structure the meaningful content of web pages and to create an environment where software agents roam from page to page (Tim, Hendler et al. 2001). Besides semantic extension of web, contents of web are also extended, from static web pages to web-accessible programs/sensors/devices, and such extension is realized by web services. Web services are supported by the specifications WSDL, SOAP, UDDI. These specifications are at the syntactic level without well-defined semantics. So they are obviously not powerful enough to support semantic interoperability of web services or automatic discovery, selection and composition of web services. So, like the shift from web to semantic web, in order to manipulate web services meaningfully, semantic extension of web services should also be defined. This is studied in the arena of semantic web services. There are two kinds of methods to extend web services semantically: one is to add semantic annotations into existing specifications, such as WSDL-S (W3C 2005), SAWSDL (semantic annotations for

WSDL) (Farrell and Lausen 2007) and SAWS (Salomie, Chifu et al. 2008); the other is to provide independent semantic descriptions for web services, such as OWL-S (The-OWL-Services-Coalition 2003), WSMO (Roman, Lausen et al. 2006).

OWL-S (formerly DAML-S (Ankolenkar, Burstein et al. 2001)) was proposed in order to support automatic discovery, invocation, composition and interoperation of semantic web services. It includes three main parts: service profile describing what a service does (web service capability), service model describing how a service works (web service programs) and service grounding describing how to access it (web service access) (McIlraith and Martin 2003). Execution semantics of OWL-S is formalized by different methods in different papers (Ankolekar, Huch et al. 2002; Narayanan and McIlraith 2002). OWL-S is implemented by a loose collection of individual tools like OWL-S editor, OWL-S matchmaker, OWL-S virtual machine, OWL-S IDE, etc (Shafiq, Moran et al. 2007). Reference (Paolucci, Ankolekar et al. 2003) also shows that use of OWL-S does not produce a performance penalty.

Following the research line of OWL-S, a more comprehensive framework named Web Service Modeling Framework (WSMF) was proposed (Fensel and Bussler 2002). Then WSMF was refined and extended by Web Service Modeling Ontology (WSMO). WSMO has four main elements to describe semantic web services: ontology that provides the terminology used by other elements, goals that define the problems that should be solved by web services, web service description that defines various aspects of a web service, and mediators which bypass interoperability problems. WSML is selected as the ontology language of WSMO. WSMO has been implemented by two systems: IRS-III (Domingue, Cabral et al. 2004; Hakimpour, Sell et al. 2005) and Web Service Execution Environment WSMX (Cimpian, Vitvar et al. 2005).

In fact, semantic annotation methods of web services are used more to improve automatic discovery and composition of services; however, besides service discovery and composition (Sycara, Paolucci et al. 2003; da Silva Santos, da Silva et al. 2009), independent semantic description of web services also pays attention to automatic service invocation. For example, OWL-S defines service grounding and WSMO defines “choreography” for web services and “goal” for clients. Furthermore, (Burstein 2004) discusses dynamic invocation of semantic web services described by OWL-S, and service requesters and providers use different ontologies. In this dynamic invocation method, a service requester must do numerous things in order to send out a request message:

a) reading semantic description of a target service and loading related ontologies from the service provider;

b) transforming service description to its own ontology according to mappings between ontologies of the service requester and provider;
c) transforming its own request into ontologies of the target service and transforming the request ontology to a WSDL message.

But it is unreasonable for a service requester to do so many things. This thesis believes that a service requester just needs to send out its request expressed by ontologies of service provider, and when the service provider receives the request, it can understand and deal with the request.

Besides, another invocation mechanism of semantic web services is realized by the WSMO-based project, IRS-III, and the invocation mechanism is goal-centric (Domingue, Cabral et al. 2004; Domingue, Cabral et al. 2008). In the invocation mechanism, a service request is expressed as a goal and sent out to IRS-III platform and the platform, as a broker, will discover, select and invoke the most appropriate service (Cabral, Domingue et al. 2006). This research work is excellent, but the client side and server side in IRS-III use the same ontology; furthermore its "goal" definition is limited to IOPE (input, output, precondition and effect) as defined in WSMO, and it is not precise enough for describing a web service semantically. This will be further discussed in Section I.2.

## I.2. Goal

In different domains, goal has different definitions. In some sports, goal can be game equipment consisting of the place toward which players of a game try to advance a ball or puck in order to score points[41]. In enterprises, goal can have several types: mission, vision, strategic goal, tactical goal and operative goal (Grangel, Chalmeta et al. 2008). In computer science domain, a goal also has different definitions in different branches, e.g. artificial intelligence (Russell and Norvig 2003) and requirement engineering (Lamsweerde and Letier 2004; Lapouchnian 2005). Especially, in SOA/semantic web service domain, a goal is defined as a representation of an objective for which fulfillment is sought through execution of a web service (Roman, Lausen et al. 2006).

When modeling goals, some researchers model a concrete goal (e.g. "book a flight") as a concept (da Silva Santos, Pires et al. 2008; Grangel, Chalmeta et al. 2008; da Silva Santos, da Silva et al. 2009); some model it as an instance (Roman, Lausen et al. 2006); some model it as a prescriptive assertions (Lamsweerde and Letier 2004). Researchers also model goals in

[41] http://wordnetweb.princeton.edu/perl/webwn

different methods, such as, in UML/UML Profile (Supakkul and Chung 2005; da Silva Santos, Pires et al. 2008; Grangel, Chalmeta et al. 2008; da Silva Santos, da Silva et al. 2009) or in ontology (Roman, Lausen et al. 2006; Jokhio 2009). Some goals are just identified by their name/id; and some goals have properties, for example, in (Roman, Lausen et al. 2006) goals have the properties: capability, interface, etc. In different domains, there are different related concepts surrounding the concept "goal", e.g. in (da Silva Santos, Pires et al. 2008; da Silva Santos, da Silva et al. 2009), the concepts related with "goal" are task, service and agent: a "goal" is supported by a "task", a "task" is performed by a "service" and a "service" is provided by an "agent"; however, in (Supakkul and Chung 2005; Abid 2008), concepts related with "goal" are about goals' composition. Table 5-1 provides a short overview of "Goal" in some domains of computer science.

**Table 5-1. Overview of Goal-Based Research**

| **Related Work** | **Viewpoints of goal-based research** | | | | | |
|---|---|---|---|---|---|---|
| | ***Domain*** | ***Representation Style*** | ***Form of a concrete goal*** | ***Modeling Method*** | ***Goal's properties*** | ***Related Concepts*** |
| (Lamsweerde and Letier 2004) | Requirement Engineering | Hybrid | assertion | Informal (natural language); formal (proposition logic) | N/A | Agent, capability |
| (Supakkul and Chung 2005) | Requirement Engineering | Partially explicit | class | UML (Profile) | name, criticality, satisfaction level, offspring goals, parent | Softgoal, contribution, claim, decomposition, etc |
| (Abid 2008) | Requirement Engineering | Hybrid | class | URN/GRL/ UML (Profile) | Goal type, decomposition type, important type, etc. | Decomposition, dependency, contribution, actor, etc. |
| (Grangel, Chalmeta et al. 2008) | Enterprise modeling | Hybrid | class | UML (Profile) | goal type, goal level, children, parent | Strategy, plan, variable |
| (da Silva Santos, Pires et al. 2008; da Silva Santos, da Silva et al. 2009) | WS Discovery & Composition | Partially explicit | class | UML (Profile) | N/A | Task, service, agent, etc. |
| (Roman, Lausen et al. 2006; Jokhio 2009) | WS Modeling & Testing | Implicit | instance | WSMO/WSML | Capability, interface, etc. | Service, Mediator, Ontology |

According to related work listed in Table 5-1, a goal can be represented from three aspects: 1) from its context, such as, IOPE or constraints of a goal, like descriptions in (Lamsweerde and Letier ; Roman, Lausen et al. 2006; Jokhio); 2) from its natural properties, such as, its name, criticality and relationship with other goals, etc, like that described in (Supakkul and Chung 2005; Abid 2008; Grangel, Chalmeta et al. 2008); 3) from its capability requirement, i.e., it wants to achieve what. Capability requirement of a goal can be expressed by its identifier and the relationship with other identifiers, like description in (da Silva Santos, Pires et al. 2008; da Silva Santos, da Silva et al. 2009). As a goal's context does not describe

its semantics directly, so this representation style of a goal is ***implicit***; although a goal's properties can directly describe its semantics, but such information cannot semantically reflect its real purpose, so such representation style is ***partially explicit***; a goal's real purpose can be represented by its identifier and relationship with other identifiers, but such identifier is just a string, meaningful to people, but less meaningful to computers, so this representation style is also ***partially explicit***. Of course, a goal can also be represented by combination of the above representation styles, and this is the ***hybrid*** representation style. But no above representation styles can directly define a goal's real purpose: its capability requirement. So in Section II, this chapter will propose an ***explicit*** representation style of a goal to solve this problem.

# II. ONTOLOGY-BASED AND GOAL-DRIVEN SERVICE INVOCATION

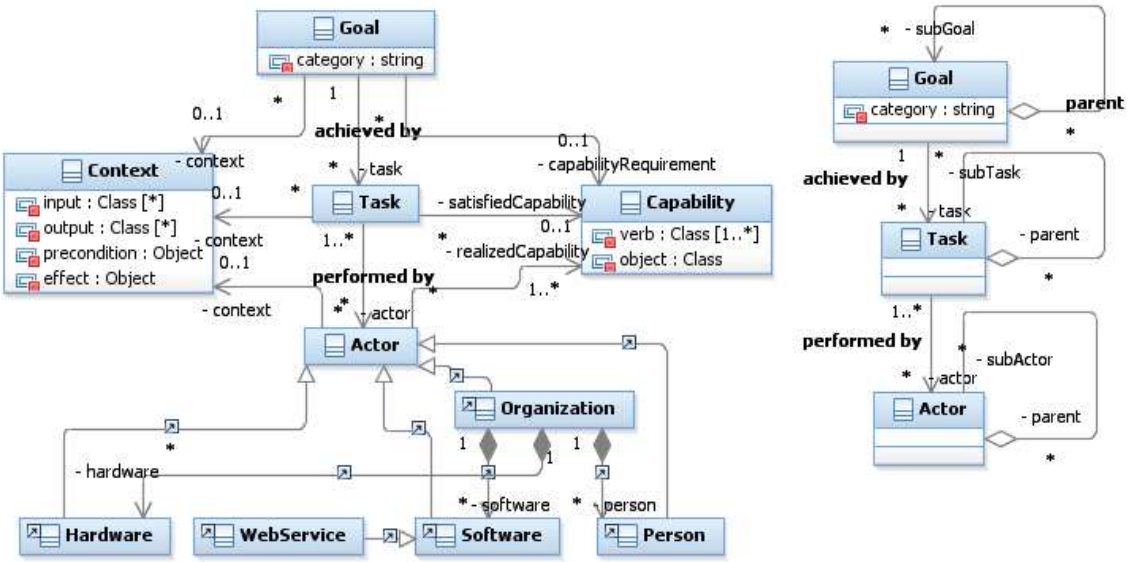
## II.1. Goal Model

After studies about goals in Section I.2, this chapter proposes goal ontology described in Figure 5-1. In order to graphically represent the model, Figure 5-1 adopts UML rather than ontology languages. The representation of the goal ontology in OWL can be gotten in Appendix E. In Figure 5-1, a *Goal* can be **achieved by** several *Tasks* and a Task can be **performed by** several *Actors*. The actors can be an *Organization*, a *Person*, *Software* or *Hardware*. The **multiplicity relationship** between the three concepts has been identified in Figure 5-1. There is an example to explain the multiplicity relationship between Goal and Task. The Goal "translate French to Chinese" can be achieved by the task "translate French to Chinese" and it can also be achieved by two tasks "translate French to English" and "translate English to Chinese".

In Figure 5-1.a, a goal's real purpose is described by its property *capabilityRequirement*, which is an instance of the concept "Capability". *Capability* provides a description of a certain capability or functionality ("do what") by a list of *verb* and an *object*, so capability must depend on a semantic dictionary to explain meaning of verbs and objects used in Capability. This is the ***explicit*** representation style of a "Goal". In this model, a goal can also be described by its context (***implicit*** style), such as, IOPE. But its precondition (P) and effect (E) are not emphasized like that in the state-based goal model described in (Stollberg and Hepp 2006; Stollberg and Norton 2007). Because precondition and effect describe the state of the world before and after a goal's achievement but the state of the world at a certain time may not be known (Stollberg and Norton 2007). Furthermore, the state of the

world will be a significant burden on system designers and it will cause some issues during collaboration of different information systems (Ankolekar, Huch et al. 2002). So in the goal model of Figure 5-1, precondition and effect are used for mappings to other goal ontologies. Besides, Goal has the property *category* which is based on a classification mechanism like the North American Industry Classification System (NAICS) (US-Census-Bureau 2007). A goal's category will limit research scope during the discovery of targeted tasks or actors. In a world, Goal in Figure 5-1 has a hybrid representation style (combining explicit style and implicit style).

A goal is created and sent out from client side; when receiving it, a server side will find an appropriate task to satisfy the required capability and then deliver the task to an actor to realize the corresponding capability. Some tasks can be performed by organizations or persons and the others can be performed by software (e.g., web services) or hardware (e.g., printers or sensors).




(a) General Model (b) Composition Model

**Figure 5-1. Goal Model**

Figure 5-1.b is a composition model for goal model. It includes Goal composition, Task composition and Actor composition. A goal can have several sub-goals, a task can have several subtasks and an actor may have several sub-actors. For example, a goal "Organize a trip" can have four sub-goals "find flight", "find hotel", "book flight" and "book hotel" (Stollberg and Norton 2007); an actor, e.g., "organization" or "web service" can be composed of sub-organizations (departments or branches) or sub web services. In fact, through analysis of goal's composition (Supakkul and Chung 2005; Stollberg and Norton 2007), task

composition (business process) (OMG 2011) and web service composition (Sycara, Paolucci et al. 2003; da Silva Santos, da Silva et al. 2009), we can find that these three kinds of compositions all revolve on capability composition, just from different viewpoints. Goal composition is constructed from clients' viewpoint to describe clients' requirements; task composition (business process) is constructed from organizations' viewpoint to satisfy clients' requirements and achieve business agility; web service composition is constructed from software developers' viewpoint to realize tasks' capabilities and increase web services' reusability and at the same time web service composition improves software development efficiency and productivity. Goal is helpful to semi-automatic composition of web services as described in (Hakimpour, Sell et al. 2005). But web service composition, like WS-BPEL2.0, lacks human interactions, so BPEL4People[42] and WS-HumanTask[43] are proposed. The two new specifications make web service composition nearer to task composition like BPMN2.0. In fact, a composite task (business process) in BPMN2.0 can use atomic/composite web services as implementation of some of its sub-tasks.

### II.2. Ontology-based and Goal-driven SOAP

After construction of goal ontology in Section II.1, this section will discuss how to use it to invoke target web services to realize tasks. As SOAP (Simple Object Access Protocol) has been widely used as transport protocol of web services, the transport protocol in this thesis is based on SOAP. Traditionally, when invoking a web service, a client will send out a SOAP message to the web service server: the structure of the message is defined in SOAP and information contained in the message is defined in a corresponding WSDL file. This means that before invoking a wanted web service, a client has to manually find its location and download its WSDL file and understand meaning of required information for its invocation and then write invocation code. This will result in tight coupling between clients and web services. Because if a web service operation changes its name or its input parameters for some reasons, clients' invocation code has to be changed correspondingly. How to solve the problem about tight coupling is critical to enterprise interoperability.

After making full advantage of the extensibility mechanism of SOAP (W3C 2007), this chapter proposes OBGD SOAP message illustrated in Figure 5-2. It contains three main scopes. SCOPE 1 is in the "Header" element and it is defined in Figure 5-3. It contains the elements that point to referenced ontologies. These ontologies are the foundation for the

[42] http://www.oasis-open.org/committees/bpel4people/

[43] http://docs.oasis-open.org/bpel4people/ws-humantask-1.1.html

definition of goals and they are also necessary for SCOPE 2 and 3. SCOPE 2 contains a concrete goal. The concrete goal is an instance of the concept *Goal* in Figure 5-1. SCOPE 3 contains all the information required to achieve the goal in SCOPE 2. The referenced (dependent) ontology of OBGD SOAP messages can be provided by service requester, service provider or third party. This will be discussed at the end of this section.

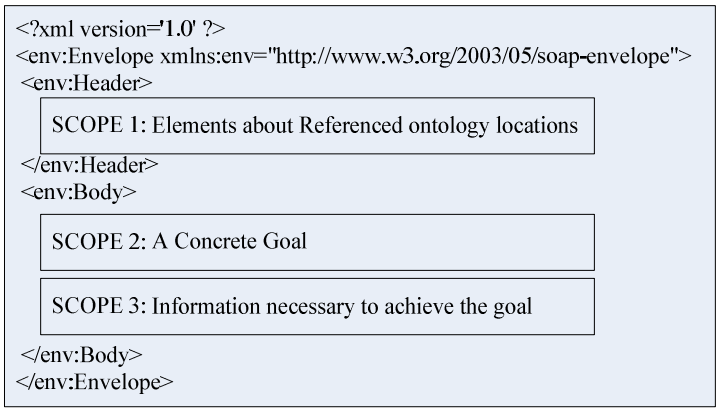



**Figure 5-2. Ontology-Based and Goal-Driven SOAP Message**

When a web service provider receives an OBGD SOAP message, it must tackle the message by the following steps:

**STEP 1:**

**For an inbound message:** *If the referenced ontologies in SCOPE 1 come from service provider, then go to STEP 2; if the referenced ontologies in SCOPE 1 are defined by others but service provider has mappings from these ontologies to its own ontologies, then service provider will transform SCOPE 2 and SCOPE 3 of the received message to its own ontologies; otherwise, service provider will return a fault message to indicate that it cannot understand the request.*

**For an outbound message:** *SCOPE 2 and SCOPE 3 of the message from STEP 2 will be transformed to original ontology of the related inbound message; the corresponding SOAP header block defined in Figure 5-3 will be updated in the outbound message if necessary.*

**STEP 2:**

**For an inbound message:** *Find an appropriate target task/service (operation) according to the goal included in the inbound message, and go to STEP 3; if there is no satisfied task/service, then return a fault message to indicate that it cannot satisfy the requested capability.*

**For an outbound message:** *Just deliver the message to STEP 1.*

**STEP 3:**

**For an inbound message:** *Transform the message of STEP 2 from ontology description to the XML description which is defined in the WSDL file of the target service. This step depends on mappings between domain ontology and web services' definitions.*

**For an outbound message:** *The message will be transformed from XML data types to domain ontology and the referenced ontology locations will be added into the outbound message, and then the message will be delivered to STEP 2.*

**STEP 4:**

**For an inbound message:** *The message from STEP 3 is tackled as a normal SOAP message by some SOAP implementations, such as Apache Axis[44] or CXF[45]. The message will be transformed from XML message to a technique-specific message, for example, a java object. The new generated message is delivered to the target service component.*

**For an outbound message:** *After execution of a target service component, a response (an outbound message) will be returned. The response is transformed from a technique-specific message to a normal SOAP message and then delivered to STEP 3.*

```
<xs:schema xmlns:xs="http://www.w3.org/2001/
XMLSchema" xmlns="http://lm2oserver.ec-lille.fr/
LM2OserverPloneSite/Members/LiuH/OBGDSOAP"
targetNamespace="http://lm2oserver.ec-lille.fr/
LM2OserverPloneSite/Members/LiuH/OBGDSOAP"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
<xs:element name="ontologyRef" type="ontologyRefType"/>
<xs:complexType name="ontologyRefType">
<xs:attribute name = "location" type="xs:anyURI"/>
</xs:complexType>
</xs:schema>
```

**Figure 5-3. Schema of SOAP module for locations of referenced ontology**

The above steps are positioned in an ontology-based and goal-driven service invocation mechanism described in Figure 5-4. They form a *Processor Chain including Processor 1, Processor 2, Processor 3 and SOAP Processor*. Processor 1 corresponds to STEP 1; Processor 2 corresponds to STEP 2; Processor 3 corresponds to STEP 3; SOAP Processor corresponds to STEP 4. In addition, in Figure 5-4, ontology base will contain domain ontology, ontology-based service descriptions, mappings to external ontologies and mappings to data types defined in WSDL files.

[44] http://axis.apache.org/axis/

[45] http://cxf.apache.org/

In Figure 5-4, an OBGD message sent out from a service requester respects the format defined in Figure 5-2. After it (as an inbound message) is received by a service provider, it will be treated by Processors 1, 2 and 3 and SOAP Processor. At the end, it will be delivered to a target service component. After the execution of the service component, a response (an outbound message) will be returned and then it will be treated by SOAP Processor and Processor 3, 2 and 1. Finally, the response will be sent back to the service requester. Furthermore, Processors 1, 2 and 3 have their *admission condition*: an inbound message, as a request message, must have SCOPE 1; otherwise, the inbound message will bypass the three processors and it will be directly treated by SOAP Processor as a normal SOAP message. The admission condition makes the service invocation mechanism in Figure 5-4 also support normal SOAP-based service invocation.

This invocation mechanism has two assumptions:

**Assumption 1:** *All service providers must understand the header block for locations of referenced ontology defined in Figure 5-3.*

**Assumption 2:** *Although service requesters do not need to know concrete service addresses, they must at least know service providers' addresses. The difference between service address and service provider's address will be discussed further in Section III.1.*

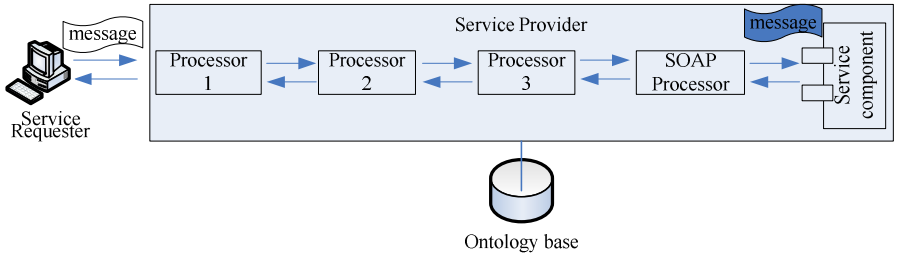



**Figure 5-4. Ontology-based and Goal-driven service invocation**

From the above elaboration, it is evident that web service requesters just need to send OBGD SOAP messages and do not need to know detailed definitions of web services. Therefore, it makes service requesters and web service definitions loosely coupled. This realizes loose coupling at service contract level (defined in Table 2-2).

In addition, when an OBGD SOAP message is sent out from a service requester, the message can depend on ontologies from the service requester, a third party or a target service provider. If the service requester makes sure that the target service provider can understand its own ontology, then OBGD SOAP message can be constructed based on its own ontology (Figure 5-5.a). If the service requester and the service provider have negotiated and decided to use a common ontology, which may be created by themselves or come from a third party,

then during their communication, OBGD SOAP message can depend on such ontology (Figure 5-5.c). If the service requester is not sure that the service provider can understand its ontology, and meanwhile the service requester understands the service provider's ontology, then OBGD SOAP message can be constructed based on the service provider's ontology (Figure 5-5.b). No matter which ontology the OBGD SOAP messages depends on, the service provider can tackle it by Processor 1. So loose coupling at semantic layer (defined in Table 2-2) can be achieved by OBGD service invocation.

However, there is another problem: how is OBGD SOAP message generated by a service requester, especially if the service requester is an enterprise? This will be discussed in Section III.1.

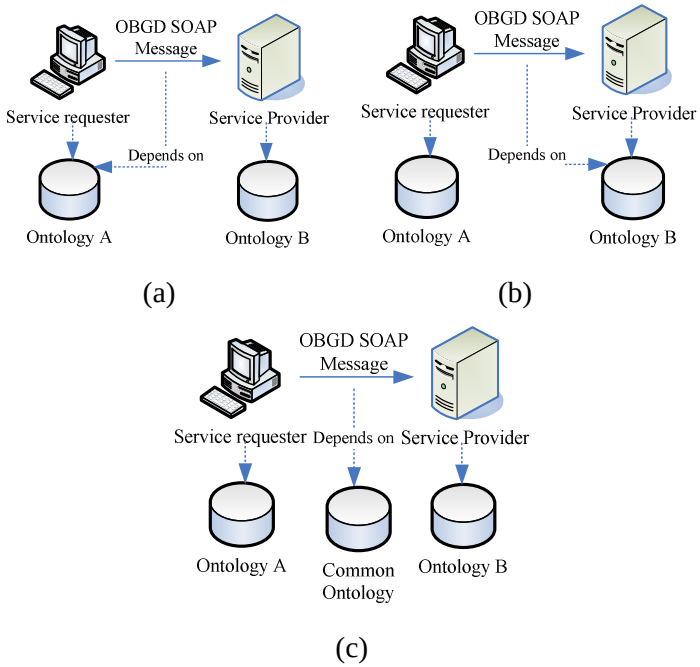



**Figure 5-5. Dependent ontology of OBGD SOAP message**

## III. ONTOLOGY-BASED AND GOAL-DRIVEN ARCHITECTURE FOR ENTERPRISE INTEROPERABILITY

Section II has proposed an ontology-based and goal-driven method of message transport between web service requesters and providers. Such method can make requesters and providers loosely coupled and make them understand each other, which is the foundation to realize enterprise interoperability. If service requesters and providers are different enterprises, then how to make them seamlessly interoperate without considering definitions or locations of web services? Section II just discusses the problem from the viewpoint of service providers;

instead, this section will provide a full landscape by supplementation of the viewpoint of client providers.

### III.1. Generation Mechanism of OBGD SOAP Messages

In order to make a service requester generate OBGD SOAP messages automatically, this section proposes a symmetric mechanism for OBGD service invocation in Figure 5-6. In Figure 5-6, Enterprise A (as a service requester) wants to send a request to Enterprise B (as a service provider). Enterprise B supports the OBGD service invocation mechanism described in Figure 5-4. Enterprise A uses the similar mechanism to generate OBGD SOAP message automatically and send the message to Enterprise B. The mechanism used by Enterprise A seems the same as that used by Enterprise B, but, in detailed aspects, they are different.

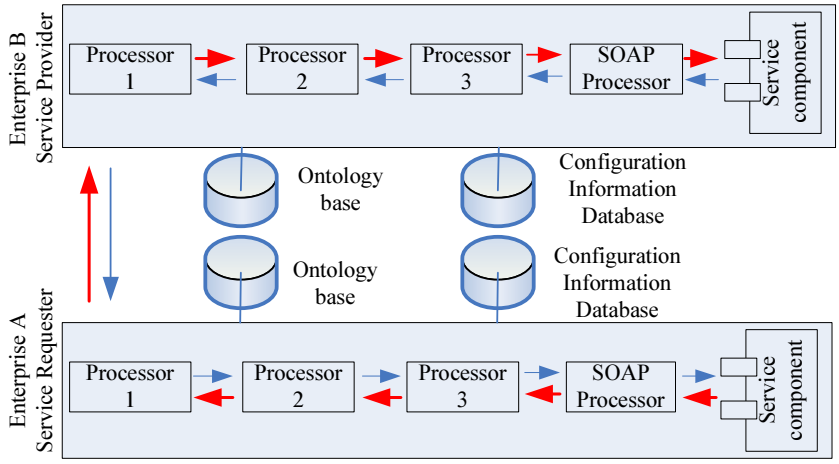



**Figure 5-6. Symmetric mechanism for OBGD service invocation**

In Figure 5-6, a service component executed in Enterprise A depends on another service provided by a service component in Enterprise B. During its execution, the service component in Enterprise A will generate a technique-specific request message, such as, a java object, including a concrete goal and necessary information to achieve the goal. Then the request (an outbound message) will be tackled by the following steps:

**STEP OI-4:**

**For an outbound message:** *The message will be transformed from a technique-specific message to a SOAP message and then it will be delivered to STEP OI-3. As the goal model is not defined by WSDL files, and related information may also not be defined by WSDL files, so there must be a mapping of data types between programming language and XML. Such mapping is stored in a* ***C****onfiguration* ***I****nformation* ***D****ata****b****ase (CI-DB).*

**For an inbound message:** *The message is transformed from a SOAP message to a technique-specific message and then delivered to the related service component.*

**STEP OI-3:**

**For an outbound message:** *The message is transformed from XML data types to domain ontology; the referenced ontology locations will be added into the message. Then the message will be delivered to STEP OI-2.*

**For an inbound message:** *The message is transformed from domain ontology to XML data types and then delivered to STEP OI-4.*

**STEP OI-2:**

**For an outbound message:** *Find out which service provider offers corresponding capability to satisfy the goal in the message, and then deliver the message to STEP OI-1; if there is no such service provider, then return a fault message. This step needs information about all available service providers, including service providers' addresses and their capabilities; such information is stored in CI-DB.*

**Four an inbound message:** *Just deliver the message to STEP OI-3.*

**STEP OI-1:**

**For an outbound message:** *Check which ontology the OBGD SOAP message depends on. If the message depends on ontology of service provider or a common ontology, then SCOPE 2 and SCOPE 3 of the message will be transformed and meanwhile referenced ontology locations will be updated. Finally, the message will be sent out to the service provider designated in STEP OI-2. This step depends on configuration information about ontology usage and such information is stored in CI-DB.*

**For an inbound message:** *If the message is based on external ontology, then SCOPE 3 of the message will be transformed onto internal ontology and meanwhile referenced ontology locations will be updated. Then the message is delivered to STEP OI-2.*

In Figure 5-6, for the service requester, SOAP Processor corresponds to STEP OI-4. Processor 3 corresponds to STEP OI-3. Processor 2 corresponds to STEP OI-2. Processor 1 corresponds to STEP OI-1. Ontology base includes definition of external service providers. CI-DB stores configuration information about ontology usage, mapping of data types between programming languages and XML, etc. Obviously, functionality of the processors for the service requester is not the same as but complementary to that for the service provider (Enterprise B).

In Figure 5-6, a request message (indicated by **red thick arrows**) is an outbound message for the service requester but it is an inbound message for the service provider. A response message (indicated by blue thin arrows) is an outbound message for the service

provider but it is an inbound message for the service requester. So, the same request/response message plays opposite roles for the service requester and provider.

Furthermore, when the service component in Enterprise A ***produces*** (generates) a request, a processor chain in Enterprise A is created. Evidently, the creation of the processor chain is driven by **internal request**. The processor chain transforms the request to an OBGD SOAP message and sends the message to the service provider. When the service provider ***consumes*** **(receives)** an OBGD SOAP message, a processor chain is also created. But the creation of the processor chain is driven by **external request**. Evidently, the execution order of processor chains for the service requester and provider is opposite.

Generally speaking, *generation mechanism* and *consumption mechanism (OBGD service invocation mechanism)* for OBGD SOAP messages are different and complementary. Their combination is defined in Figure 5-6 as *symmetric mechanism for OBGD service invocation*.

In addition, in Figure 5-6, Enterprises A and B may exchange their roles (service requester, service provider) in some situations, i.e. Enterprise A can also provide its services to others, especially, *Enterprise A can provide its services to itself*. Thereby, for an enterprise, it must implement symmetric mechanism for OBGD service invocation, not just one mechanism (generation or consumption mechanism).

Symmetric mechanism for OBGD service invocation has an assumption:

**Assumption 3:** *API for goal model and related business concepts must be implemented by the techniques corresponding to service components of Enterprises. The API helps software developers to create concrete goals and necessary information in programming languages.*

According to symmetric mechanism of OBGD service invocation, if a service component wants to invoke a service, software developers just need to write some code to send a concrete goal and related information to a processor chain and they do not need to know what or where the target service is, which ontology the target service depends on or how the target service is implemented. Hence, symmetric mechanism of OBGD service invocation has the following properties: *location transparency, semantics transparency* and *technique transparency*.

Location transparency is support by service discovery in symmetric mechanism for OBGD service invocation. The service discovery is depicted in Figure 5-7. Enterprise A is a service requester and Enterprise B is a service provider. A request message is sent out from a

service component[46] in Enterprise A (generation mechanism). At the beginning, the request message does not declare the address of the target service provider. After Processor 2 (STEP OI-2: For an outbound message), the target service provider (target Enterprise) is determined according to the goal of the request message. When Enterprise B (consumption mechanism) receives the request message, Processor 2 (STEP 2: For an inbound message) will determine the target service component according to the goal of the request message. Through the above analysis, a service requester (service component in Enterprise A) does not need to know the address of the target service provider (service component in Enterprise B). It just needs to send out its request including its goal and necessary information. This is defined as *location transparency*.

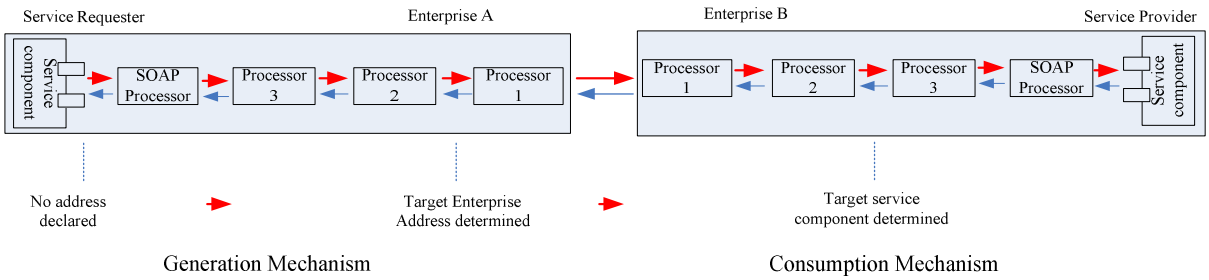



**Figure 5-7. Service discovery in symmetric mechanism for OBGD service invocation**

Semantic transparency is supported by message transformations in symmetric mechanism of OBGD service invocation. The message transformations are depicted in Figure 5-8. Figure 5-8 just analyzes a request message sent out from Enterprise A and received by Enterprise B. In Enterprise A (generation mechanism), a request message sent out from a service component, and the message is technique-specific. It will be transformed to an XML-based message by SOAP Processor (STEP OI-4: For an outbound message), and then transformed to OBGD message by Processor 3 (STEP OI-3: For an outbound message). In this moment, the message is based on the ontology constructed by Enterprise A. Finally, the message will be sent out from Enterprise A. In Enterprise B (consumption mechanism), the received message will be transformed by Processor 1 (STEP 1: For an inbound message) to another OBGD message. The message now is based on the ontology constructed by Enterprise B. Then, the message will be transformed to an XML-based message by Processor 3 (STEP 4: For an inbound message). Through analysis of the above message transformations, service requester (service component in Enterprise A) does not need to know its own ontology

[46] In fact, a service component in Enterprise A is the real service requester. Enterprise A is just the representative of the service component when it invokes external services.

and service provider's ontology; it just needs to send out its request including its goal and necessary information. This is defined as *semantic transparency*.

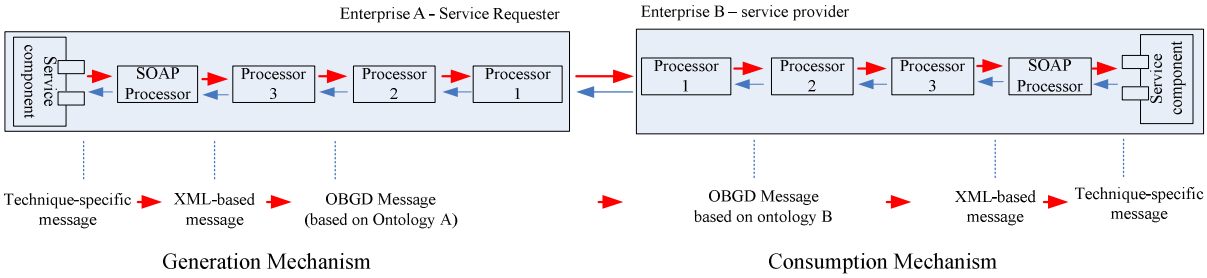



**Figure 5-8. Message transformations in symmetric mechanism for OBGD service invocation**

Figure 5-8 also implies technique transparency. Implementation techniques of a service provider and a service requester are separated by ontology and SOAP protocols. When sending a request message, a service requester does not need to know implementation technique of a service provider. This is defined as *technique transparency*. For example, a service requester can use JAVA technique and a service provider can use C# technique.

According to the above study, the three properties are realized by service requesters and providers with the help of their processor chains, ontologies, ontology mappings and related configuration information. These properties are significant to enterprise interoperability.

## III.2. OBGD Architecture for Enterprise Interoperability

After the above research, we propose the **o**ntology-**b**ased and **g**oal-**d**riven (OBGD) architecture for enterprise interoperability in Figure 5-9. The core of the architecture is **semantic service bus** (**SSB**). SSB implements the symmetric mechanism of OBGD service invocation. SSB is in charge of construction, maintenance and destruction of processor chains.

In Figure 5-9, service components are managed by component containers from the aspects: life-cycle management, instance pooling and instance persistence, etc. **Service components** can be software components, business rules or **business processes**, etc. **Component containers** can be software containers, business rule engines or **business process engines**, etc. Ontology base used in an enterprise contains all domain ontologies, ontology-based service descriptions, ontology-based descriptions of other enterprises, mappings to ontologies of other enterprises, mappings to its own XML-based business concepts, etc. Configuration information database (CI-DB) contains mappings of data types between in programming languages and in XML, and it also contains locations of service components and configuration information about ontology usage, etc. Normally, an enterprise

authenticates service requesters and controls their access, so CI-DB also contains service requesters' authentication information, access control information, etc. Such security functionality can be implemented by processors and then added into processor chains. SSB, component containers, ontology base and CI-DB can be deployed together or dispersedly. For example, containers can be deployed with SSB or not, so SSB must be able to invoke local or remote containers. This will be further discussed in Section III.3.

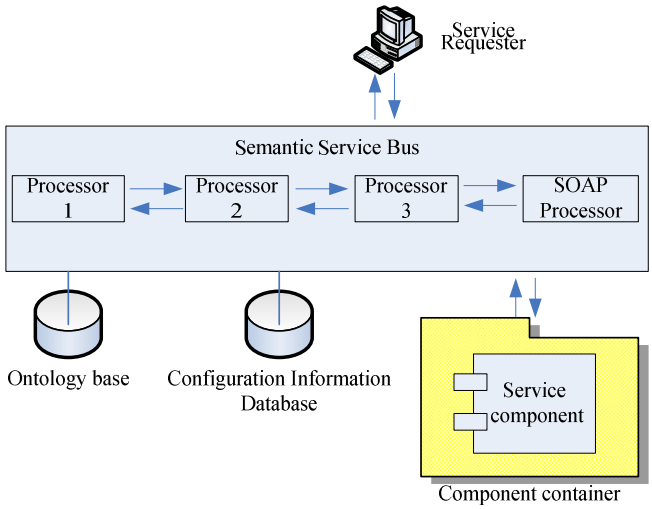



**Figure 5-9. OBGD architecture for enterprise interoperability**

As the above architecture realizes the symmetric mechanism of OBGD service invocation, therefore, it can support the properties: location transparency, semantic transparency and technique transparency. That is to say, it can make enterprises collaborate with others transparently in the above three aspects. In the above architecture, containers make service components more manageable and make their invocation more efficient.

Besides, the above architecture is flexible. New functionalities can be added as processors into processor chains. New containers can also be added as long as SSB can invoke such new containers locally or remotely (from this viewpoint, SSB can also be regarded as a container manager). New service components can also be deployed in corresponding service containers.

### III.3. Deployment of OBGD-SSB for Intra-Enterprise Interoperability

OBGD-SSB can be used to support intra-enterprise interoperability. For a small enterprise, the OBGD-SSB can be deployed in a **centralized** method. That is to say, SSB, containers, service components, ontology base and configuration information database are

deployed just on a server. But if an enterprise is big and its departments/affiliates have their own service containers and service components, then the OBGD architecture has at least three other deployment methods described in Figure 5-10. The three deployment methods are all **decentralized**.

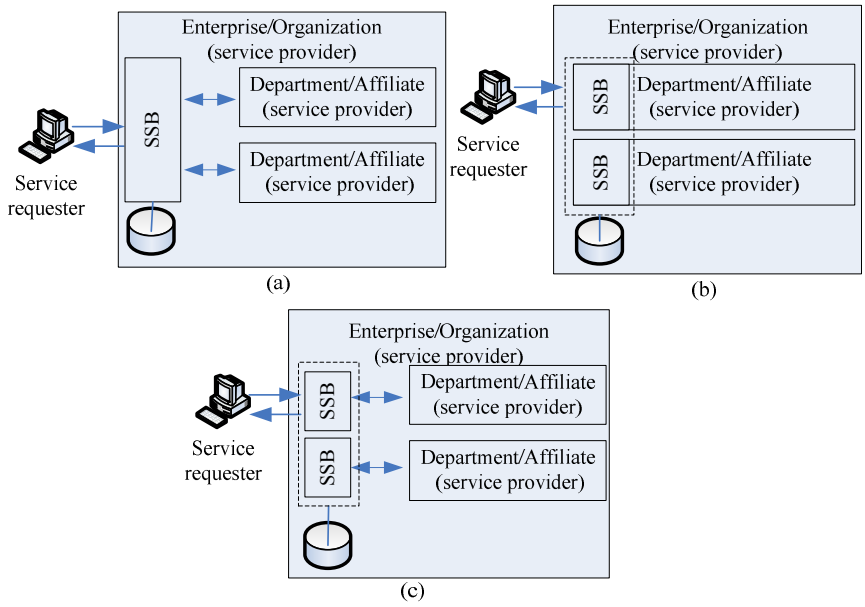



**Figure 5-10. Deployment of OBGD architecture for enterprise interoperability**

The first method is an **integral style** for decentralized deployment, described in Figure 5-10.a. In Figure 5-10.a, service containers deployed in all departments or affiliates are registered into SSB. A service requester just needs to send a request message to SSB, and then SSB will find an appropriate service and invoke it **remotely**. In this method, SSB looks like glue to integrate all service containers (including service components) from different departments/affiliates. Of course, one department/affiliate can use services offered by other departments/affiliates through SSB, as described in Section III.1.

The second method is a **decentralized style** for decentralized deployment, described in Figure 5-10.b. SSB is deployed for each department/affiliate, but all SSB share the same information base (including ontology base and configuration information database), and this will generate a virtual SSB (indicated by a dashed rectangle in Figure 5-10.b) among different departments/affiliates. When a service requester wants to send a request message to the enterprise, it can send the message to any department/affiliate of the enterprise. If one department/affiliate receives a message and finds the requested service is on another department/affiliate, then the message will be delivered from the current SSB to the target SSB, but the whole transfer process is transparent to the service requester. In fact, this is also

a level of location transparency. In this method, all service components in different departments/affiliates are invoked **locally** by their own SSB.

Comparing the above two styles for decentralized deployment, the first style physically integrates information systems of all departments/affiliates by a single SSB. But, if SSB fails, all departments/affiliates are disconnected and all service requesters cannot use services published by SSB. The second style logically integrates information systems of all departments/affiliates by their own SSB, so each SSB can be implemented in different techniques. Most importantly, if one SSB fails, other available SSB can also interoperate, and service requesters can also use services offered by available SSB. Evidently, the first style does not influence original information system for each department/affiliate, but the second style will influence them. In order to make full use of the two styles' advantage, the third method for decentralized deployment is created in Figure 5-10.c and it is a **hybrid style**: combination of the integral and decentralized styles. Each department/affiliate has its own SSB, and all SSB share the same information base and each SSB will remotely invoke its own service components; if one SSB finds a required service is provided by another SSB, then it will deliver the request message to the target SSB. The hybrid style has no single-point failure like the integral style and does not influence original information system.

### III.4. Federated Deployment of OBGD-SSB for Inter-Enterprise Interoperability

Section III.3 has analyzed deployment methods for OBGD-SSB in one enterprise to realize intra-enterprise interoperability. In fact, OBGD-SSB can also be deployed in a federated style to support inter-enterprise interoperability. The federated deployment is depicted in Figure 5-11. In Figure 5-11, OBGD-SSB is deployed by each enterprise in the Internet. Each SSB is controlled and managed by the corresponding enterprise. Each SSB has its own database (ontology base and CI-DB) for interoperability. Any SSB can be attached to or leave from the Internet. The attachment or leaving of any SSB will not be managed or controlled by a central manager. Therefore, each SSB is autonomous (see Figure 1-3.c and Table 1-2 in Chapter 1-Section III). SSB is the facade of business functionalities for each enterprise. That is to say, all internal business functionalities (service providers) have been encapsulated by SSB. In the Internet, any service requester just sees a single (*physical* or *logical*) SSB for an enterprise, and it cannot see internal implementations of services. Any request to services will be delivered to SSB, and SSB will invoke an appropriate service to respond to the request.

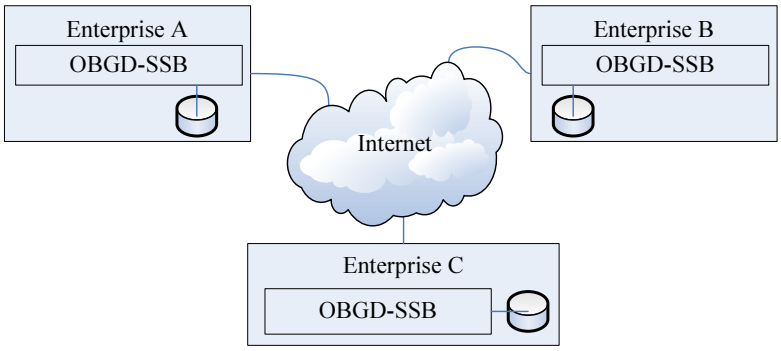



**Figure 5-11. Federated deployment of OBGD-SSB for Inter-Enterprise Interoperability**

## IV. OBGD ARCHITECTURE AND PBMEI

OBGD architecture for enterprise interoperability is designed to realize the platform/ infrastructure in the framework for IT solutions to enterprise interoperability problems (see Figure 2-15 in Chapter 2-Section VI). Ontology-based PBMEI is designed to realize the modeling space of the framework (see Figure 2-15). OBGD architecture can support PBMEI from two aspects: horizontal transformation of collaborative business processes and execution of interoperability processes generated in PBMEI.

As studied in Section I.2.b, at Level 2 of PBMEI, the core cooperator will deliver collaborative processes and its sub-processes to relevant collaborators. The deliverance can be supported by OBGD architecture. As illustrated in Figure 5-8, the core cooperator can transform XML-based collaborative processes into OBGD message and send the message to relevant collaborators. The message sent out is based on ontology of the core cooperator. When the message is received by a collaborator, the message will be transformed by Processor 1 (STEP 1: For an inbound message). After the transformation, the message is described based on ontology of the collaborator. Then, collaborative processes in the message can be transformed to the XML-based processes that can be understood by the collaborator. The above procedure is the horizontal transformation of collaborative processes. This procedure is also required at CIM level in Figure 2-15.

Besides supporting horizontal transformation, OBGD architecture can also support execution of interoperability processes generated in PBMEI. In OBGD architecture, component container can be process engine and service component can be business processes. So interoperability processes (in BPMN) can be deployed into (BPMN) process engines (like BPMN engines discussed in Chapter 2-Section I.3). However, process engine for (BPMN) interoperability processes in OBGD architecture is different from normal (BPMN) process engines. Because in interoperability processes, there are numerous semantic annotations for

process elements, and process engine in OBGD architecture will use these annotations to help generation of OBGD SOAP messages. It means that OBGD SOAP messages sent out from process engine will bypass "SOAP Processor" in OBGD-SSB. Further study about process engine in OBGD architecture will be done in the future.

## V. CONCLUSIONS

After analyzing problems existing in semantic web services and goal-based researches, we have proposed a goal model which can directly express semantics of goals. Based on the goal model, we have designed an ontology-based and goal-driven SOAP which defines a message format, a corresponding SOAP module (header block) and a related SOAP processing model (OBGD service invocation mechanism). This makes a service requester be able to invoke its desired services according to its goal and related necessary information. In fact, OBGD SOAP makes service requesters and service definitions loosely coupled at the contract and semantic levels (see Table 2-2).

In order to facilitate generation of OBGD SOAP message for service requesters, we have proposed a symmetric mechanism for OBGD service invocation. This symmetric mechanism makes service requesters invoke a desired service without knowledge of its location, semantics or implementation technique.

Based on the symmetric mechanism of OBGD service invocation, OBGD architecture for enterprise interoperability has been proposed. The architecture inherits properties of the symmetric mechanism: location transparency, semantics transparency and technique transparency. These properties are critical for enterprise interoperability. Besides these properties, OBGD architecture also satisfies some general requirements, such as, making its service components more manageable and making their invocation more efficient.

In order to deeply research usage of OBGD architecture, we have also studied its deployment. For intra-enterprise interoperability, we have proposed three styles of decentralized deployment. For inter-enterprise interoperability, we have proposed a federated deployment method. The federated style is one objective of our research as described in Chapter 1-Section II.

At last, we have studied the relationship between OBGD architecture and ontology-based PBMEI (proposed in Chapter 3 and 4). OBGD architecture can at least support PBMEI in two aspects: horizontal transformation of collaborative processes and execution of interoperability processes.

A prototype for OBGD SOAP should be constructed in the future. An algorithm of OBGD service discovery used in STEP 2 (Section II) and an algorithm of OBGD service provider discovery used in STEP OI-2 (Section III) should also be studied in the future. In order to support PBMEI, the OBGD architecture should be integrated with business process/rule engines. The architecture should also make human beings interact with business processes during their execution.

# CHAPTER 6: Conclusions and Perspectives

When enterprises collaborate with others to achieve business objectives, enterprise interoperability problems will be encountered. A mediator-based approach to enterprise interoperability problems has been studied in (Touzi 2007) and (Truptil 2011). Instead, in this thesis, we have proposed a federated approach to enterprise interoperability problems at methodological and technical levels.

In this thesis, firstly, we have summarized enterprise interoperability in four dimensions: its definition, framework, solutions and maturity models. Secondly, in order to solve enterprise interoperability problems, we have analyzed five related research domains: collaborative business process, MDA, SOA, ESB and ontology. Then, we have proposed a framework for IT solutions to enterprise interoperability problems, see Figure 6-1. The framework integrates the above five research domains together. Thirdly, in order to realize the above framework at methodological and technical levels, we have proposed an ontology-based and **p**rocess-**b**ased **m**ethod for **e**nterprise **i**nteroperability (ontology-based PBMEI) and an ontology-base and goal-driven (OBGD) architecture for enterprise interoperability. Therefore, **the main contribution** of our work is the framework for IT solutions to enterprise interoperability problem and its realizations at methodological and technical levels: ontology-based PBMEI and OBGD architecture.

The framework (see Figure 6-1) starts from business environment and ends at IT environment. At the methodological level (in the upper red rectangle), the framework employs business process, MDA, SOA and ontology to align business and IT environments. At the technical level (in the lower red rectangle), the framework employs ESB and ontology (semantic ESB) as the platform/infrastructure in IT environment. This framework also covers three key research domains about enterprise interoperability proposed in (Chen and Doumeingts 2003): enterprise modeling, architecture & platform and ontology. The three key research domains have been identified in green rectangles in Figure 6-1.

Ontology-based and Process-Based Method for Enterprise Interoperability (Ontology-based PBMEI) is constructed to realize the framework in Figure 6-1 at the methodological level. This method has five levels. At the first level, the method uses a collaborative process to represent collaboration requirements between enterprises. At the second level, collaborative process is annotated with collaborator's information and it is transformed into sub-processes. The above two levels are global for all collaborators while the rest steps are local for each collaborator. The rest steps are affected by different uses of ontology in PBMEI. If, in enterprise collaboration, there is not a core cooperator, then collaborative process is created by negotiation of all collaborators. After the first and second levels, collaborative process and

its sub-processes are delivered to all relevant collaborators and they are transformed to local collaborative processes (at the third level). At the fourth level, message types in processes are determined and mapping from collaborators to participants is finished. At the fifth level, message transport protocols are fixed. At last, executable interoperability processes are generated. The above levels constitute the first variant of PBMEI. If, in enterprise collaboration, there is a core cooperator, then collaborative process is created by the core cooperator. After the first and second levels, the core cooperator directly executes the fourth and fifth levels of the first variant of PBMEI, but the other collaborators execute the third to the fifth levels of the first variant of PBMEI. This constitutes the second variant of PBMEI.

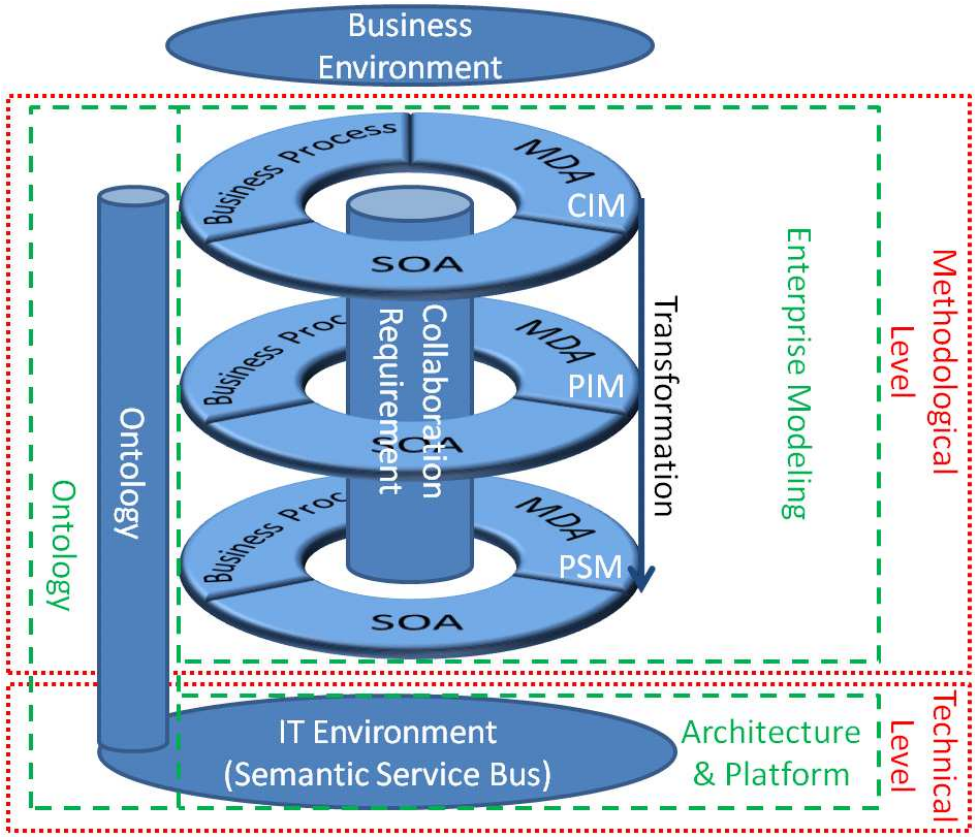



**Figure 6-1. Individual View of the Framework for IT solutions to Enterprise Interoperability Problems**

Ontology-based PBMEI also includes process transformations. The transformation between the first and second levels is based on two quantitative criteria: rank of collaborative process and cooperation rate. The transformation has been studied base on a case "ShoppingDrive". The case study indicates that the transformation can reduce message numbers between collaborators in enterprise collaboration and it can also improve reusability of collaborative processes. In ontology-based PBMEI, transformation between other levels is

based on semantic annotations in collaborative processes. Semantic annotations in collaborative processes have been studied based on the extension mechanism of BPMN2.0.

The ontology-based and goal-driven (OBGD) architecture is designed to realize the framework in Figure 6-1 at the technical level. The core of the OBGD architecture is an OBGD semantic service bus. This service bus is based on a symmetric mechanism for OBGD service invocation. The symmetric mechanism is designed according to OBGD Simple Object Access Protocol (SOAP). The OBGD SOAP is made up of three parts: OBGD message format definition, SOAP module definition and SOAP processing model definition. Such symmetric mechanism has three properties: location transparency, semantics transparency and technique transparency. The properties are critical to enterprise interoperability, especially to execution of interoperability processes. This architecture can deploy OBGD semantic service bus in different styles to support intra- or inter- enterprise interoperability. Especially, it can deploy OBGD semantic service bus in a federated style to support inter-enterprise interoperability.

Ontology-based PBMEI and OBGD architecture have a close relationship. In ontology-based PBMEI, at the second level, collaborative business processes and its sub-processes will be delivered to other collaborators. This depends on horizontal process transformation. The Horizontal transformation is supported by OBGD architecture. Besides, in PBMEI, executable interoperability processes will be generated. Execution of these processes is supported by process engines in OBGD architecture.

Ontology-based PBMEI and OBGD architecture are all grounded in ontology. The influence of ontology on PBMEI and OBGD architecture is concluded in Table 6-1. We have analyzed the influence of ontology from three interoperability concerns: data interoperability, service interoperability and process interoperability. The three concerns have been defined in (Chen and Daclin 2006). According to Table 6-1, conceptual barriers in data, service and process aspects can be removed by ontology-based PBMEI (at the methodological level in Figure 6-1). Technical barriers in data, service and process aspects can be removed by OBGD architecture (at the technical level in Figure 6-1).

Besides, ontology-based PBMEI and OBGD architecture together constitute a federated approach to enterprise interoperability problems. Firstly, as ontology-based PBMEI, except its global levels, is respected and performed separately by all collaborators, and each collaborator is autonomous, so this method is federated. Secondly, OBGD semantic service bus in OBGD architecture can be deployed in a federated style to support collaborations between

enterprises, so this architecture can support federation when solving enterprise interoperability problems.

**Table 6-1. Influence of ontology on ontology-based PBMEI and OBGD architecture**

| **Interoperability concerns** | **Ontology-based PBMEI** | **OBGD architecture** |
|---|---|---|
| Process | Ontology base (semantic description about processes, etc); semantic annotations in processes; Ontology-based vertical transformation | Component container (ontology-based process engine); OBGD semantic service bus (ontology-based horizontal transformation) |
| Service | Ontology base (semantic description about services, etc); semantic annotations in processes; | OBGD semantic service bus (STEP 2, and STEP OI-2) |
| Data | Ontology base (business terminology, etc); semantic annotations in processes; | OBGD semantic service bus (STEP 1, STEP 3, STEP OI-1 and STEP OI-3) |

In a word, our work has proposed and designed a federated approach to enterprise interoperability problems. The federated approach can achieve interoperability at conceptual and technical level in three interoperability concerns: data, service and process.

In our work, there are also some limitations. For example, ontology-based PBMEI and OBGD architecture all depend closely on ontology. The interoperability level they can achieve is determined by quality of ontology and capability of ontology mapping. In addition, OBGD architecture is based on OBGD SOAP, so its transport protocol is limited to SOAP.

In our study, there is still much engineering and scientific work to do in the future. In engineering aspect, software tools and platforms must be constructed to support ontology-based PBMEI and OBGD architecture. In addition, ATL should be extended to invoke external services during model transformation. This is necessary in ontology-based process transformation. In scientific research aspect, goal-driven service discovery and service provider discovery should be studied. They are used in OBGD semantic service bus. In OBGD architecture, ontology-based process engines should also be studied to support execution of interoperability processes. In addition, goal-driven composition of business processes should be studied. In our study, a goal model has been proposed, and the model will be a good start to construct an approach to automatically composite (collaborative) business processes.

# *ACRONYMS*

| | |
|---|---|
| API | Application Programming Interface |
| BAL | Business Action Language |
| BOM | Business Object Model |
| BPDM | Business Process Definition Metamodel<br>http://www.omg.org/technology/documents/br_pm_spec_catalog.htm |
| BPEL | Business Process Execution Language |
| BPM | Business Process Management |
| BPMS | Business Process Management System (Suite) |
| BPMN | Business Process Model and Notation |
| BPML | Business Process Modeling Language<br>http://xml.coverpages.org/bpml.html |
| BPMO | Business Process Modeling Ontology |
| BPSS | Business Process Specification Schema<br>http://www.ebxml.org/specs/ebBPSS.pdf |
| BRLDF | Business Rule Language Definition Framework |
| CBP | Collaborative Business Process |
| CM | Common Logic<br>http://cl.tamu.edu/ |
| CORBA | Common Object Request Broker Architecture<br>http://www.corba.org/ |
| CIM | Computation Independent Model |
| CI-DB | Configuration Information DataBase |
| CpBP | Cooperation Business Process |
| CrBP | Coordination Business Process |
| DAML | DARPA Agent Markup Language<br>http://www.daml.org/ |
| DBMS | DataBase Management System |
| ebXML | electronic business using XML |

| | |
|---|---|
| EAI | Enterprise Application Integration |
| ESB | Enterprise Service Bus |
| XOM | eXecution Object Model |
| XML | eXtensible Markup Language |
| GUI | Graphical User Interface |
| HTTP | HyperText Transfer Protocol |
| IRL | ILOG Rule Language |
| ICT | Information and Communication Technology |
| IT | Information Technology |
| IOPE | Input, Output, Precondition and Effect |
| IDE | Integrated Development Environment |
| JMS | Java Message Service<br>http://jcp.org/aboutJava/communityprocess/final/jsr914/index.html |
| KP | Knowledge Representation |
| MOF | Meta-Obejct Facility<br>http://www.omg.org/mof/ |
| MDA | Model-Driven Architecture<br>http://www.omg.org/mda/ |
| OMG | Object Management Group |
| ODM | Ontology Definition Metamodel |
| OIL | Ontology Inference Layer<br>http://www.cs.vu.nl/~frankh/abstracts/IEEE-IS01.html |
| OBGD | Ontology-Based and Goal-Driven |
| OKBC | Open Knowledge Base Connectivity<br>http://www.ai.sri.com/~okbc/ |
| OS | Operating System |

| | |
|---|---|
| OCML | Operational Conceptual Modeling Language<br>http://technologies.kmi.open.ac.uk/ocml/ |
| OASIS | Organization for the Advancement of Structured Information Standards |
| P2P | Peer-to-Peer |
| PIM | Platform Independent Model |
| PSM | Platform Specific Model |
| PIF | Process Interchange Format |
| PBMEI | Process-Based Method for Enterprise Interoperability |
| QoS | Quality of Service |
| REST | Representational State Transfer |
| RDFS | Resource Description Framework Schema<br>http://www.w3.org/TR/PR-rdf-schema |
| RDF | Resource Description Framework<br>http://www.w3.org/TR/PR-rdf-syntax |
| RES | Rule Execution Server |
| RIF | Rule Interchange Format<br>http://www.w3.org/2005/rules/wiki/RIF_Working_Group |
| SAWSDL | Semantic Annotations for WSDL<br>http://www.w3.org/2002/ws/sawsdl/ |
| sBPEL | Semantic BPEL |
| sEPC | Semantic Event-driven Process Chain |
| SUPER | Semantics Utilised for Process management within and between EnteRprise |
| SCA | Service Component Architecture<br>http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications |
| SMDA | Service Model Driven Architecture |
| SOA | Service Oriented Architecture |
| SHOE | Simple HTML Ontology Extensions<br>http://www.cs.umd.edu/projects/plus/SHOE/ |
| SOAP | Simple Object Access Protocol |

| | |
|---|---|
| SSB | Semantic Service Bus |
| TM | Topic Maps |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| TDS | Transparent Decision Service |
| TMS | Truth Maintenance System |
| UML | Unified Modeling Language |
| UDDI | Universal Description Discovery and Integration<br>http://www.oasis-open.org/committees/uddi-spec/ |
| OWL | Web Ontology Language<br>http://www.w3.org/TR/owl-features/ |
| WSCI | Web Service Choreography Interface |
| WSCL | Web Service Conversation Language<br>http://www.w3.org/TR/wscl10/ |
| WSMF | Web Service Modeling Framework |
| WSMO | Web Service Modeling Ontology<br>http://www.wsmo.org/ |
| WS-BPEL | Web Service-BPEL<br>http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel |
| WS-CDL | Web Service-Choreography Description Language<br>http://www.w3.org/TR/ws-cdl-10/ |
| WSDL | Web Services Description Language |
| WSFL | Web Services Flow Language<br>http://xml.coverpages.org/wsfl.html |
| WfMC | Workflow Management Coalition |
| WMS | Workflow Management System |
| WPDL | Workflow Process Definition Language |
| W3C | World Wide Web Consortium |

| | |
|---|---|
| XPDL | XML Process Definition Language<br>http://www.xpdl.org/nugen/p/xpdl/public.htm |
| XOL | XML-based Ontology exchange Language<br>http://www.ai.sri.com/pkarp/xol/ |

# *REFERENCES*

Abid, M. R. (2008). UML Profile for Goal-Oriented Modeling. Ottawa-Carleton Institute for Computer Science. Ottawa, Canada, University of Ottawa. **Master**

Alonso, G., U. Fiedler, et al. (1999). Wise: business to business e-commerce. Proceedings of 9th International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises, Sydney, NSW , Australia.

Alve, A. and A. Arkin (2007). Web Services Business Process Execution Language Version 2.0, OASIS, from http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

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

Ankolekar, A., F. Huch, et al. (2002). Concurrent Execution Semantics of DAML-S with Subtypes. Proceedings of the First International Semantic Web Conference on The Semantic Web, Springer-Verlag.

Ankolenkar, A., M. Burstein, et al. (2001). "DAML-S: Semantic Markup For Web Services." from http://www.daml.org/services/daml-s/2001/10/daml-s.html.

Avgerou, C. (2000). "Information systems: what sort of science is it?" Omega **28**(5): 567-579.

Bartonitz, M. (2010) "A brief history of business process management."

Batory, D. (2006). "Multilevel models in model-driven engineering, product lines, and metaprogramming." IBM Syst. J. **45**(3): 527-539.

Baude, F., I. Filali, et al. (2010). ESB federation for large-scale SOA. Proceedings of the 2010 ACM Symposium on Applied Computing. Sierre, Switzerland, ACM**:** 2459-2466

Bauer, B., S. Roser, et al. (2005). Adaptive Design of Cross-Organizational Business Processes Using a Model-Driven Architecture. Wirtschaftsinformatik 2005 (7th International Conference on WI). O. K. Ferstl, E. J. Sinz, S. Eckert and T. Isselhorst, Physica-Verlag HD**:** 103-121.

BEDNÁR, P., K. FURDÍK, et al. (2009). Design of a Semantic Service Bus for Networked Enterprises. AMIF 2009 - Ambient Intelligence Forum 2009. Hradec Králové, Czech Republic, from http://web.tuke.sk/fei-cit/furdik/publik/amif09_spike.pdf.

Bernauer, M., G. Kramler, et al. (2003). Specification of Interorganizational Workflows - A Comparison of Approaches. Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics.

Bernstein, P. A. (1996). "Middleware: a model for distributed system services." Commun. ACM **39**(2): 86-98.

Berre, A.-J., A. Hahn, et al. (2004). State of the art for interoperability architecture approaches, INTEROP VLab, from http://interop-vlab.eu/ei_public_deliverables/interop-noe-deliverables/dap-domain-architecture-and-platforms/D91/?searchterm=d9.1.

BizAgi. (2009). "Functional Description v9." from http://www.bizagi.com/docs/BizAgi%20Xpress%20Functional%20Description.pdf.

BonitaSoft. (2011). "BOS V5.5.1-User & Reference Guide." from http://www.bonitasoft.org/.

Borgida, A. (1996). "On the relative expressiveness of description logics and predicate logics." Artif. Intell. **82**(1-2): 353-367.

Bourey, J.-P., R. Grangel, et al. (2007). Deliverable DTG2.3 Report on Model-Driven Interoperability, from http://interop-vlab.eu/ei_public_deliverables/interop-noe-deliverables/dap-domain-architecture-and-platforms/D91/.

Brahmandam, P. (2008). Business Process Life-cycle, Princeton Blue, Inc., from http://www.princetonblue.com/whitepapers/business_process_lifecycle.pdf.

Brickley, D. and R. V. Guha. (1999). "Resource Description Framework (RDF) Schema Specification." from http://www.w3.org/TR/PR-rdf-schema.

Buelow, H., M. Das, et al. (2010) "Getting Started with Oracle BPM Suite 11gR1 A Hands-On Tutorial."

Burstein, M. H. (2004). "Dynamic invocation of semantic Web services that use unfamiliar ontologies." IEEE Intelligent Systems **19**(4): 67-73.

C4ISR-Interoperability-Working-Group (1998). Levels of Information Systems Interoperability (LISI). Department-of-Defense. Washington, DC

Cabral, L., J. Domingue, et al. (2006). IRS-III: A Broker for Semantic Web Services based Applications. The 5th International Semantic Web Conference (ISWC 2006), Athens, GA, USA.

Chandrasekaran, B. and T. Johnson (1993). "Generic tasks and task structures: history, critique and new directions." Second Generation Expert Systems: 232-272.

Chaudhri, V. K., A. Farquhar, et al. (1997). Open Knowledge Base Connectivity 2.0, Knowledge Systems Laboratory, Stanford University, from http://www.ai.sri.com/~okbc/.

Chaudhri, V. K., A. Farquhar, et al. (1998). OKBC: A Programmatic Foundation for Knowledge Base Interoperability. Proceedings of AAAI-98**:** 600-607

Chebbi, I., S. Dustdar, et al. (2006). "The view-based approach to dynamic inter-organizational workflow cooperation." Data Knowl. Eng. **56**(2): 139-173.

Chen, D. and N. Daclin (2006). Framework for enterprise interoperability. IFAC TC5.3 Workshop EI2N06, Bordeaux, France.

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

Chen, D., G. Doumeingts, et al. (2008). "Architectures for enterprise integration and interoperability: Past, present and future." Comput. Ind. **59**(7): 647-659.

Cimpian, E., T. Vitvar, et al. (2005). "Overview and Scope of WSMX. WSMX Deliverable D13.0, WSMX Final Draft v0.2." from http://www.wsmo.org/TR/d13/d13.0/v0.2.

Corcho, O., M. Fernández-López, et al. (2003). "Methodologies, tools and languages for building ontologies. Where is their meeting point?" Data & Knowledge Engineering **46**(1): 41-64.

Corcho, O. and A. Gómez-Pérez (2000). Evaluating knowledge representation and reasoning capabilities of ontology specification languages. Proceedings of the ECAI 2000 Workshop on Applications of Ontologies and Problem-Solving Methods. Berlin, from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.6867.

Czarnecki, K. and S. Helsen (2003). Classification of Model Transformation Approaches. OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture. Anaheim, from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.122.8124.

da Silva Santos, L. O. B., E. G. da Silva, et al. (2009). Towards a Goal-Based Service Framework for Dynamic Service Discovery and Composition. Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on.

da Silva Santos, L. O. B., L. F. Pires, et al. (2008). A Trust-Enabling Support for Goal-Based Services. The 9th International Conference for Young Computer Scientists.

Davenport, T. H. (1993). Process Innovation. Reengineering Work through Information technology. Boston.

Dayal, U., M. Hsu, et al. (2001). Business Process Coordination - State of the Art, Trends, and Open Issues. Proc. of the 27th VLDB Conference, Roma, Italy, Morgan Kaufmann.

Dean, M. and G. Schreiber. (2004, 10 February). "OWL Web Ontology Language Reference." from http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

Domingue, J., L. Cabral, et al. (2008). "IRS-III: A broker-based approach to semantic Web services." Web Semantics: Science, Services and Agents on the World Wide Web **6**(2): 109-132.

Domingue, J., L. Cabral, et al. (2004). Demo of IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services. Proceedings of the Workshop on WSMO Implementations (WIW 2004) Frankfurt, Germany.

Dumas, M., W. M. P. van der Aalst, et al. (2005). Process-Aware Information Systems: Bridging People and Software through Process Technology, Wiley& Sons.

ebXML. (2001, May 8). " EbXML Requirement Specification v1.06." Retrieved July, 2011, from http://www.ebxml.org/specs/ebREQ.pdf.

EIF (2004). European Interoperability Framework (White Paper). Brussels

ENIX (2006) "BPM Focus-An independent evaluation of BizAgi."

Farrell, J. and H. Lausen (2007). Semantic Annotations for WSDL and XML Schema, W3C, from http://www.w3.org/TR/sawsdl/.

Fensel, D. and C. Bussler (2002). "The Web Service Modeling Framework WSMF." Electronic Commerce Research and Applications **1** (2).

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Information and Computer Science. IRVINE, UNIVERSITY OF CALIFORNIA. **PhD**, from http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

Filipowska, A., M. Hepp, et al. (2009). Organisational Ontology Framework for Semantic Business Process Management. Business Information Systems. W. Abramowicz, Springer Berlin Heidelberg. **21:** 1-12.

Filipowska, A., M. Kaczmarek, et al. (2009). Organizational ontologies to support semantic business process management. Proceedings of the 4th International Workshop on Semantic Business Process Management. Heraklion, Greece, ACM**:** 35-42

Fisher, D. A. (2006). An emergent perspective on interoperation in systems of systems. Pittsburgh, Software Engineering Institute, Carnegie Mellon, from http://www.sei.cmu.edu/reports/06tr003.pdf.

Flouris, G., D. Plexousakis, et al. (2006). Evolving Ontology Evolution. SOFSEM 2006: Theory and Practice of Computer Science. J. Wiedermann, G. Tel, J. Pokorný, M. Bieliková and J. Štuller, Springer Berlin / Heidelberg. **3831:** 14-29.

Ford, T. C., J. M. Colombi, et al. (2007). A Survey on Interoperability Measurement. 12th ICCRTS “Adapting C2 to the 21st Century”.

Gärdenfors, P. (1992). Belief Revision: An Introduction, Cambridge University Press.

Gasevic, D., D. Djuric, et al. (2006). Model Driven Architecture and Ontology Development. Berlin Heidelberg, Springer-Verlag.

Geminiuc, K. (2006). "A Service-Oriented Approach to Business Rule Development." Retrieved July 8, 2011, from http://www.oracleimg.com/technetwork/articles/geminiuc-097530.html.

Goh, C. H. (1997). Representing and reasoning about semantic conflicts in heterogeneous information systems. Sloan School of Management, Massachusetts Institute of Technology. **PhD:** 180

Gómez-Pérez, A., M. Fernández-López, et al. (2004). Ontological Engineering, Springer.

Gottschalk, P. (2009). "Maturity levels for interoperability in digital government." Government Information Quarterly **26**(1): 75-81.

Graham, I. (2005). "Service Oriented Business Rules Management Systems." from http://www.trireme.com/whitepapers/Business%20rules/Trireme_Report_Service_Oriented_Business_Rules_Management_Systems_Ver2b.pdf.

Grangel, R., R. Chalmeta, et al. (2008). A Proposal for Goal Modelling Using a UML Profile. Enterprise Interoperability III. K. Mertins, R. Ruggaber, K. Popplewell and X. Xu, Springer London**:** 679-690.

Gruber, T. R. (1993). "A translation approach to portable ontology specifications." Knowl. Acquis. **5**(2): 199-220.

Guarino, N. (1995). "Formal ontology, conceptual analysis and knowledge representation." Int. J. Hum.-Comput. Stud. **43**(5-6): 625-640.

Haase, P. and L. Stojanovic (2005). Consistent Evolution of OWL Ontologies. The Semantic Web: Research and Applications. A. Gómez-Pérez and J. Euzenat, Springer Berlin / Heidelberg. **3532:** 91-133.

Hakimpour, F., D. Sell, et al. (2005). Semantic Web service composition in IRS-III: the structured approach. E-Commerce Technology, 2005. CEC 2005. Seventh IEEE International Conference on.

Halpin, H. (2004). The Semantic Web: The Origins of Artificial Intelligence Redux. Third Intern. Workshop on the History and Philosophy of Logic, Mathematics, and Computation. Donostia San Sebastian, Spain

Hamilton, J. A. and M. G. A. Catania (2003). A Practical Application of Enterprise Architecture for Interoperability. The 2003 International Conference on Information Systems and Engineering ISE 2003, Quebec, Ca.

Hammer (1990). "Reengineering Work: Don't automate, Obliberate." Harvard Business Review **68**: 104-112.

Heflin, J., J. Hendler, et al. (1999). Coping with Changing Ontologies in a Distributed Environment. Proceedings of AAAI-99 Workshop on Ontology Management.

Hendler, J. (2001). "Agents and the Semantic Web." IEEE Intelligent Systems **16**(2): 30-37.

Héon, M., G. Paquette, et al. (2008). Transformation of Semi-formal Models into Ontologies According to a Model Driven Architecture. 2èmes Journées Francophones sur les Ontologies. Lyon, France

Horridge, M. and S. Bechhofer (2009). The OWL API: A Java API for Working with OWL 2 Ontologies. OWLED, CEUR-WS.org.

Horridge, M. and S. Bechhofer (2010). "The OWL API: A Java API for OWL Ontologies." Semantic Web Journal **2**(1): 11-21.

Horrocks, I., P. F. Patel-Schneider, et al. (2003). "From SHIQ and RDF to OWL: the making of a Web Ontology Language." Web Semantics: Science, Services and Agents on the World Wide Web **1**(1): 7-26.

IBM. (2009). "Deciding on an execution mode." Retrieved July 8, 2011, from http://publib.boulder.ibm.com/infocenter/brjrules/v7r0/index.jsp?topic=/ilog.rules.jrules.doc/Content/Business_Rules/Documentation/_pubskel/JRules/ps_JRules_Global963.html.

IEC-TC65/290/DC (2002). Device Profile Guideline, TC65: Industrial Process Measure-ment and Control

IEEE (1990). IEEE : Standard Computer Dictionary- A Compilation of IEEE Standard Computer Glossaries. New York, NY, Institute of Electrical and Electronics Engineers

ILOG. (2005a, Sept 31). "ILOG JRules Performance Analysis and Capacity Planning." Retrieved July 8, 2011, from http://logic.stanford.edu/POEM/externalpapers/iRules/jrules_cap_wp.pdf.

ILOG. (2005b, March). "ILOG JRules: leading the ways in Business Rule Management Systems." Retrieved July 8, 2011, from http://www.docslibrary.com/leading-the-way-in-business-rule-management-systems.

ILOG. (2006, 06). "Decision Services: the next SOA challenge." Retrieved July 8, 2011, from http://logic.stanford.edu/POEM/externalpapers/WP-SOA.pdf.

IONA (2006). The evolution from EAI to ESB, from http://www.immagic.com/eLibrary/ARCHIVES/GENERAL/IONA_IE/I070809E.pdf.

ISO-14258 (1998). Industrial automation systems and integration – Concepts and rules for enterprise models, ISO TC 184 SC5

Jarrar, M. and R. Meersman (2008). Ontology Engineering -The DOGMA Approach. Advances in Web Semantics, Springer. **I**.

JBPMCommunity (2011). Jbpm User Guide, from http://docs.jboss.org/jbpm/v5.1/userguide/.

Jenz, D. E. (2003). Ontology-based business process management-the vision statement. Erlensee, Jenz&Partner GmbH, from http://www.bpiresearch.com/Resources/WP_BPMVision.pdf.

Jokhio, M. S. (2009). Goal-Based Testing of Semantic Web Services. Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering, IEEE Computer Society.

Kalfoglou, Y. and M. Schorlemmer (2003). "Ontology mapping: the state of the art." Knowl. Eng. Rev. **18**(1): 1-31.

Karastoyanova, D., B. Wetzstein, et al. (2007). Semantic Service Bus: Architecture and Implementation of a Next Generation Middleware. the Second International ICDE Workshop on Service Engineering SEIW 2007, IEEE Computer Society.

Kiefer, C., A. Bernstein, et al. (2007). Semantic Process Retrieval with iSPARQL. Proceedings of the 4th European conference on The Semantic Web: Research and Applications. Innsbruck, Austria, Springer-Verlag**:** 609-623

Kifer, M., G. Lausen, et al. (1995). "Logical foundations of object-oriented and frame-based languages." J. ACM **42**(4): 741-843.

Kosanke, K. (2006). Interoperability: INTEROP PROJECT- overview and results. INTEROP Workshop. Helsinki, Finland, from http://www.cimosa.de/Modelling/01-1_Interoperability_INTEROP_Project_-_Overview_and_Results.pdf.

Lamsweerde, A. v. and E. Letier (2004). From Object Orientation to Goal Orientation: A Paradigm Shift for Requirements Engineering. Radical Innovations of Software and Systems Engineering in the Future. M. Wirsing, A. Knapp and S. Balsamo, Springer Berlin / Heidelberg. **2941:** 153-166.

Lapouchnian, A. (2005). "Goal-Oriented Requirements Engineering: An Overview of the current Research." from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.399&rep=rep1&type=pdf.

Lassila, O. and R. Swick. (1999). "Resource Description Framework (RDF) Model and Syntax Specification." from http://www.w3.org/TR/PR-rdf-syntax.

Lin, Y. and H. Ding (2005). Ontology-based Semantic Annotation for Semantic Interoperability of Process Models. Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-1 (CIMCA-IAWTIC'06) IEEE Computer Society.

Ling, C. and L. Xin (2009). Achieving Business Agility by Integrating SOA and BPM Technology. 2009 International Forum on Information Technology and Applications, Chengdu, China, IEEE Computer Society.

Liu, H. (2008). 基于 ESB 的企业信息集成方法和技术研究 (RESEARCH ON METHODS AND TECHNIQUES FOR ESB-BASED ENTERPRISE INFORMATION INTEGRATION). Nanjing, Southeast University. **Master**, from http://d.wanfangdata.com.cn/Thesis_Y1385661.aspx.

Liu, H. and J.-P. Bourey (2010). Transformation from a Collaborative Process to Multiple Interoperability Processes. Enterprise Interoperability IV (Conference I-ESA'10). K. Popplewell, J. Harding, R. Poler and R. Chalmeta, Springer London**:** 135-144.

Losavio, F., D. Ortega, et al. (2002). Modeling EAI [Enterprise Application Integration]. Computer Science Society, 2002. SCCC 2002. Proceedings. 22nd International Conference of the Chilean.

Luke, S. and J. Heflin (2000). SHOE 1.01: Proposed Specification. University of Maryland, from http://www.cs.umd.edu/projects/plus/SHOE/spec.html.

MacGregor, R. M. (1991). "Inside the LOOM description classifier." SIGART Bull. **2**(3): 88-92.

McIlraith, S. A. and D. L. Martin (2003). "Bringing Semantics to Web Services." IEEE Intelligent Systems **18**(1): 90-93.

McIlraith, S. A., T. C. Son, et al. (2001). "Semantic Web Services." IEEE Intelligent Systems **16**(2): 46–53.

Mehandjiev, N., I. D. Stalker, et al. (2006). Interoperability Contributions of CrossWork. Interoperability of Enterprise Software and Applications. D. Konstantas, J.-P. Bourrières, M. Léonard and N. Boudjlida, Springer London**:** 449-450.

Miller, J. and J. Mukerji (2003). MDA Guide Version 1.0.1, Object Management Group, from http://www.omg.org/cgi-bin/doc?omg/03-06-01.

Morris, E., L. Levine, et al. (2004). System of Systems Interoperability (SOSI): final report. Pittsburgh, Software Engineering Institute, Carnegie Mellon

Motta, E. (1999). Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving. Amsterdam, The Netherlands, IOS Press.

Narayanan, S. and S. A. McIlraith (2002). Simulation, verification and automated composition of web services. Proceedings of the 11th international conference on World Wide Web, Honolulu, Hawaii, USA, ACM.

Natis, Y. V. (2003, April 16). "Service-Oriented Architecture Scenario." Retrieved July, 2011, from http://www.gartner.com/DisplayDocument?doc_cd=114358.

NATO (2003). NATO C3 Technical Architecture (NC3TA) Reference Model for Interoperability. The Hague, The Netherlands

NEHTA (2005). Towards an interoperability framework, from http://www.nehta.gov.au/component/docman/doc_download/26-towards-an-interoperability-framework-v18.

Norton, B., L. Cabral, et al. (2009). Ontology-Based Translation of Business Process Models. Proceedings of the 2009 Fourth International Conference on Internet and Web Applications and Services, IEEE Computer Society**:** 481-486

Noy, N. F. (2004). "Semantic integration: a survey of ontology-based approaches." SIGMOD Rec. **33**(4): 65--70.

Noy, N. F. and M. Klein (2004). "Ontology Evolution: Not the Same as Schema Evolution." Knowledge and Information Systems **6**(4): 428-440.

OASIS-ebXML. (2001, Sept ). "ebXML Technical Architecture Specification v1.0.4." Retrieved July, 2009, from http://www.ebxml.org/specs/ebTA.pdf.

OASIS-ebXML. (2002a, Sept 23). "ebXML Collaboration-Protocol Profile and Agreement Specification V2.0." Retrieved Sept, 2009, from http://www.oasis-open.org/committees/ebxml-cppa/documents/ebCPP-2_0.pdf.

OASIS-ebXML. (2002b, April). " Registry Services Specification v2.0." Retrieved Sep, 2009, from http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebRS.pdf.

OASIS-ebXML. (2006, 21 Dec). "ebXML Business Process Specification Schema Technical Specification v2.0.4." Retrieved Sept, 2009, from http://docs.oasis-open.org/ebxml-bp/2.0.4/OS/spec/ebxmlbp-v2.0.4-Spec-os-en.pdf.

OASIS (2006). Reference Model for Service Oriented Architecture 1.0, from http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf.

OMG-ODM (2009). Ontology Definition Metamodel: OMG Adopted Specification, from http://www.omg.org/spec/ODM/1.0/PDF.

OMG (2006). CORBA Component Model v4.0, from http://www.omg.org/spec/CCM/4.0/PDF

OMG (2008). Business Process Definition Metamodel, Process Definition, from http://www.omg.org/cgi-bin/doc?dtc/2008-05-09.

OMG. (2011, 03 January). "Business Process Model and Notation (BPMN) Version 2.0." from http://www.omg.org/spec/BPMN/2.0.

Paolucci, M., A. Ankolekar, et al. (2003). The DAML-S Virtual Machine. The Semantic Web - ISWC 2003. D. Fensel, K. Sycara and J. Mylopoulos, Springer Berlin / Heidelberg. **2870:** 290-305.

Patel-Schneider, P. F., P. Hayes, et al. (2004, 10 February). "OWL Web Ontology Language Semantics and Abstract Syntax." from http://www.w3.org/TR/2004/REC-owl-semantics-20040210/.

Polyak, S., J. Lee, et al. (1998). Applying the Process Interchange Format(PIF) to a Supply Chain Process Interoperability Scenario. ECAI'98: Workshop on Applications of Ontologies and Problem Solving Methods, Brighton, England.

Puschmann, T. and R. Alt (2001). Enterprise application integration-the case of the Robert Bosch Group. System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on.

Qiming, C. and H. Meichun (2001). Inter-enterprise collaborative business process management. Proceedings. 17th International Conference on Data Engineering, Germany.

Rajsiri, V., J.-P. Lorré, et al. (2008). Collaborative Process Definition Using An Ontology-Based Approach. Pervasive Collaborative Networks. L. Camarinha-Matos and W. Picard, Springer Boston. **283:** 205-212.

Rajsiri, V., J. P. Lorre, et al. (2009). Prototype of an Ontology-Based Approach for Collaborative Process Specification. Proceedings of the 2009 International Conference on Interoperability for Enterprise Software and Applications China, IEEE Computer Society**:** 53-59

Reiersgaard, N., H. Salvesen, et al. (2005). EAI Implementation Project and Shakedown: An Exploratory Case Study. System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on.

Roman, D., H. Lausen, et al. (2006, 21 October). "D2v1.3. Web Service Modeling Ontology (WSMO)." from http://www.wsmo.org/TR/d2/v1.3/.

Roser, S. and B. Bauer (2005). A Categorization of Collaborative Business Process Modeling Techniques. Proceedings of the Seventh IEEE International Conference on E-Commerce Technology Workshops, IEEE Computer Society**:** 43-54

Roser, S. and B. Bauer (2006). Ontology-Based Model Transformation. Satellite Events at the MoDELS 2005 Conference. J.-M. Bruel, Springer Berlin / Heidelberg. **3844:** 355-356.

Russell, S. and P. Norvig (2003). Artificial Intelligence: A Modern Approach, Pearson Education.

Salomie, I., V. R. Chifu, et al. (2008). SAWS: A tool for semantic annotation of web services. IEEE International Conference on Automation, Quality and Testing, Robotics.

Schantz, R. E. and D. C. Schmidt (2001). "Middleware for distributed systems: Evolving the common structure for network-centric applications." Encyclopedia of Software Engineering: 1-9.

Schmelzer, R. (2007, November 28). "The seven levels of loose coupling." Retrieved July, 2011, from http://www.zapthink.com/2007/11/28/the-seven-levels-of-loose-coupling/.

Shafiq, O., M. Moran, et al. (2007). Investigating Semantic Web Service Execution Environments: A Comparison between WSMX and OWL-S Tools. Internet and Web Applications and Services, 2007. ICIW '07. Second International Conference on.

Shet, A. P. and M. Rusinkiewicz (1993). "On Transactional Workflows." Data Engineering Bulletin **16/2**.

Sheth, A., M. Rusinkiewicz, et al. (1992). Using polytransactious to manage interdependent data. Database Transaction Models for Advanced Applications. A. K. Elmagarmid. San Mateo, California.

Silver, B. (2009 ) "Things to Love About BPMN 2.0."

Smith, M. K., C. Welty, et al. (10 February). "OWL Web Ontology Language Guide." from http://www.w3.org/TR/2004/REC-owl-guide-20040210/.

Stegwee, R. A. and B. D. Rukanova (2003). Identification of Different Types of Standards for Domain-Specific Interoperability. Proceedings of the Workshop on Standard Making: A Critical Research Frontier for Information Systems. J. L. King and K. Lyytinen. Seattle, WA**:** 161- 170, from http://www.si.umich.edu/misq-stds/proceedings/139_161-170.pdf.

Stineman, B. (2009, September). "Why Business Rules?: A Case for Business Users of Information Technology." Retrieved July 8, 2011, from ftp://public.dhe.ibm.com/common/ssi/ecm/en/wsw14061usen/WSW14061USEN.PDF .

Stojanovic, L., A. Maedche, et al. (2003). Ontology evolution as reconfiguration-design problem solving. Proceedings of the 2nd international conference on Knowledge capture. Sanibel Island, FL, USA, ACM**:** 162-171

Stollberg, M. and M. Hepp (2006). Goal Description Ontology. Deliverable D3.10, DIP

Stollberg, M. and B. Norton (2007). A Refined Goal Model for Semantic Web Services. Proceedings of the 2nd International Conference on Internet and Web Applications and Services, IEEE Computer Society.

Supakkul, S. and L. Chung (2005). A UML profile for goal-oriented and use case-driven representation of NFRs and FRs. Software Engineering Research, Management and Applications, 2005. Third ACIS International Conference on.

SUPER-Project (2007). D.1.1. Business Process Ontology Framework, SUPER Deliverable,

Swartout, W. R. and A. Tate (1999). "Guest editors' introduction: Ontologies." IEEE Intelligent Systems **14**(1): 18-19.

Sycara, K., M. Paolucci, et al. (2003). "Automated Discovery, Interaction and Composition of Semantic Web Services." Journal of Web Semantics **1**(1): 27-46.

Thatte, S. (2001, May). "XLANG: Web services for Business Proccess Design." Retrieved July, 2011, from http://xml.coverpages.org/XLANG-C-200106.html.

The-OWL-Services-Coalition. (2003, Dec 27). "OWL-S: Semantic Markup for Web Services." from http://www.daml.org/services/owl-s/1.0/owl-s.html.

Tim, B.-L., J. Hendler, et al. (2001). "The Semantic Web." Scientific American Magazine **284**(5): 34-43.

Tolk, A., S. Diallo, et al. (2007). "Applying the Levels of Conceptual Interoperability Model in Support of Integratability, Interoperability, and Composability for System-of-Systems Engineering." Systemics, Cybernetics and Informatics **5**(5): 65-74.

Tolk, A. and J. A. Muguira (2003). The levels of conceptual interoperability model. Proceedings of the 2003 Fall Simulation Interoperability Workshop, Orlando, Florida.

Touzi, J. (2007). Aide à conception de système d'information Collaboratif support de l'interopérabilité des enterprises. Centre de Génie Industriel, Ecole des Mines d'Albi Carmaux. **PhD**

Truptil, S. (2011). Etude de l'approche de l'interopérabilité par médiation dans le cadre d'une dynamique de collaboration appliquée à la gestion de crise. Centre de Génie Industriel. Toulouse, Université de Toulouse - Mines Albi. **PhD**

US-Census-Bureau (2007). North American Industry Classification System (NAICS), from http://www.census.gov/cgi-bin/sssd/naics/naicsrch?chart=2007.

Uschold, M. and M. Gruninger (2004). "Ontologies and semantics for seamless connectivity." SIGMOD Rec. **33**(4): 58-64.

Uschold, M. and M. Grüninger (1996). "Ontologies: principles, methods, and applications." Knowledge Engineering Review **11**(2): 93-155.

Van der Aalst, W. M. P. (1999). "Process-oriented architectures for electronic commerce and interorganizational workflow." Information Systems **24**: 639–671.

van der Aalst, W. M. P., A. H. M. ter Hofstede, et al. (2003). Business Process Management: A Survey. Proceedings of the International Conference of Business Process Management, Eindhoven, The Netherlands, Springer.

van der Aalst, W. M. P. and M. Weske (2001). The P2P approach to Interorganizational Workflows. Proceeding of the 13th Intl. Conf. on Advanced Information Systems Engineering (CAiSE'01), Berlin.

W3C-RIF-WG. (2005). "List of Rule Systems." Retrieved July 8, 2011, from http://www.w3.org/2005/rules/wg/wiki/List_of_Rule_Systems.

w3C-RIF-WG (2008). RIF Use Cases and Requirements, from http://www.w3.org/TR/2008/WD-rif-ucr-20081218/.

W3C (2005). Web Service Semantics - WSDL-S, from http://www.w3.org/Submission/WSDL-S/.

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

Wache, H., T. Vögele, et al. (2001). Ontology-based integration of information --- a survey of existing approaches. IJCAI--01 Workshop: Ontologies and Information Sharing, Seattle, WA.

Wang, H. and J. N. K. Liu (2009). Analysis of Semantic Heterogeneity Using a New Ontological Structure Based on Description Logics. Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09. Sixth International Conference on.

Wang, W., A. Tolk, et al. (2009). The levels of conceptual interoperability model: applying systems engineering principles to M\&S. Proceedings of the 2009 Spring Simulation Multiconference. San Diego, California, Society for Computer Simulation International**:** 1-9

Weigand, H. and A. H. H. Ngu (1998). "Flexible specification of interoperable transactions." Data & Knowledge Engineering **25**.

Xu, X., T. Mo, et al. (2007). SMDA: A Service Model Driven Architecture. Enterprise Interoperability II. London, Springer-Verlag**:** 291-302.

Xu, Z. and Y. C. Lee (2002). Semantic Heterogeneity Of Geodata ISPRS Commission IV, WG IV/2, Ottawa, Canada.

# *APPENDIX*

# Appendix A: Overview of Business Rule Management System

## Introduction

Rule-languages and rule-based systems have played seminal roles in the history of computer science and the evolution of information technology. From expert systems to deductive databases, the theory and practice of automating inference based on symbolic representations have had a rich history and continue to be a key technology driver (w3C-RIF-WG 2008). (W3C-RIF-WG 2005) lists many rule-based systems, such as FLORA-2, Hoolet and JenaRules.

Business Rule Management System (BRMS) is one of the rule-based systems. As we know, the periods of an application development and update are much longer than that of business rule development and update. BRMS mainly solves the **mismatching** between the application development life cycle and the business rule management life cycle.

Traditionally, the process of building an application system often requires freezing business rules into software systems. This limits business sponsors' flexibility to adapt their operations to dynamic market conditions, individual customer demands or regulatory environment changes (Stineman 2009). Furthermore, the traditional programming style about business rules also makes developers not able to **reuse** business rules in other applications. If a business rule is changed, it can't be updated automatically in other relevant applications. This will make business rule **inconsistent**. However, BRMSs can get rid of the above disadvantages. BRMSs enable business people to define their business policies[1] and business rules. BRMSs also provide clear communication between policy managers[2] and developers. BRMS has a business rule repository for all the application systems in a whole enterprise. The repository guarantees **reusability** and **consistency** of business rules and meanwhile it avoids redundancy of business rules.

Nowadays, there are many BRMSs (Graham 2005), such as HaleyAutority, ILOG JRules, Blaze Advisor, Drools[3] and so on. ILOG JRules is much more widely used because of its comprehensive feature set, reliability, customizability, extensibility, trace record and complete offering (ILOG 2005b). Therefore this appendix will analyze BRMS based on ILOG JRules.

## BRMS

Generally speaking, a BRMS must at least have its own rule language, rule editor, rule execution/management system and rule repository, like Figure A-1. ILOG JRules, as the market leading of BRMS, has done much more. ILOG JRules V7.0 contains a collection of modules that work together to provide a comprehensive BRMS, see Figure A-2. JRules V7.0 has three broad areas:

- *Business rule applications Development*: it is focused on design, Java development, rule project development, and troubleshooting about rules. It is supported by Rule Studio and Decision Validation services;
- *Business rule management and authoring*: it is in charge of creation, maintenance, testing, simulation and publication for business rules. It is supported primarily by Rule Team Server and Rule Solutions for Office;

[1] Business policies, business rules and their relationships are defined in (Stineman 2009).
[2] (ILOG 2006) defines architects, developers, business analysts, policy managers and system administrators. It also defines their responsibilities.
[3] http://www.jboss.org/drools/

- *Enterprise application*: it executes, integrates, monitors and audit business rules. It is supported mainly by rule execution server.

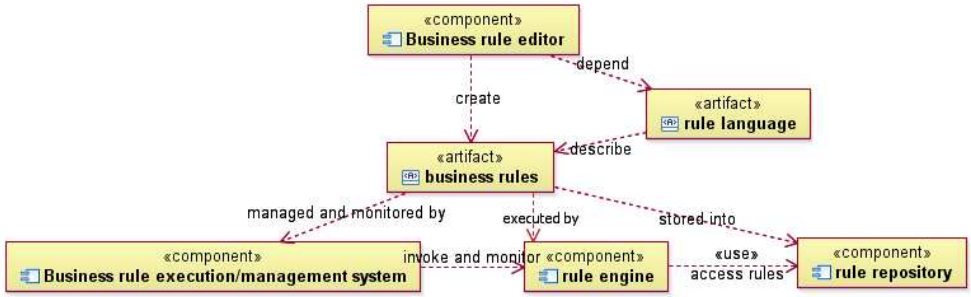



**Figure A-1. Basic Components for BRMS**

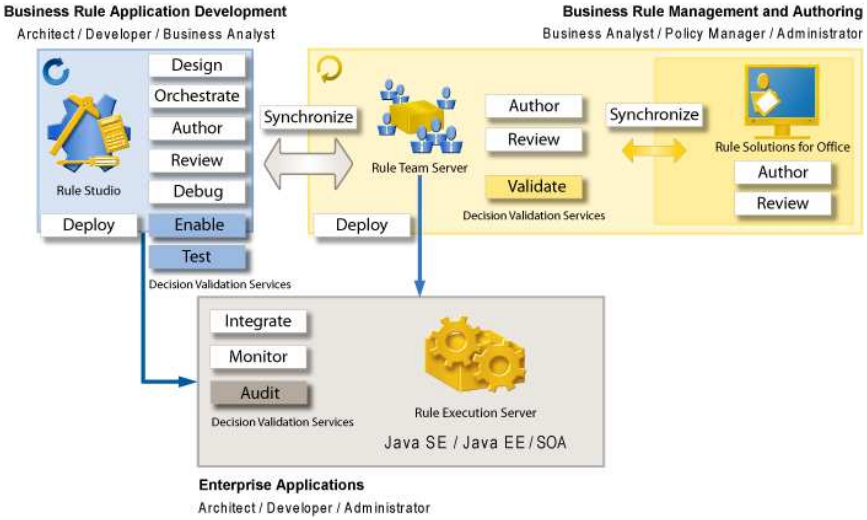



**Figure A-2. Components in ILOG JRules V7.0[4]**

In order to support business rule application development, JRules proposes its own conceptual model described in four figures (from Figure A-3 to Figure A-6). Figure A-3 shows two object models and vocabulary that are used by rule artifacts. Figure A-4 shows two rule languages in ILOG JRules: BAL (Business Action Language) and IRL (ILOG Rule Language). They are also defined by BRDLF (Business Rule Definition Language Framework). Figure A-5 shows two rule models in JRules: business and technical rule artifacts. The business rule artifacts (not executable) are created in BAL with the help of BOM (Business Object Model) and vocabulary; the technical rule artifacts (executable) are created in IRL with the help of XOM (eXecution Object Model). In fact, technical rule artifacts can result from transformation of business rule artifacts. Figure A-6 describes the contents of a ruleset archive. The archive will be deployed in BRMS.

[4] http://publib.boulder.ibm.com/infocenter/brjrules/v7r1/index.jsp?topic=/com.ibm.websphere.ilog.jrules.doc/Content/Business_Rules/Documentation/_pubskel/JRules/ps_JRules_Global7.html

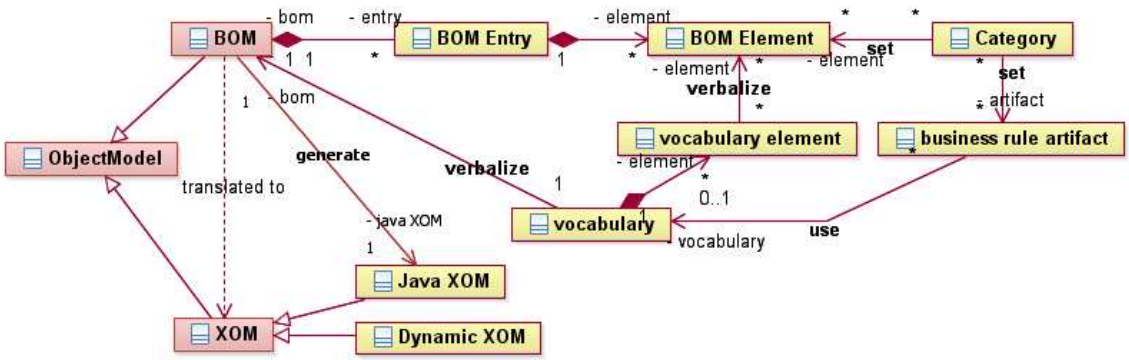



**Figure A-3. Relationships between BOM and XOM in JRules**


**Figure A-4. Rule Languages in JRules**

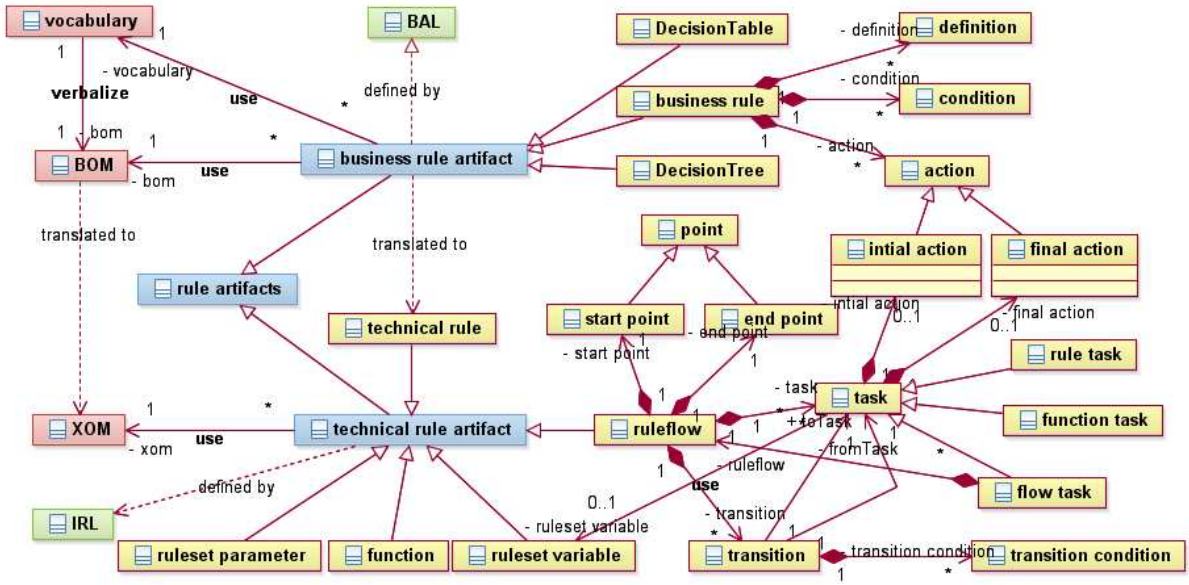



**Figure A-5. Relationships between Business and Technical Rule Artifacts in JRules**

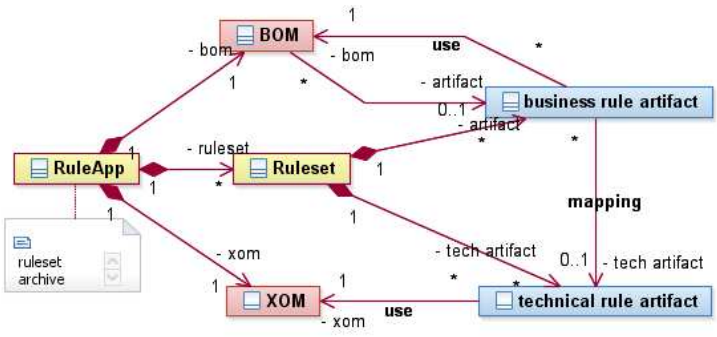



**Figure A-6. Ruleset Archive in JRules**

After establishing the concept model, how does JRules support business rule application development? The following steps are the simplest process: Firstly, business analysts create the BOM, vocabulary and business rule artifacts according to business policies with the help of JRules Rule Studio; then the business rule artifacts are translated automatically to technical

rule artifacts in terms of their mapping definition. Of course, the relevant BOMs are also translated into XOMs. Then ruleflows will be orchestrated if necessary. After debug and test, rulesets and relevant object models will be archived (Figure A-6) and deployed on Rule Execution Server. After passing the test, rulesets will be made available for relevant applications. The rulesets can also be maintained in Rule Team Server by policy managers after their deployment.

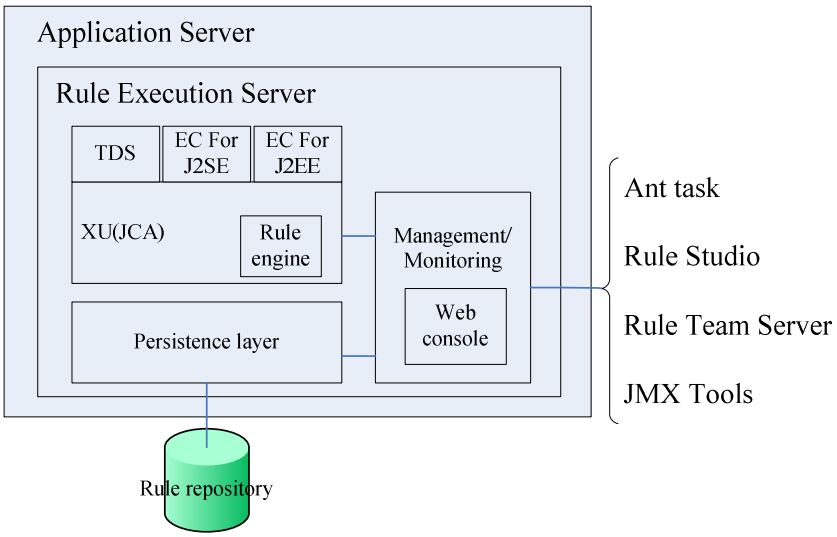



**Figure A-7. A Simple Architecture of the Rule Execution Server in JRules**

For further study of business rule execution, the rule execution server and rule engine of JRules are analyzed. Figure A-7 depicts a simple architecture of the Rule Execution Server[4] of JRules. In Figure A-7, the server provides other enterprise information systems with several invocation ways to execute required rules: the interface of J2SE, the interface of J2EE, the interface of web service, the interface of transparent decision service (TDS), etc. The server also provides management/monitoring interfaces invoked by external applications (Ant task, Rule Studio, Rule Team Server, JMX Server). The core of the server is the execution unit (XU) and the persistence layer. When the server executes rules, XU will retrieve required ruleset from the persistence layer and then it will invoke the rule engine to execute the ruleset. Figure A-8 depicts the runtime environment of the rule engine in JRules.

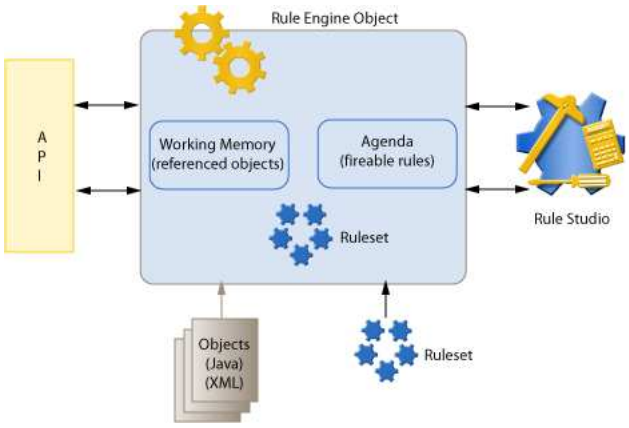



**Figure A-8. Runtime Environment of Rule Engine in JRules[5]**

[5] http://publib.boulder.ibm.com/infocenter/brjrules/v7r1/index.jsp?topic=/com.ibm.websphere.ilog.jrules.doc/Content/Business_Rules/Documentation/_pubskel/JRules/ps_JRules_Global737.html

In Figure A-8, a rule engine object has a working memory and an agenda. *Working memory* contains the objects used in the ruleset execution, such as application objects and parameter objects. *Agenda* contains the rule instances which are eligible to be fired in an order. If a *rule engine object* is created, it will load the relevant ruleset (and compile the ruleset to java byte code if required) and it will receive the application objects and parameter objects. When application objects[6] are added into the rule engine, two things happen[5]:

1) References to the native Java application objects are added to the rule engine. These references enable the rule engine to monitor the application objects;
2) The conditions of all rules in the ruleset archive are evaluated. If the conditions of a rule are met, the rule is declared eligible to be executed (added into agenda) or fired.

The following UML Class Diagram indicates the partial concepts used in the rule engine of JRules.

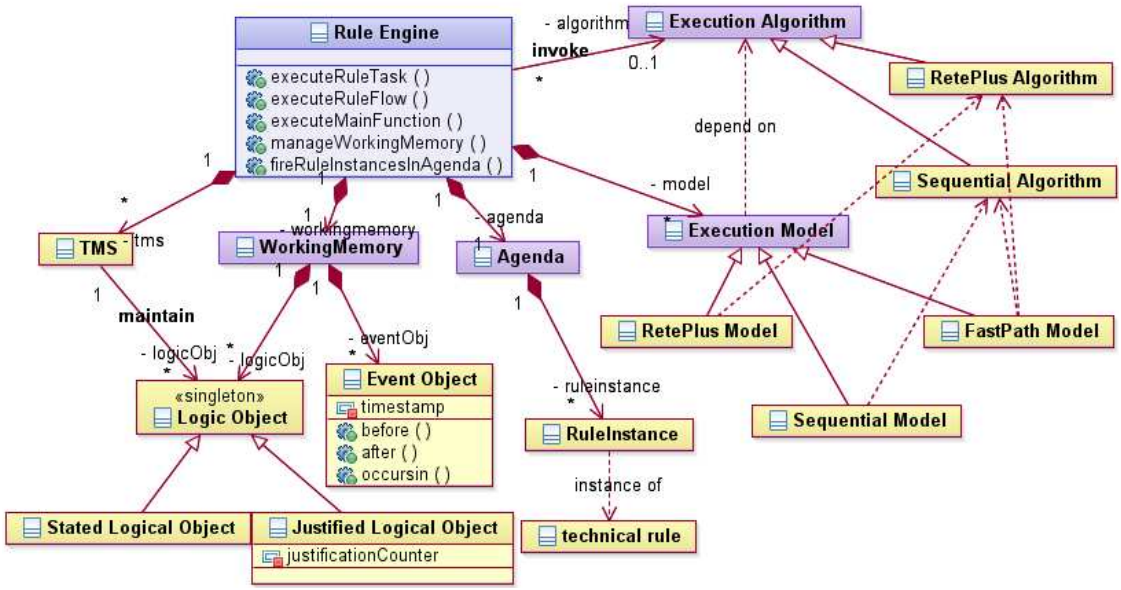



**Figure A-9. Concepts in the Rule Engine of JRules**

In Figure A-9, the rule engine has three execution models[7]: RetePlus Model, Sequential Model and FastPath Model. The execution mode determines which rules to fire and in what sequence. RetePlus Model follows the RetePlus algorithm. Sequential Model follows the sequential algorithm. FastPath model is a sequential mode of execution, but it also detects semantic relations between rule tests during the pattern matching process, like RetePlus. So FastPath Model depends on both of Rete algorithm and Sequential algorithm.

Each execution model has its own advantages and disadvantages. To make the best choice of the execution model, the following questions need to be answered: what type of application do your rules implement, what types of objects are used by your rules, what is the impact of rule actions, what sort of tests do you find in rule conditions, and what priorities have you set on your rules. (IBM 2009) has proposed some advice about how to make a good decision.

After analyzing rule engines, the following question is how to evaluate rule engines. There are several academic benchmarks, such as Manners, Waltz or Fibonacci, but they are not very representative of most eBusiness applications(ILOG 2005a). Thus (ILOG 2005a) tests its rule engine under some conditions determined by itself, such as hardware configuration, test

[6] Application data can be generated from java program, execution of rules, etc.
[7] http://publib.boulder.ibm.com/infocenter/brdotnet/v7r1/index.jsp?topic=/com.ibm.websphere.ilog.brdotnet.doc/Content/Business_Rules/Documentation/_pubskel/Rules_for_DotNET/ps_RFDN_Global261.html

content, working memory, the number of rules and average evaluation time. (ILOG 2005a) also offers some test results about several activities of its rule engine under different execution model.

After have evaluated rule engines, how to optimize rule engines will be interesting. (ILOG 2005a) proposed some optimization methods about rule engine execution, such as adopting auto-hashing, hashers and finders, dynamic rule compilation, selecting a proper execution model for ruleset, using rule task runner, configuring rule engine (caching, sharing working memory, multi- threading and pooling, running in parallel, etc).

## Rule Interchange Format

Rule Interchange Format (RIF) should enable interchange of rules[8]. The basic usage scenario[9] for RIF is as follows (see Figure A-10):

- a producer agent produces a set of rules in some rule language, serializes it in RIF and publishes the resulting RIF document;
- a consumer agent gets the RIF document, deserializes it into some rule language and deals with it for some purpose.

The general architecture for a RIF-based interchange can thus be represented as in Figure A-11. Nowadays, Rule Interchange Format (RIF) is still being constructed by W3C RIF Working Group (W3C RIP-WG).

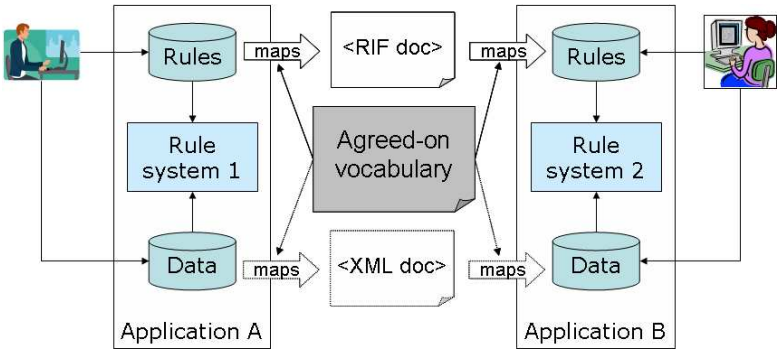



**Figure A-10. A Basic Usage Scenario for RIF[10]**

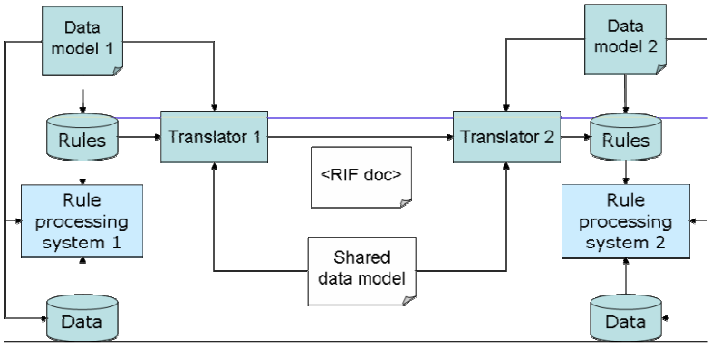



**Figure A-11. General Architecture for RIF-based Interchange[11]**
(The shared data model can be an XML schema, an OWL ontology or RDF vocabulary, etc)

[8] http://www.w3.org/2005/rules/wg/wiki/UCR/What_is_a_Rule_Interchange_Format_And_Why_Create_One
[9] Usage scenarios may vary in many ways with respect to the basic scenario. The interchange model may be point-to-point, one-to-many and many-to-one. Another dimension is whether the interchange works in push mode (send/receive or broadcast/receive) or in pull mode (publish/retrieve).
[10] http://www.w3.org/2005/rules/wg/wiki/Arch/Using_RIF
[11] http://www.w3.org/2005/rules/wg/wiki/UCR/Basic_RIF_Processing_Model

## Relationship between BRMS and SOA/BPEL

SOA as an enterprise integration strategy has been widely used. In order to support SOA, BRMS must be able to be accessed through SOA protocols (e.g., SOAP, WSDL) by other information systems and meanwhile it must also be able to access the services deployed by other information systems. For example, BRMS must be able to publish its rules as services; it must also be able to make its rules invoke the services published out of its own system.

Currently, ILOG JRules can publish its rules as services through three ways: a simple web service publish tool, hosted transparent decision service[12] and monitored transparent decision service and it can also make the action part of the rules invoke the web services. Further more, JRules also extends SOA. As we know, traditionally in SOA, services are not visible, and they are black boxes. However, in JRules, all services based on rules are visible and can be changed easily and rapidly and they can also be monitored and audited (ILOG 2006).

WS-BPEL (Web Service-Business Process Execution Language) is one of the key standards improving the wide adoption of SOA. WS-BPEL engine can make enterprises automatically execute business processes which are composed of services. But if business rules aren't separated from business processes, they will also bring the same problems elaborated in Section "Introduction" into the development and maintenance of business processes. So rule engines can be introduced into the SOA environment where BPEL engine is running. In this case, when a BPEL process reaches a decision point, it can invoke the service corresponding to relevant business rules. It is very clear that rule engine and WS-BPEL are complementary technologies. (Geminiuc 2006) provided an architecture separating business rules from business process and it also illustrated how to implement the architecture by integrating JRules rule engine with the Oracle BPEL Process Manager.

## Relationship between BRMS/RIF and MDA/MDI

MDA provides a methodology to develop software system by model transformation. In fact, developing a business rule application is similar to MDA. See Figure A-12, the top level is about the business policies which are written in natural language. The second level is about business rule artifacts based on vocabulary and BOM. Business rule artifacts are modelled in BAL and they are created from business policies by policy mangers. The vocabulary and BOM are also created from business policies by business analysts. All the outputs of this layer are not executable. At the third level, technical rule artifacts written in IRL are generated from transformation of the business rule artefacts. Meanwhile XOM will be generated from BOM according to the predefined mappings between BOM and XOM. Of course, in this level, some of the technical rule artifacts will be created by developers, such as functions or ruleflows. In this level, all the outputs are executable. The bottom level is the running environment for all technical rule artefacts and XOMs. According to the above narration, the level structure for JRules/BRMS implicitly practices the methodology of MDA. Besides, RIF can also be regarded as an application of MDI at the PSM level. From the side of MDA/MDI, model transformation needs transformation rules which can be created, executed and managed by BRMS. So BRMS/RIF and MDA/MDI are associated closely with each other.

[12] (IBM 2009) proposed a table to compare the features and constraints of JRules web services and hosted/ monitored transparent decision services.

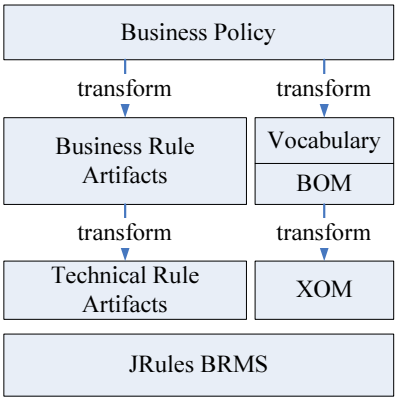



**Figure A-12. Level structure for JRules**

# Appendix B: Research domains in ontology

## Role/Architecture of ontologies

In (Wache, Vögele et al. 2001), the researched approaches of information integration use ontologies not only for content explication, but also either as a global query model or for the verification of integration description. When used for content explication, ontologies can be employed in three possible ways (see Figure B-1 and Table B-1).

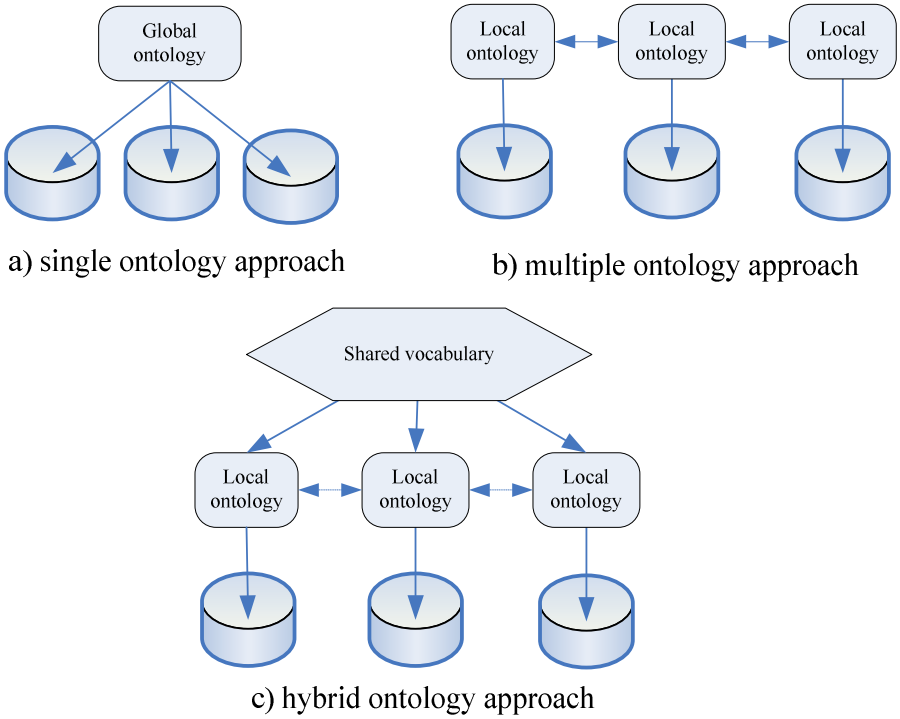



**Figure B-1. Three Ways for Employing Ontologies (Wache, Vögele et al. 2001)**

**Table B-1. Three Ways for Employing Ontologies (adapted from (Wache, Vögele et al. 2001))**

| Approach | Feature | Advantage | Disadvantage |
|---|---|---|---|
| single ontology approach | Share the same view on a domain. | It is straight-forward to implement. | Subject to changes in the information sources. |
| multiple ontology approach | No common ontology with the agreement of all sources is needed. | This ontology architecture can simplify the change, i.e. modifications in one information source. | In reality, the lack of a common vocabulary makes it extremely difficult to compare different source ontologies. |
| hybrid approach | Built upon one global shared vocabulary which contains the basic terms (the primitives) and some operators of a domain. The operators combine the primitives to construct complex terms. | New sources can easily be added without the need of modification in the mappings or in the shared vocabulary. | Existing ontologies cannot be reused easily, but have to be re-developed from scratch because all source ontologies have to refer to the shared vocabulary. |

## Ontology representation

There are lots of ontology languages, such as Ontolingua, OKBC, LOOM[13], OCML, FLogic, XOL, RDF(S), SHOE, OIL, DAML-ONTO, DAML+OIL, OWL and so on. Some of them are traditional ontology languages and some are web-based ontology languages. All of the web-based ontology languages are XML-based[14] (see Figure B-2). Why do web-based ontology languages are XML-based? Because there are some advantages (Corcho and Gómez-Pérez 2000):

- They have the definition of a common syntactic specification;
- They can be easily read for human beings (compared with traditional ontology languages);
- They represent distributed knowledge.

XML-based ontology languages also have some disadvantages:

- lack of structure for information;
- no standard tools for making inference for such language (Corcho and Gómez-Pérez 2000).

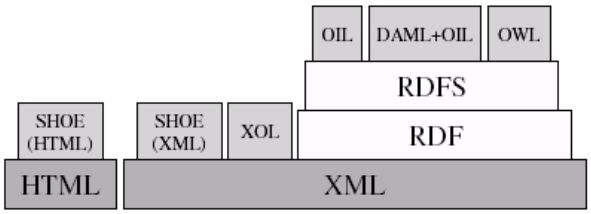



**Figure B-2. The Stack of Ontology Markup Languages (Corcho, Fernández-López et al. 2003)**

**Table B-2. Theories for Ontology languages (Corcho and Gómez-Pérez 2000; Horrocks, Patel-Schneider et al. 2003) (Gruber 1993; Chaudhri, Farquhar et al. 1997; Chaudhri, Farquhar et al. 1998) (MacGregor 1991; Kifer, Lausen et al. 1995; Lassila and Swick 1999; Motta 1999; Luke and Heflin 2000) (Brickley and Guha 1999) (Smith, Welty et al. ; Dean and Schreiber 2004; Patel-Schneider, Hayes et al. 2004)**

| KR Paradigms | Traditional Ontology languages | | | | | Web-based Ontology Languages | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Ontolingua | OKBC | OCML | LOOM | FLogic | XOL | RDF(S) | SHOE | OIL | OWL |
| Description Logic | | | | X | | | | | X | X |
| Object Oriented | | X | | | X | | | | | |
| Frame based | X | X | | X | X | X | | X | X | X |
| first/second order (predicate calculus) | X | | | | X | | | | | |
| Model theory | | | | | | | X | | X | X |

Each ontology language has its own background theories - "knowledge representation paradigms". Table B-2 provides the knowledge representation paradigms for the traditional and web-based ontology languages. There are five paradigms for the ontology languages: description logic, object-oriented, frame-based, first/second order predicate, model theory. In fact, description logic is a subset of first-order logic with well-known properties (Halpin

[13] http://www.isi.edu/isd/LOOM/
[14] Although SHOE was firstly an extension of HTML, now it is adapted in order to XML compliant.

2004). Unlike first-order logic, description logics have proved to be decidable and of a tractable complexity class (Borgida 1996).

In a project, which ontology language should be chosen? (Corcho and Gómez-Pérez 2000) has proposed an evaluation framework (see Figure B-3) for comparing the expressiveness and inference mechanisms of potential ontology languages. It has applied the evaluation framework to most of the ontology languages and found that:

- the traditional ontology languages are more expressive than web-based ontology languages;
- the inference capabilities of each language are very different. Corcho and his colleague have concluded that different needs in knowledge representation exist nowadays for applications, and some languages are more suitable than others.

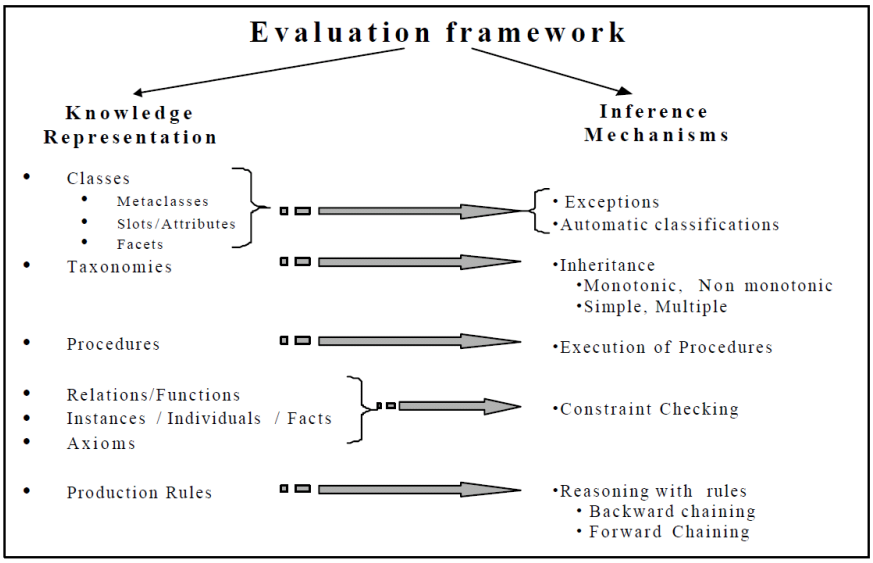



**Figure B-3. Evaluation Framework for Ontology Languages (Corcho and Gómez-Pérez 2000)**

## Ontology mapping

According to (Wache, Vögele et al. 2001), there are two kinds of ontology mappings: the mapping between ontology and information systems and the mapping between different ontologies (inter-ontology mapping). For the former mapping, (Wache, Vögele et al. 2001) lists four general approaches to establish the connection between ontology and information systems: structure resemblance, definition of terms, structure enrichment and meta-annotation. For the inter-ontology mapping, it is caused by the differences between ontologies, and such mapping is researched in the context of **semantic integration**[15] **/interoperability**. Such differences come from two levels (Noy 2004): **Language-level** and **ontology-level**.

- *Language-level*: the differences focus on the mismatching in expressiveness and semantics of ontology languages, e.g., different syntax and constructs.
- *Ontology-level*: even for the ontologies expressed in the same language, there are also some ontology-level mismatches including the same linguistic terms to describe different concepts; using different terms to describe the same concepts; using different modeling paradigms; using different modeling conventions and

[15] Semantic integration is an active area of research in several disciplines, such as databases, information-integration, and ontologies (Noy 2004). Most researchers agree that semantic integration is one of the most serious challenges for the Semantic Web today (Noy 2004).

levels of granularity; having ontologies with different coverage of the domain, and so on.

In order to research the problem of ontology mapping, (Noy 2004) starts from three dimensions: mapping discovery, declarative formal representations of mapping and reasoning with mappings.

- *Mapping discovery*: (Noy 2004) identifies two major approaches for mapping discovery between ontologies: using a **shared ontology** and using **heuristics-based or machine learning technique** [16] . The second approach uses various characteristics of ontology, such as concept names, natural-language description of concepts, class hierarchy, property definitions, instances of classes and class descriptions.
- *Mappings representation*: (Noy 2004) has discussed three methods:
  - representing mappings as **instances** in an ontology;
  - defining **bridging axioms** in first-order logic to represent transformations;
  - using **views** to describe mappings from a global ontology to local ontology.
- *Reason with mappings*: Naturally, defining the mappings between ontologies, either automatically, semi-automatically or interactively, is not a goal in itself. The resulting mappings are used for various integration tasks: **data transformation**, **query answering** or **web-service composition**.

## Ontology Engineering

After the discussion of role/architecture of ontology, ontology representation and ontology mapping, it is crucial to support ontology development - ontology engineering. In (Wache, Vögele et al. 2001), ontology engineering contains three aspects: development methodology, development tools and ontology evolution.

### *Ontology engineering methodology*

According to (Héon, Paquette et al. 2008), ontology development implies **three main activities** which are generally conducted by a knowledge engineer:

1) **knowledge elicitation**,
2) **formalization** of the elicited knowledge into an ontology,
3) **syntactic and semantic** validation of the ontology.

The whole process is complicated and requires knowledge engineers with high competencies. The goal of knowledge engineers is to reduce as much as possible the **gap between** the richness of the actual expertise of the **domain experts** and its formal ontological representation. (Héon, Paquette et al. 2008) has proposed its own approach to develop ontology. Its approach has three steps as follows:

1) The domain experts participate directly to the elicitation operation through semi-formal visual knowledge modelling;
2) The engineer transforms the semi-formal model to a formal one taking the form of ontology. An expert system is developed in (Héon, Paquette et al. 2008) to aid the engineers in this transformation process. A “transformation ontology” serves as a

[16] The task of finding mappings (semi-)automatically has been an active area of research in both database and ontology communities (Rahm E. et al, 2001) (Kalfoglou Y. et al, 2003).

knowledge base for the transformation service to be carried out by the expert system;

3) Finally, the knowledge engineer validates the result with the domain experts.

Besides the above work about ontology engineering, (Jarrar and Meersman 2008) has proposed its own ontology engineering methodological framework and it points out the main foundational challenge in ontology engineering: ***trade-off between ontology usability and ontology reusability.*** The trade-off is caused by the fact that there doesn't exist a strict line between **specific** and **generic** knowledge (Chandrasekaran and Johnson 1993)**.** In detail, from a *methodological viewpoint,* if a methodology emphasizes usability perspectives or evaluates ontologies based on how they fulfil specific application requirements, the resulting ontology will be similar to a conceptual data schema (or a classical knowledge base) containing application specific and thus, less reusable knowledge (Jarrar and Meersman 2008). Likewise, if a methodology emphasizes the independency of the knowledge, the resulting ontology in general will be less *usable,* since it has no intended use by ignoring application perspectives (Jarrar and Meersman 2008). To tackle such a foundational challenge, (Jarrar and Meersman 2008) proposes a methodological framework - DOGMA. The idea of DOGMA is that: ontology is doubly articulated into a *domain axiomatization* and *application axiomatization*. A domain axiomatization is primarily related with characterizing the "intended meanings" of domain vocabulary (typically shared and public); an application axiomatization (typically local) is primarily associated with the usability of these vocabularies. The **double articulation** implies that all concepts and relationships introduced in an application axiomatization are predefined in its domain axiomatization (Jarrar and Meersman 2008). Multiple application axiomatizations share and reuse the same intended meanings in a domain axiomatization (Jarrar and Meersman 2008). This approach increases reusability of domain axiomatization, as well as usability of application axiomatizations.

A general overview on ontology engineering methodologies is provided by (Gómez-Pérez, Fernández-López et al. 2004), including short descriptions of the methods.

### *Ontology Development Tools*

(Wache, Vögele et al. 2001) has sketched three available tools at that time: OntoEdit, SHOE's knowledge annotator and DWQ. The web site[17] of Michael K. Bergman lists 185 ontology development tools, 35 of which are recently new and 45 added at various times since the first release. For these 185 tools, there is diversity both in terms of scope and function across the entire ontology development stack. The web site of Michael K. Bergman also shows that nearly all of those 185 tools do not communicate with one another[18]. However, recently, simpler, task-focused tools with intuitive interfaces[18] are more demanded in the market. Therefore, the general tools architecture needs to be shifted from IDEs and comprehensive toolkits to APIs and Web services, such as OWL API (Horridge and Bechhofer 2009; Horridge and Bechhofer 2010).

### *Ontology evolution*

Ontology evolution is the problem of modifying ontology in response to a certain change in the domain or its conceptualization (Flouris, Plexousakis et al. 2006). There are several cases where ontology evolution is applicable:

- An ontology, just like any structure holding information, may need to change simply because the world has changed (Stojanovic, Maedche et al. 2003);

[17] http://www.mkbergman.com/904/listing-of-185-ontology-building-tools/
[18] http://www.mkbergman.com/909/a-new-landscape-in-ontology-development-tools/

- We may need to change the perspective under which the domain is viewed (Noy and Klein 2004);
- We may discover a problem in the original conceptualization of the domain (Flouris, Plexousakis et al. 2006);
- We might also wish to incorporate additional functionality, according to a change in users' needs (Haase and Stojanovic 2005);
- Furthermore, new information, which was previously unknown, classified or otherwise unavailable may become accessible, or different features of the domain may become important (Heflin, Hendler et al. 1999).

(Flouris, Plexousakis et al. 2006) argues that the currently used ontology evolution model has several weaknesses, and it presents an abstract proposition for a future research direction that will hopefully resolve these weaknesses, based on the related field of belief change (Gärdenfors 1992).

# Appendix C: Graphical User Interfaces for Six CBP Tools

The following sections list the graphical user interfaces for six collaborative business process tools: BizAgi, jBPM, Bonita Open Solution, Oracle BPM Suit 11g, ADONIS and MEGA. These figures illustrate the primary components for each CBP tools, such as business process modeler, simulation, system console, business process monitoring, dependent database, etc. For example, in Section "jBPM V5.1.0", Figure C-5 depicts BPMN Visual Editor for jBPM, which can construct conversation diagrams, choreography diagrams and collaboration diagrams, etc. Figure C-6 depicts the knowledge base "Guvnor" and the web-based BPMN editor for jBPM. All business processes in jBPM are stored and deployed in Guvnor. If jBPM creates an instance for a process, it will retrieve the corresponding process from Guvnor. Figure C-7 depicts the console of jBPM, which can configure, control and monitor the jBPM process engine, execute personal tasks and generate statistic report for process execution. Figure C-8 shows the execution progress of a business process in jBPM, and Figure C-9 illustrates a statistical report of process execution.

## BizAgi

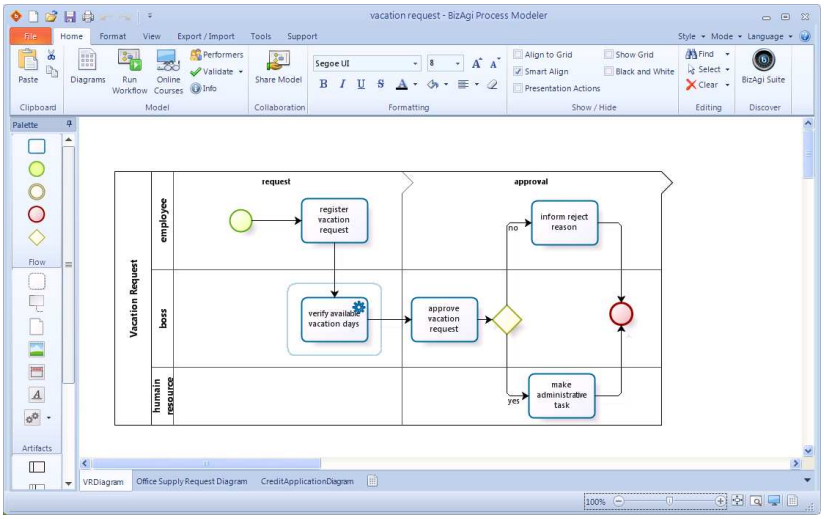



**Figure C-1. BizAgi Process Modeler V2.0.0.2**

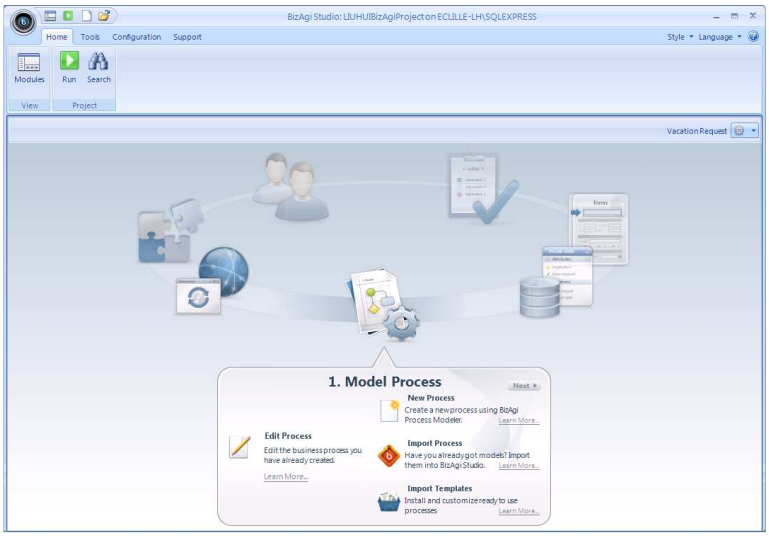



**Figure C-2. BizAgi Studio V9.1.6.1005**

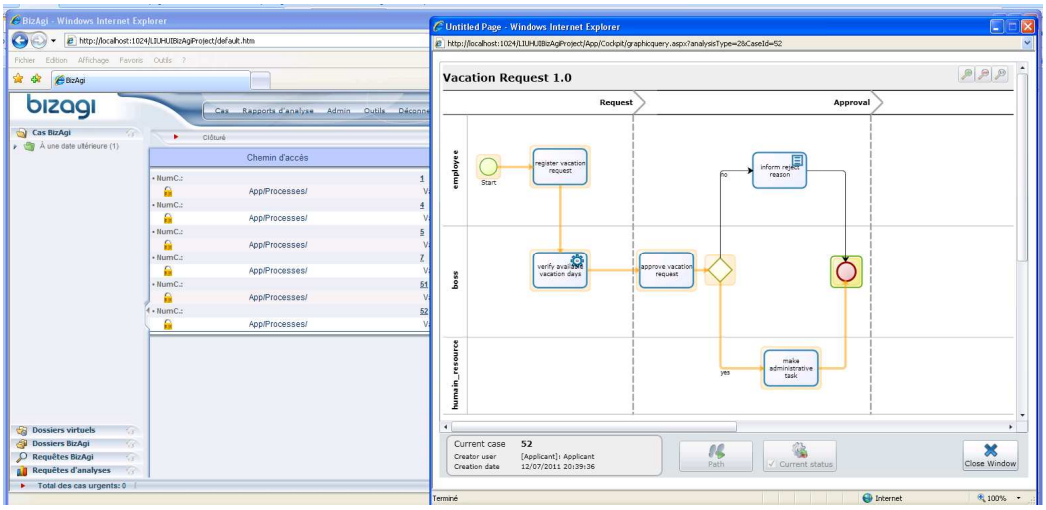



**Figure C-3. BizAgi Console (Work portal)**

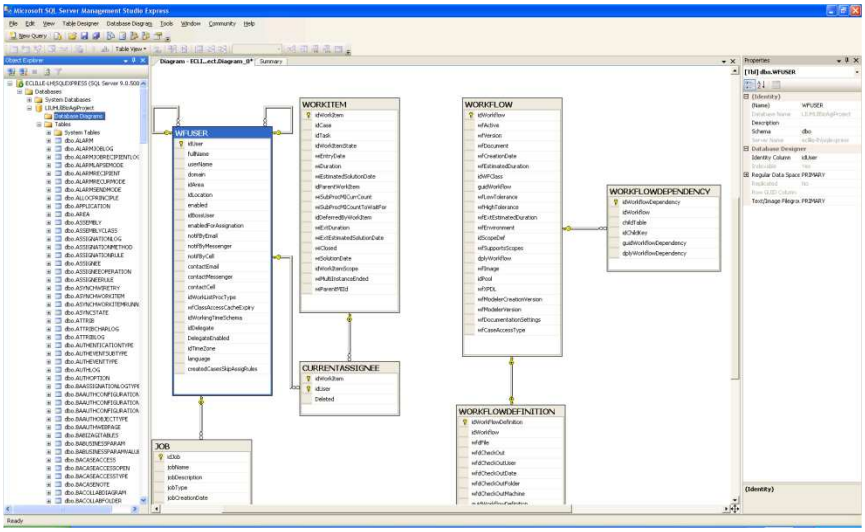



**Figure C-4. Database of BizAgi**

# jBPM V5.1.0

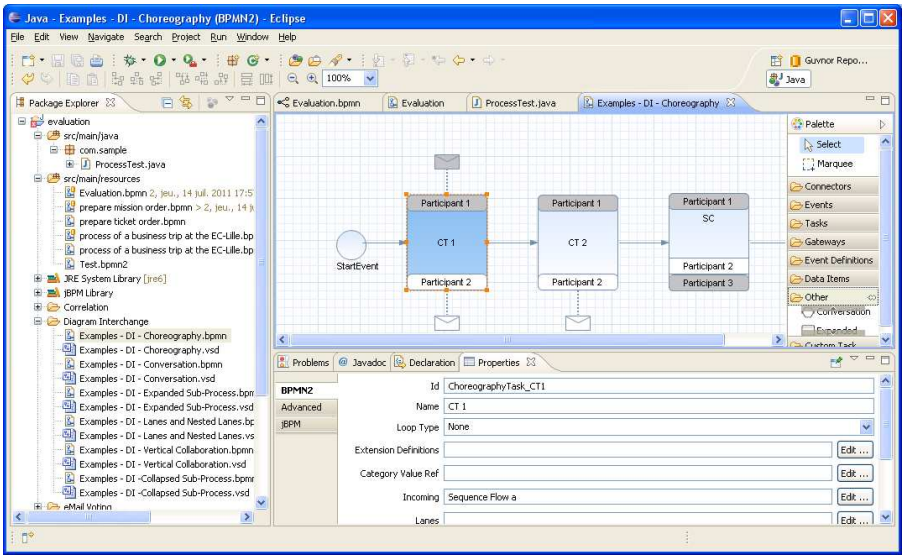



**Figure C-5. BPMN Visual Editor (Eclipse plugin) for jBPM**

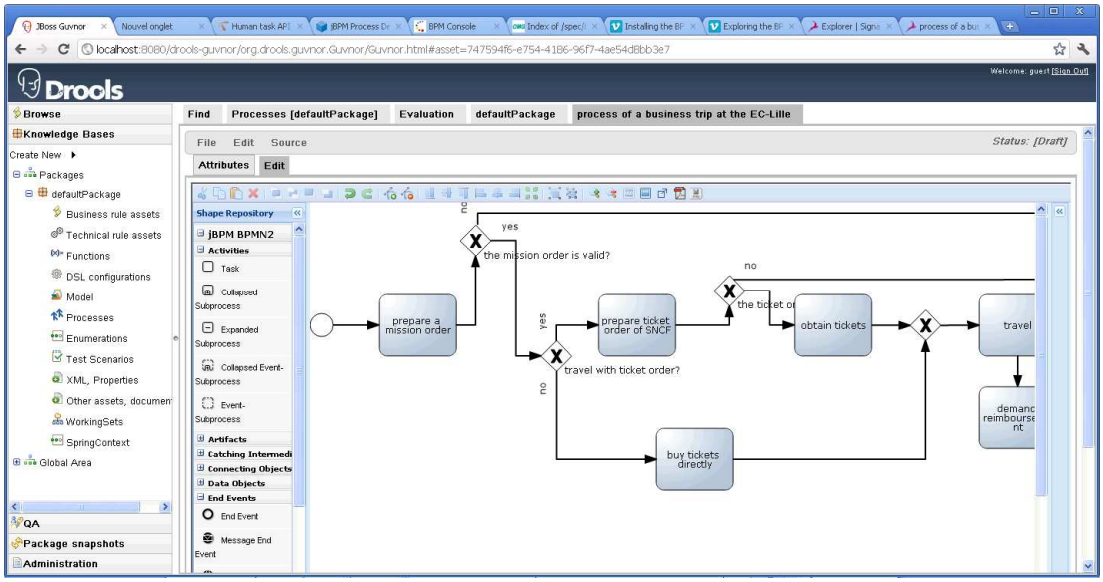



**Figure C-6. Knowledge base (Guvnor) and web-based BPMN editor for jBPM**

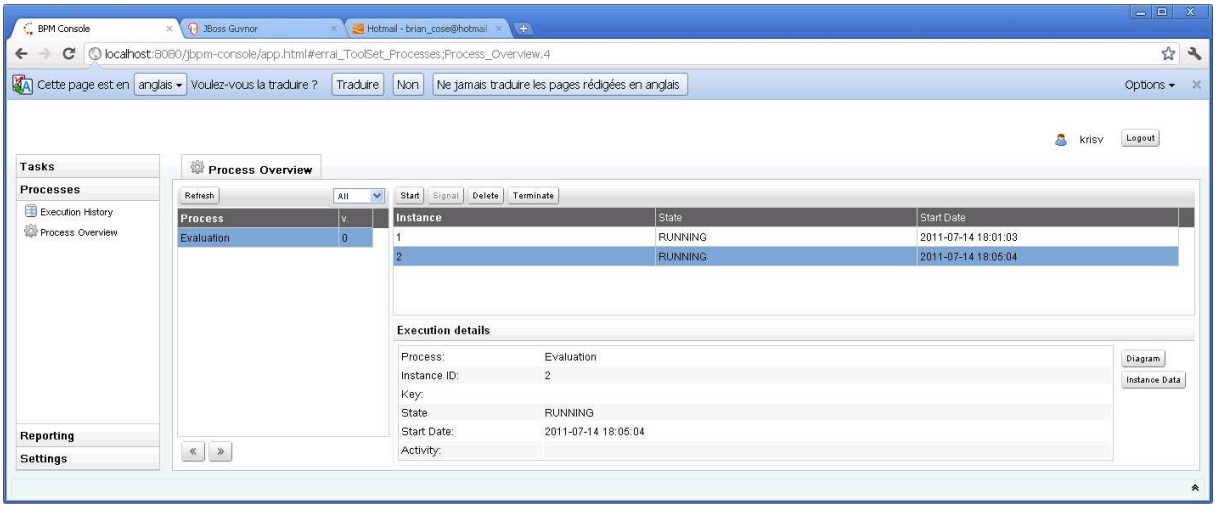



**Figure C-7. jBPM Console**

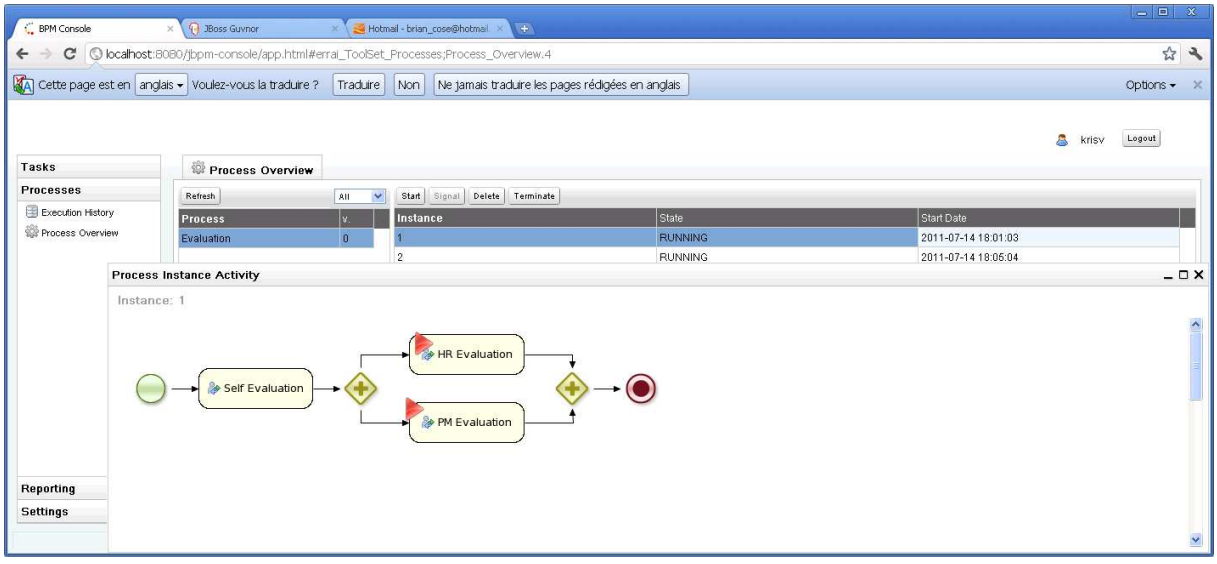



**Figure C-8. Execution Progress of a Business Process in jBPM**

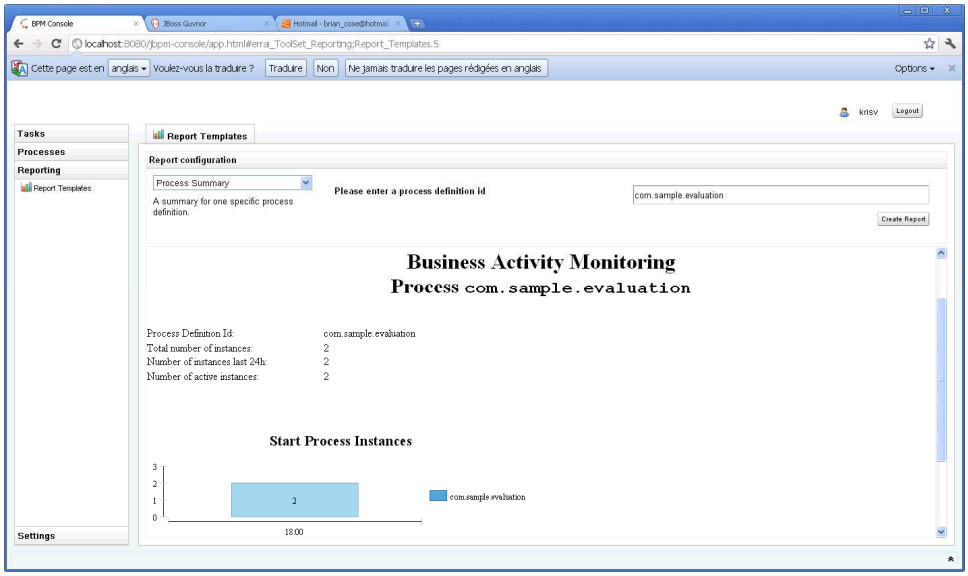



**Figure C-9. Execution Report from jBPM BAM**

# Bonita Open Solution V5.5.1

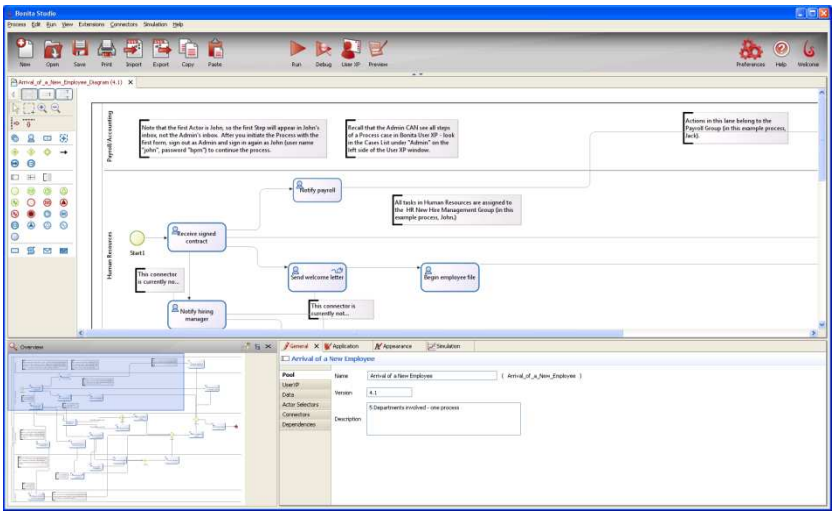


**Figure C-10. Bonita Studio**

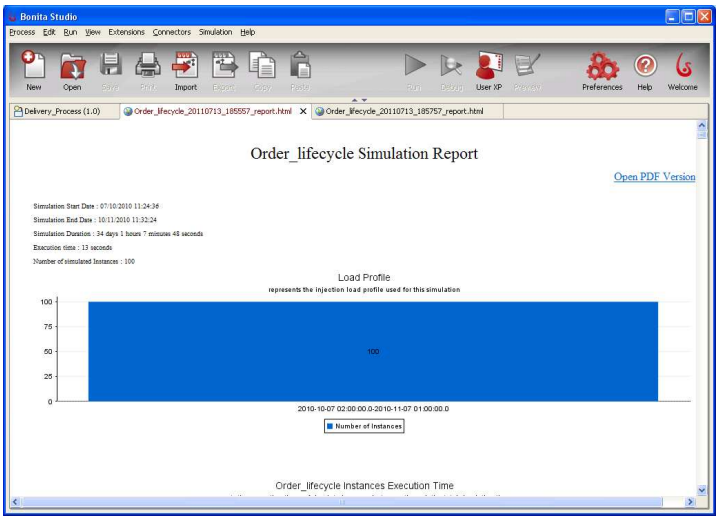



**Figure C-11. Bonita Studio Simulation Report**

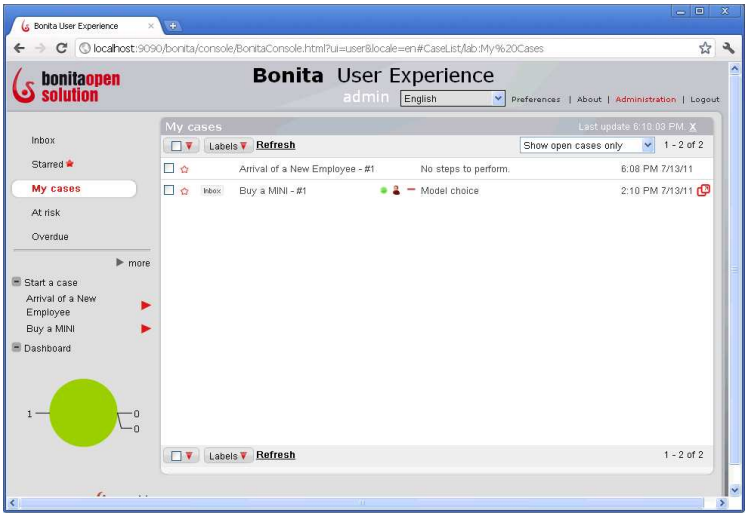



**Figure C-12. Bonita User Experience**

# Oracle BPM Suite 11g

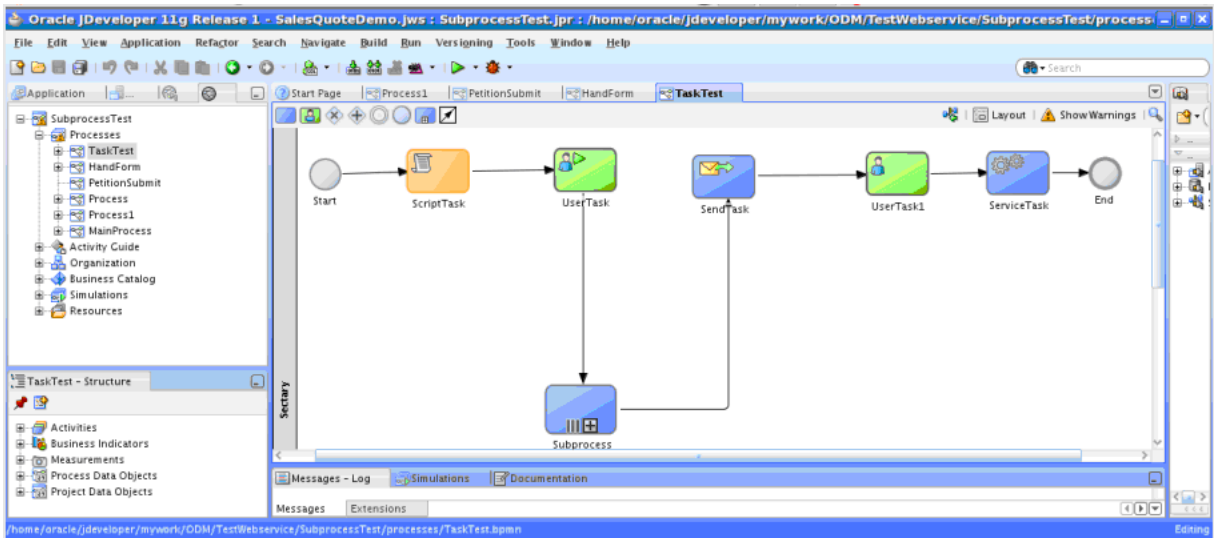



**Figure C-13. Oracle JDeveloper 11g for BPMN2.0**

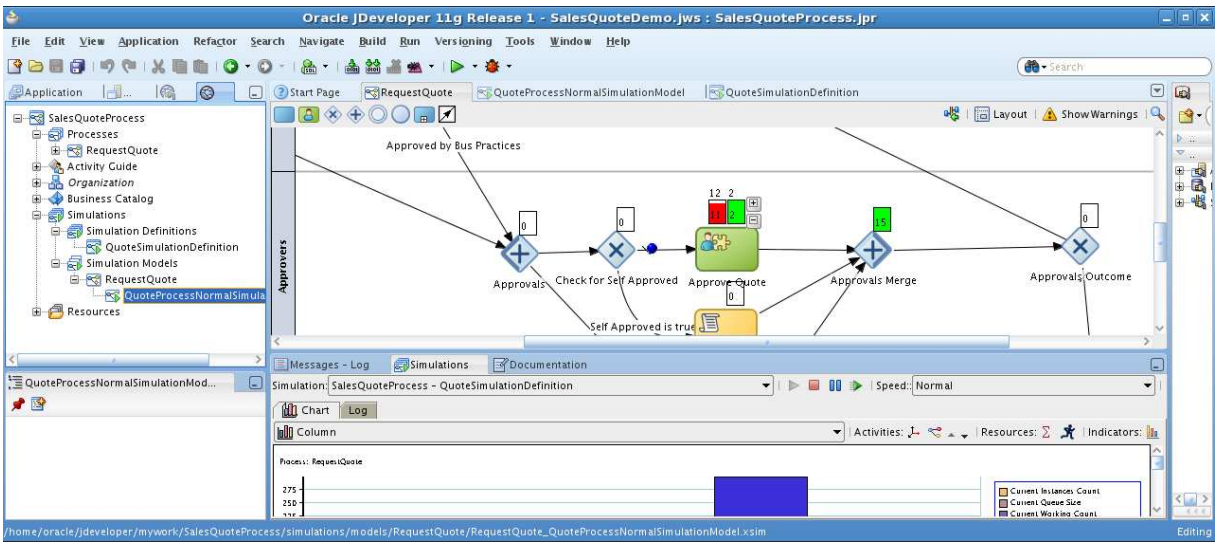



**Figure C-14. Simulation in Oracle JDeveloper 11g for BPMN2.0**

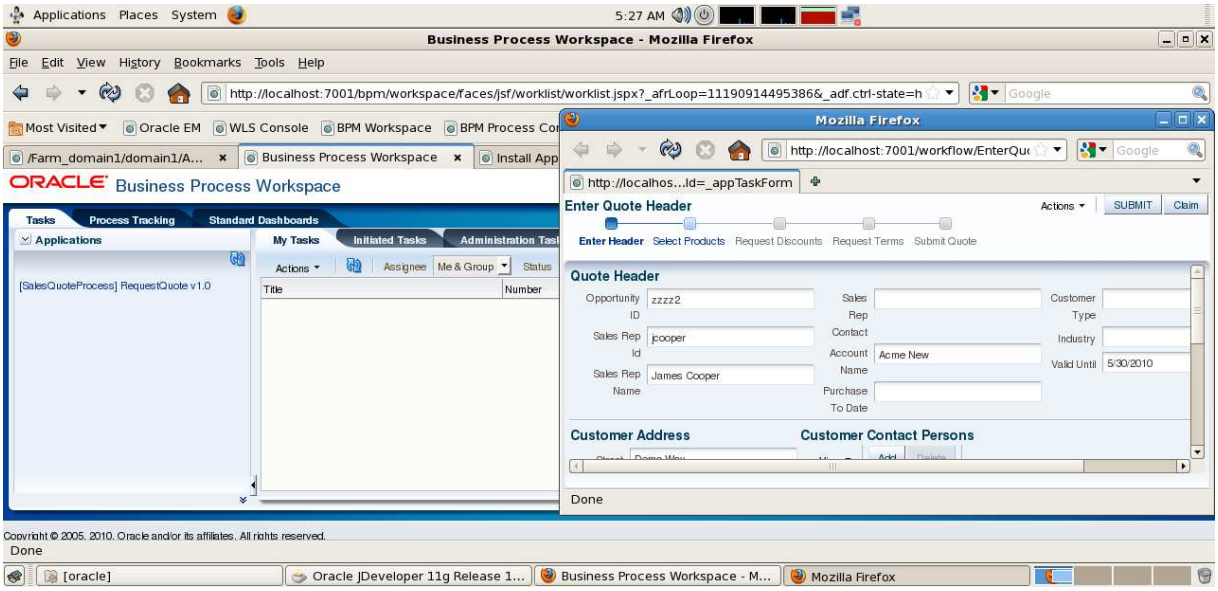



**Figure C-15. Web console for Oracle BPM Suite 11g**

# ADONIS

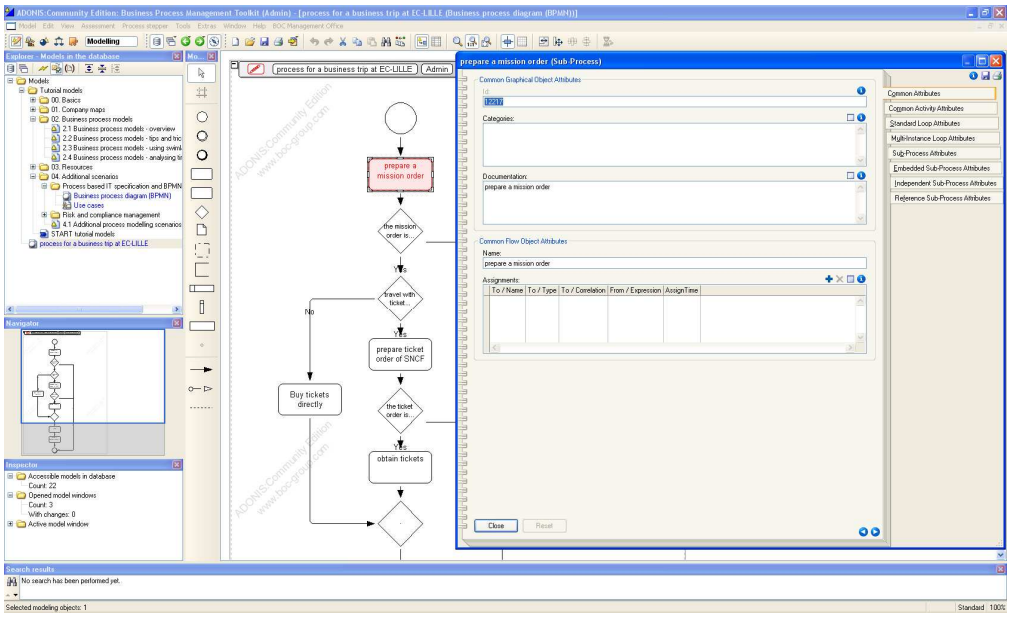


**Figure C-16. Process Modeling in ADONIS V3.90.01.98**

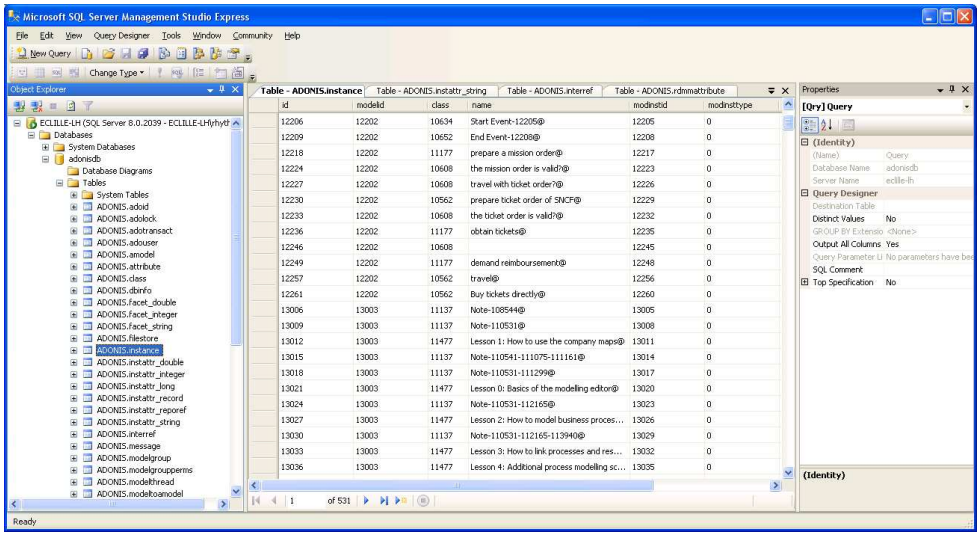



**Figure C-17. Database for ADONIS**

# MEGA

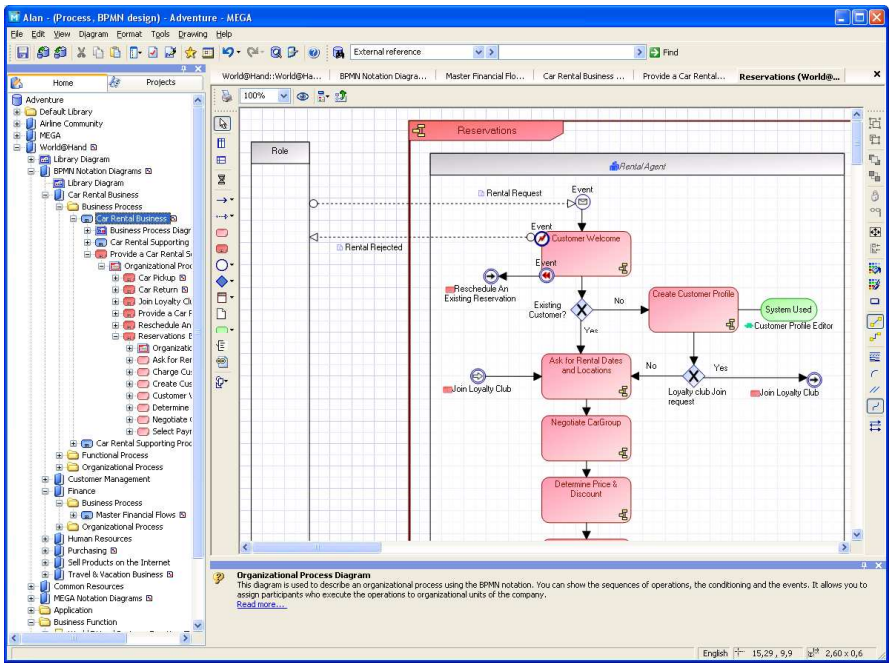



**Figure C-18. MEGA**

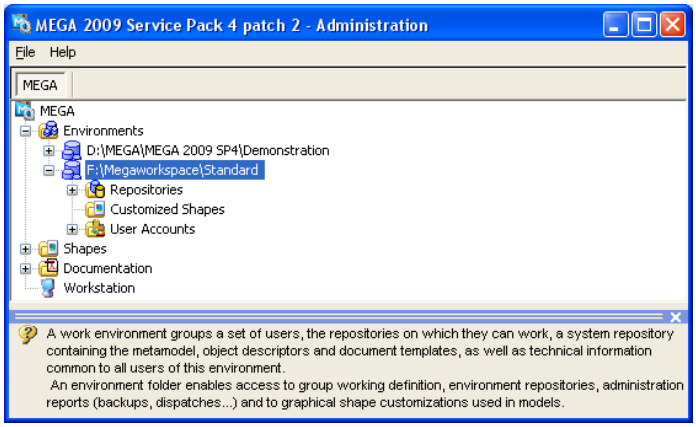



**Figure C-19. Databases for MEGA**

# Appendix D: Schema definition for semantic annotations of BPMN2.0

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:bpmn20="http://www.omg.org/spec/BPMN/20100524/MODEL"
targetNamespace="http://www.omg.org/spec/BPMN/20100524/MODEL"
elementFormDefault="qualified" attributeFormDefault="unqualified">
	<xs:include schemaLocation="Semantic.xsd"/>
	<xs:complexType name="tSemanticAnnotation">
		<xs:complexContent>
			<xs:extension base="bpmn20:tRootElement">
				<xs:sequence>
					<xs:element name="detail" type="tSemanticDetail"
minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
				<xs:attribute name="bpmnElement" type="xs:QName"/>
				<xs:attribute name="ontologyRef" type="xs:anyURI"
use="optional"/>
				<xs:attribute name="level" type="tMDALevel" use="optional"/>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:complexType name="tSemanticDetail">
		<xs:sequence>
			<xs:element name="actor" type="tSemanticRef" minOccurs="0"
maxOccurs="unbounded"/>
			<xs:element name="effect" type="tSemanticRef" minOccurs="0"
maxOccurs="unbounded"/>
			<xs:element name="do" type="tSemanticRef" minOccurs="1"
maxOccurs="unbounded"/>
			<xs:element name="what" type="tSemanticRef" minOccurs="0"
maxOccurs="unbounded"/>
			<xs:element name="where" type="tSemanticRef" minOccurs="0"
maxOccurs="unbounded"/>
			<xs:element name="when" type="tSemanticRef" minOccurs="0"
maxOccurs="unbounded"/>
			<xs:element name="how" type="tSemanticRef" minOccurs="0"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:simpleType name="tMDALevel">
		<xs:restriction base="xs:string">
			<xs:enumeration value="CIM"/>
			<xs:enumeration value="PIM"/>
			<xs:enumeration value="PSM"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="tSemanticRef" mixed="true">
		<xs:sequence>
			<xs:any namespace="##any" processContents="lax" minOccurs="0"/>
		</xs:sequence>
```

```
            <xs:attribute name="ontologyRef" type="xs:anyURI" use="optional"/>
        </xs:complexType>
        <xs:complexType name="tSemanticAnnotationList">
            <xs:complexContent>
                <xs:extension base="bpmn20:tRootElement">
                    <xs:sequence>
                        <xs:element name="semanticAnnotation"
type="tSemanticAnnotation" minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="semanticAnnotationList" type="tSemanticAnnotationList"
substitutionGroup="bpmn20:rootElement"/>
        <xs:attribute name="ontologyRef" type="xs:anyURI"/>

</xs:schema>
```

# Appendix E: Goal Ontology

This appendix explains the goal ontology defined in Chapter 5. In Figure E-1, the goal ontology is created in OWL by Protégé 4.2.1. The corresponding OWL document for the goal ontology is shown behind Figure E-1.

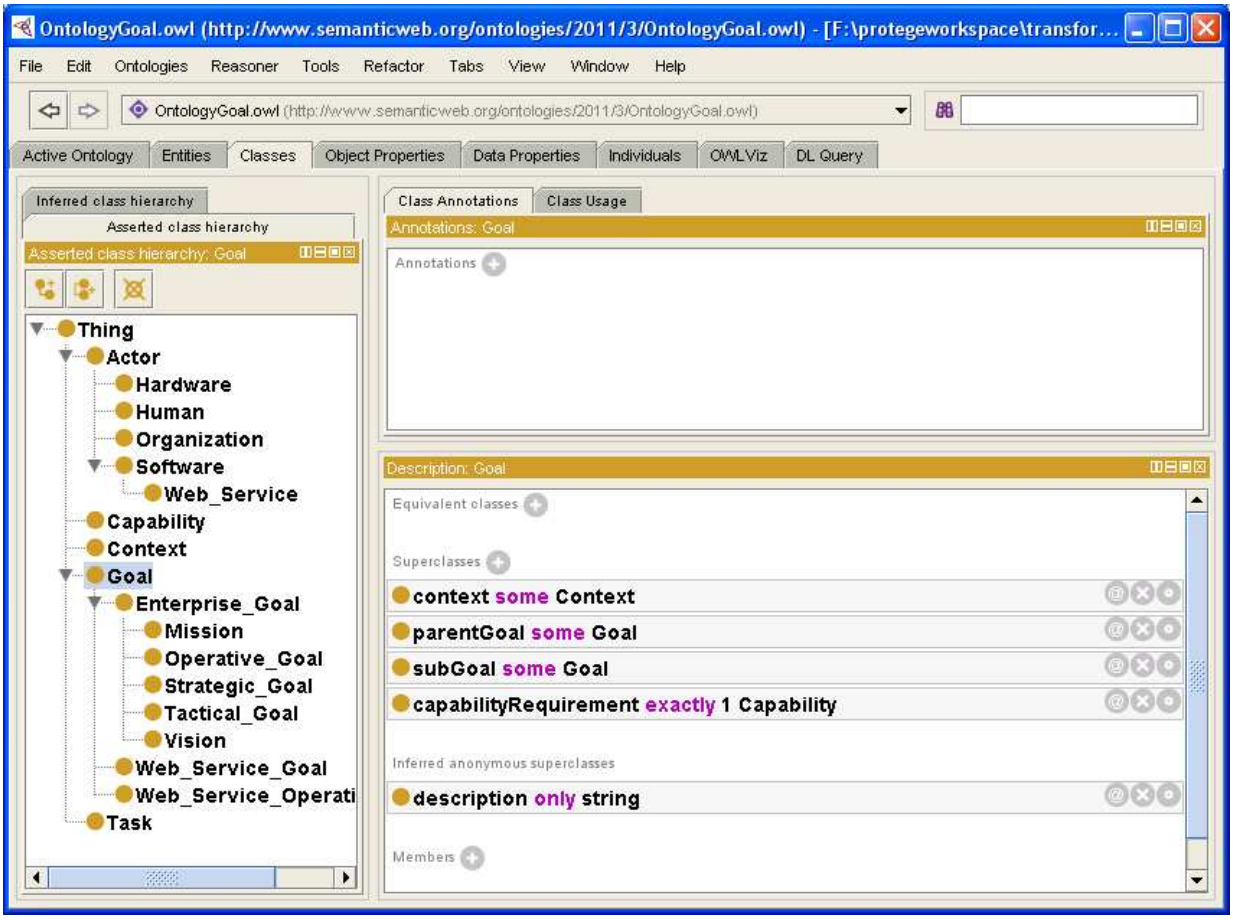



**Figure E-1. Goal Ontology in Hierarchical style Developed by Protégé 4.2.1**

OWL document for Goal Ontology:

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


<!DOCTYPE owl2xml:Ontology [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY OntologyGoal "http://www.semanticweb.org/ontologies/2011/3/OntologyGoal.owl#" >
]>


<owl2xml:Ontology xmlns="http://www.semanticweb.org/ontologies/2011/3/OntologyGoal.owl#"
    xml:base="http://www.w3.org/2006/12/owl2-xml#"
    xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
    xmlns:OntologyGoal="http://www.semanticweb.org/ontologies/2011/3/OntologyGoal.owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    owl2xml:URI="http://www.semanticweb.org/ontologies/2011/3/OntologyGoal.owl">
   <owl2xml:SubClassOf>
      <owl2xml:Class owl2xml:URI="&OntologyGoal;Actor"/>
      <owl2xml:ObjectSomeValuesFrom>
         <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;parentActor"/>
         <owl2xml:Class owl2xml:URI="&OntologyGoal;Actor"/>
      </owl2xml:ObjectSomeValuesFrom>
   </owl2xml:SubClassOf>
```

```
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Actor"/>
   <owl2xml:ObjectSomeValuesFrom>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;subActor"/>
      <owl2xml:Class owl2xml:URI="&OntologyGoal;Actor"/>
   </owl2xml:ObjectSomeValuesFrom>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Actor"/>
   <owl2xml:ObjectMinCardinality owl2xml:cardinality="1">
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;realizedCapability"/>
      <owl2xml:Class owl2xml:URI="&OntologyGoal;Capability"/>
   </owl2xml:ObjectMinCardinality>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Capability"/>
   <owl2xml:ObjectSomeValuesFrom>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;object"/>
      <owl2xml:Class owl2xml:URI="&owl;Thing"/>
   </owl2xml:ObjectSomeValuesFrom>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Capability"/>
   <owl2xml:ObjectExactCardinality owl2xml:cardinality="1">
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;verb"/>
   </owl2xml:ObjectExactCardinality>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Context"/>
   <owl2xml:ObjectSomeValuesFrom>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;effect"/>
      <owl2xml:Class owl2xml:URI="&owl;Thing"/>
   </owl2xml:ObjectSomeValuesFrom>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Context"/>
   <owl2xml:ObjectSomeValuesFrom>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;input"/>
      <owl2xml:Class owl2xml:URI="&owl;Thing"/>
   </owl2xml:ObjectSomeValuesFrom>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Context"/>
   <owl2xml:ObjectSomeValuesFrom>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;output"/>
      <owl2xml:Class owl2xml:URI="&owl;Thing"/>
   </owl2xml:ObjectSomeValuesFrom>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Context"/>
   <owl2xml:ObjectSomeValuesFrom>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;precondition"/>
      <owl2xml:Class owl2xml:URI="&owl;Thing"/>
   </owl2xml:ObjectSomeValuesFrom>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Enterprise_Goal"/>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Goal"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Goal"/>
   <owl2xml:ObjectSomeValuesFrom>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;context"/>
      <owl2xml:Class owl2xml:URI="&OntologyGoal;Context"/>
   </owl2xml:ObjectSomeValuesFrom>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Goal"/>
   <owl2xml:ObjectSomeValuesFrom>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;parentGoal"/>
      <owl2xml:Class owl2xml:URI="&OntologyGoal;Goal"/>
   </owl2xml:ObjectSomeValuesFrom>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Goal"/>
   <owl2xml:ObjectSomeValuesFrom>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;subGoal"/>
      <owl2xml:Class owl2xml:URI="&OntologyGoal;Goal"/>
   </owl2xml:ObjectSomeValuesFrom>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Goal"/>
   <owl2xml:ObjectExactCardinality owl2xml:cardinality="1">
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;capabilityRequirement"/>
      <owl2xml:Class owl2xml:URI="&OntologyGoal;Capability"/>
   </owl2xml:ObjectExactCardinality>
```

```
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Hardware"/>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Actor"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Human"/>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Actor"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Mission"/>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Enterprise_Goal"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Operative_Goal"/>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Enterprise_Goal"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Organization"/>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Actor"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Software"/>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Actor"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Strategic_Goal"/>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Enterprise_Goal"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Tactical_Goal"/>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Enterprise_Goal"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Task"/>
   <owl2xml:ObjectSomeValuesFrom>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;parentTask"/>
      <owl2xml:Class owl2xml:URI="&OntologyGoal;Task"/>
   </owl2xml:ObjectSomeValuesFrom>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Task"/>
   <owl2xml:ObjectSomeValuesFrom>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;subTask"/>
      <owl2xml:Class owl2xml:URI="&OntologyGoal;Task"/>
   </owl2xml:ObjectSomeValuesFrom>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Task"/>
   <owl2xml:ObjectExactCardinality owl2xml:cardinality="1">
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;context"/>
      <owl2xml:Class owl2xml:URI="&OntologyGoal;Context"/>
   </owl2xml:ObjectExactCardinality>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Task"/>
   <owl2xml:ObjectExactCardinality owl2xml:cardinality="1">
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;satisfiedCapability"/>
      <owl2xml:Class owl2xml:URI="&OntologyGoal;Capability"/>
   </owl2xml:ObjectExactCardinality>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Vision"/>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Enterprise_Goal"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Web_Service"/>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Software"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Web_Service_Goal"/>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Goal"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Web_Service_Operation_Goal"/>
   <owl2xml:Class owl2xml:URI="&OntologyGoal;Goal"/>
</owl2xml:SubClassOf>
<owl2xml:SubClassOf>
   <owl2xml:Class owl2xml:URI="&owl;Thing"/>
   <owl2xml:DataAllValuesFrom>
      <owl2xml:DataProperty owl2xml:URI="&OntologyGoal;description"/>
      <owl2xml:Datatype owl2xml:URI="&xsd;string"/>
   </owl2xml:DataAllValuesFrom>
</owl2xml:SubClassOf>
<owl2xml:ObjectPropertyRange>
   <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;Dependent_Resource"/>
```

```
    <owl2xml:ObjectSomeValuesFrom>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;Dependent_Resource"/>
      <owl2xml:Class owl2xml:URI="&owl;Thing"/>
    </owl2xml:ObjectSomeValuesFrom>
  </owl2xml:ObjectPropertyRange>
  <owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;context"/>
    <owl2xml:ObjectAllValuesFrom>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;context"/>
      <owl2xml:Class owl2xml:URI="&OntologyGoal;Context"/>
    </owl2xml:ObjectAllValuesFrom>
  </owl2xml:ObjectPropertyRange>
  <owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;effect"/>
    <owl2xml:Class owl2xml:URI="&OntologyGoal;Context"/>
  </owl2xml:ObjectPropertyDomain>
  <owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;effect"/>
    <owl2xml:Class owl2xml:URI="&owl;Thing"/>
  </owl2xml:ObjectPropertyRange>
  <owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;input"/>
    <owl2xml:Class owl2xml:URI="&OntologyGoal;Context"/>
  </owl2xml:ObjectPropertyDomain>
  <owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;input"/>
    <owl2xml:Class owl2xml:URI="&owl;Thing"/>
  </owl2xml:ObjectPropertyRange>
  <owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;object"/>
    <owl2xml:ObjectSomeValuesFrom>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;object"/>
      <owl2xml:Class owl2xml:URI="&owl;Thing"/>
    </owl2xml:ObjectSomeValuesFrom>
  </owl2xml:ObjectPropertyRange>
  <owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;output"/>
    <owl2xml:Class owl2xml:URI="&OntologyGoal;Context"/>
  </owl2xml:ObjectPropertyDomain>
  <owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;output"/>
    <owl2xml:Class owl2xml:URI="&owl;Thing"/>
  </owl2xml:ObjectPropertyRange>
  <owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;parentActor"/>
    <owl2xml:Class owl2xml:URI="&OntologyGoal;Actor"/>
  </owl2xml:ObjectPropertyDomain>
  <owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;parentActor"/>
    <owl2xml:Class owl2xml:URI="&OntologyGoal;Actor"/>
  </owl2xml:ObjectPropertyRange>
  <owl2xml:Declaration>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;parentActor"/>
  </owl2xml:Declaration>
  <owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;parentTask"/>
    <owl2xml:Class owl2xml:URI="&OntologyGoal;Task"/>
  </owl2xml:ObjectPropertyDomain>
  <owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;parentTask"/>
    <owl2xml:Class owl2xml:URI="&OntologyGoal;Task"/>
  </owl2xml:ObjectPropertyRange>
  <owl2xml:Declaration>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;parentTask"/>
  </owl2xml:Declaration>
  <owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;precondition"/>
    <owl2xml:Class owl2xml:URI="&OntologyGoal;Context"/>
  </owl2xml:ObjectPropertyDomain>
  <owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;precondition"/>
    <owl2xml:Class owl2xml:URI="&owl;Thing"/>
  </owl2xml:ObjectPropertyRange>
  <owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;subActor"/>
    <owl2xml:Class owl2xml:URI="&OntologyGoal;Actor"/>
  </owl2xml:ObjectPropertyDomain>
  <owl2xml:ObjectPropertyRange>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;subActor"/>
    <owl2xml:Class owl2xml:URI="&OntologyGoal;Actor"/>
  </owl2xml:ObjectPropertyRange>
  <owl2xml:Declaration>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;subActor"/>
  </owl2xml:Declaration>
  <owl2xml:ObjectPropertyDomain>
    <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;subTask"/>
```

```
      <owl2xml:Class owl2xml:URI="&OntologyGoal;Task"/>
   </owl2xml:ObjectPropertyDomain>
   <owl2xml:ObjectPropertyRange>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;subTask"/>
      <owl2xml:Class owl2xml:URI="&OntologyGoal;Task"/>
   </owl2xml:ObjectPropertyRange>
   <owl2xml:Declaration>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;subTask"/>
   </owl2xml:Declaration>
   <owl2xml:ObjectPropertyRange>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;verb"/>
      <owl2xml:ObjectSomeValuesFrom>
         <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;verb"/>
         <owl2xml:Class owl2xml:URI="&owl;Thing"/>
      </owl2xml:ObjectSomeValuesFrom>
   </owl2xml:ObjectPropertyRange>
   <owl2xml:ObjectPropertyRange>
      <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;who"/>
      <owl2xml:ObjectSomeValuesFrom>
         <owl2xml:ObjectProperty owl2xml:URI="&OntologyGoal;who"/>
         <owl2xml:Class owl2xml:URI="&owl;Thing"/>
      </owl2xml:ObjectSomeValuesFrom>
   </owl2xml:ObjectPropertyRange>
   <owl2xml:DataPropertyDomain>
      <owl2xml:DataProperty owl2xml:URI="&OntologyGoal;description"/>
      <owl2xml:Class owl2xml:URI="&owl;Thing"/>
   </owl2xml:DataPropertyDomain>
   <owl2xml:DataPropertyRange>
      <owl2xml:DataProperty owl2xml:URI="&OntologyGoal;description"/>
      <owl2xml:Datatype owl2xml:URI="&xsd;string"/>
   </owl2xml:DataPropertyRange>
</owl2xml:Ontology>


<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->
```

**Résumé :** Quand des entreprises collaborent entre elles pour atteindre leurs objectifs métiers, des problèmes d'interopérabilité seront rencontrés. Afin de résoudre ces problèmes, nous étudions les domaines suivants : les processus métier collaboratifs, MDA, SOA, ESB et l'ontologie. Nous proposons alors un cadre intégrant ces cinq domaines pour les solutions TI (technologies de l''information) aux problèmes d'interopérabilité. Pour construire ce cadre, nous proposons une Méthode Basée sur des Processus pour l'Interopérabilité d'Entreprise (MBPIE), qui utilise des processus collaboratifs pour représenter des exigences de collaboration. MBPIE transforme des processus collaboratifs en plusieurs processus d'interopérabilité exécutables par des transformations de modèles. En MBPIE, l'ontologie est utilisée pour annoter les processus collaboratifs. Pendant la transformation des processus, de nouvelles informations ontologiques sont ajoutées dans les processus pour les rendre exécutables. Nous avons conçu un bus de services sémantiques Basé sur l'Ontologie et Dirigé par des Buts (BODB) pour supporter l'exécution des processus d'interopérabilité. Ce bus est basé sur un mécanisme symétrique pour l'invocation de services sémantiques. Ce mécanisme utilise l'extension de SOAP (Simple Object Access Protocol) qui est composée de trois parties : le format des messages BODB, le module BODB et le modèle de traitement BODB. Ce mécanisme a trois propriétés de transparence (emplacement, sémantique et technique) qui sont essentielles à l'exécution des processus d'interopérabilité. Ensemble, MBPIE et le bus constituent une approche fédérée pour résoudre les problèmes d'interopérabilité.

**Mots clés :** interopérabilité d'entreprise, collaboration, processus métier, processus collaboratif, processus d'interopérabilité, transformation, rang, taux de coopération, ontologie, annotation sémantique, bus de services sémantiques, MDA, MDI, SOA, BPMN2.0, approche dirigée par les buts, SOAP, cadre de comparaison

---

**Abstract:** When enterprises collaborate with others to achieve business objectives, enterprise interoperability problems will be encountered. In order to solve the problems, in this thesis, we analyze the five related research domains: collaborative business process, MDA, SOA, ESB and ontology. Consequently, we propose a framework for IT solutions to interoperability problems, which integrates the above five domains together. In order to realize the framework, we propose a Process-Based Method for Enterprise Interoperability (PBMEI), which employs collaborative processes to represent collaboration requirements between enterprises. PBMEI transforms collaborative processes into multiple executable interoperability processes through model transformations. In PBMEI, ontology is used to annotate collaborative processes. During model transformation, new ontology information will be added into processes. Such information will contribute to process execution. In order to support execution of interoperability processes, an ontology-based and goal-driven (OBGD) semantic service bus is designed. This bus is based on a symmetric mechanism for OBGD service invocation. The mechanism is designed according to OBGD Simple Object Access Protocol (SOAP) which is composed of three parts: OBGD message format, OBGD module and OBGD processing model. This mechanism has three properties: location transparency, semantics transparency and technique transparency, which are critical to execution of interoperability processes. The bus also supports federated deployment for inter-enterprise interoperability. PBMEI and the OBGD bus together constitute a federated approach for solving interoperability problems.

**Keywords:** Enterprise Interoperability, collaboration, business process, collaborative process, interoperability process, transformation, rank, cooperation rate, ontology, semantic annotation, Semantic Service Bus, MDA, MDI, SOA, BPMN2.0, goal-driven, SOAP, comparison framework