Knowledge-based system for collaborative process specification

Vatcharaphun Rajsiri

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *l'Institut National Polytechnique de Toulouse*
Discipline ou spécialité : *Systèmes Industriels*

Présentée et soutenue par *Vatcharaphun RAJSIRI*
Le *3 mars 2009*

Titre : *Knowledge-based system for collaborative process specification*

JURY

Chihab HANACHI	*Professeur Université Toulouse 1*	*Examinateur*
Keith POPPLEWELL	*Professeur Coventry University*	*Rapporteur*
Michele MISSIKOFF	*Professeur Lab for Enterprise Knowledge & Systems*	*Rapporteur*
Khalid BENALI	*HDR Universite Nancy 2*	*Rapporteur*
Hervé PINGAUD	*Professeur Ecole des Mines d'Albi-Carmaux*	*Directeur de thèse*
Jean-Pierre LORRE	*Directeur R&D EBM WebSourcing*	*Examinateur*
Frédérick BENABEN	*Maître-Assistant Ecole des Mines d'Albi-Carmaux*	*Examinateur*

Ecole doctorale : *Ecole Doctorale Systèmes*
Unité de recherche : *Centre de Génie Industriel, Université Toulouse, Mines Albi*
Directeur(s) de Thèse : *Pr. Hervé PINGAUD*

Acknowledgement

Foremost, I would like to thank my supervisors, Professor Hervé Pingaud and M. Fréderick Bénaben, who shared with me a lot of their expertise and research insight. Their thoughtful advice often served to give me a sense of direction during my PhD. I am deeply grateful for their detailed and constructive comments, and for their important supports throughout this work.

I also like to express my gratitude to M. Jean-Pierre Lorré, R&D Director of EBM WebSourcing, who brought me into the world of technology and with whom I learnt about how to attack theory, methodology and experimental activities in a balanced way.

I wish to thank Professor Keith Popplewell, Professor Michele Missikoff and M. Khalid Benali for accepting to review and examine my PhD thesis.

I warmly thank also to Professor Chihab Hanachi for his interest in my work and accepting to chair my thesis defence.

During this work I have collaborated with many colleagues for whom I have great regard, and I wish to extend my warmest thanks to all those who have helped me with my work in both EBM WebSourcing and Centre de Génie Industriel of Ecole des Mines d'Albi.

I owe my most sincere gratitude to M. Bertrand Escudié and M. Gaël Blondelle who gave me the opportunity to work with them at EBM WebSourcing. Special thanks to my colleagues, Seb, Christophe, Olivier, Anne-Marie, for their warm welcome, their kindness and their friendship.

My sincere thanks are due to the members of Centre de Génie Industriel of Ecole des Mines d'Albi, particularly Professor Lionel Dupont and Miss Isabelle Fournier, for their warm welcome and their assistance during these three years.

Many thanks to my friends: Jelly, Aey, Jane, Fone and Lenx for long chatting and enjoying our life in France together…I just realized that we've known each other for almost ten years!, Jihed and Seb for your kindness and pleasant working time we had during the last three years…

Finally, I wish to express my love and gratitude to all my beloved family: my parents, my sister Noon and my dearest Thacha for their loving support and never advising me to quit this work. Without their encouragement, understanding and support it would have been impossible for me to finish this work.

Table of contents

Table of figures

Table of tables

General Introduction

Enterprises are now operating in an environment where markets are more open, globalized, and competitive. Changes in market conditions are obliging enterprises to become involved in various kinds of industrial networks in order to maintain their business efficiency. Different forms of networks are emerging continuously and progressively and their structure is becoming more flexible. The capacity of networked enterprises to adapt and react rapidly to market developments is the key factor in ensuring their survival.

The efficiency of networked enterprises is determined by the speed and accuracy with which information can be managed and exchanged among the business partners. Enterprises normally use information systems to manage their internal information. The ability to capture and share information seamlessly between the information systems of different enterprises is therefore very important, given the heterogeneities in culture, language, business, or technology.

New technologies have been emerging exponentially in order to address this question of interoperability.

By definition, interoperability is the ability of two or more systems or components to exchange information and to use the information that has been exchanged [IEEE, 1990]. Interoperability can be seen as the capacity of enterprises to structure, formalize, and present their knowledge and know-how in order to be able to exchange or share it. In this case, interoperability is a crucial requirement for enterprises that need to be dynamically integrated. Interoperability is thus essential for ensuring the economic strength of the enterprise.

This dissertation addresses the use of knowledge engineering for dealing with enterprise interoperability. It is complementary to the work of EBM WebSourcing and to the MISE project.

EBM WebSourcing is an IT company found in 2004. EBM WebSourcing focuses on providing solutions for the integration of enterprise applications, for information exchange in highly distributed networks, and for SME ecosystems. These solutions are open source and based on BPM (Business Process Management) and SOA (Service Oriented Architecture). The work developed in this dissertation can be positioned in the design time (left side) of the company's perspectives. It provides collaborative process models for the runtime collaborative platform. The following schema shows the company's perspectives:

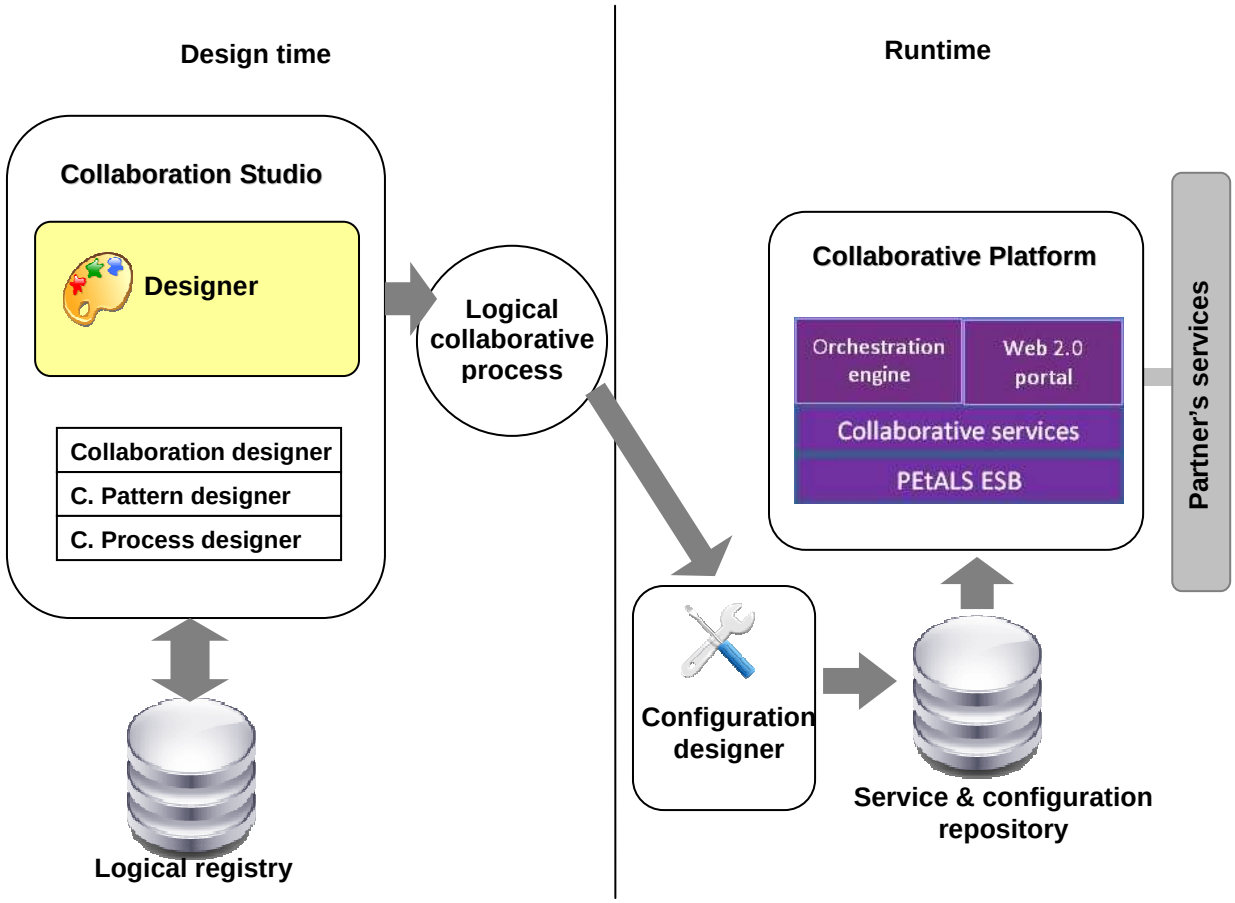


Fig.Intro. EMB WebSourcing's perspectives

The MISE (Mediation Information System Engineering) project was launched in 2004 with the aim to deliver an information technology solution, namely MIS (Mediation Information System), to support enterprise interoperability. The main leaders of this project are the Centre de Génie Industriel of the Ecole des Mines d'Albi-Carmaux and EBM WebSourcing. The first PhD dissertation was written by Touzi, defended in late 2007 and followed up with a post-doc. His work addresses the conceptualization of the logic and technological models of MIS. The work presented in this dissertation can complement this first PhD by providing its input (collaborative process model).

The concept of MIS is based on MDE (Model Driven Engineering) [Millet et al., 2003] and consists of three levels: CIM (business), PIM (logic), and PSM (technological). In our context (enterprise interoperability), the CIM level concerns the organization, objectives, processes, and responsibilities of participating enterprises. It requires a model that can represent interactions occurring between enterprises, including the exchanged data, services exposed to others, etc. The PIM level requires a model representing the different functional entities of a system, expressed in terms of enterprise logic. Such logic concerns the semantic aspect of knowledge and information. The PSM level requires a model for generating executable code in the definition of a technological platform, such as a service model. The MDE approach allows the complexity of the information system design to be reduced by separating the business and technological needs. This approach proposes to design across the different abstraction levels and exploits at each level the associated models to build the models of the

next level. The works of Touzi (PhD and post-doc) cover the PIM and PSM levels by using SOA as the MIS's logical model and PEtALS ESB (provided by EBM WebSourcing) as the target architecture that facilitates the communication between information systems. SOA contributes the agility, accessibility, and diminution of complexity of an information system. These contributions allow us to manage the communication difficulty between information systems. However, it obliges the information systems of business partners to be based on SOA. The work of Touzi deals particularly with the transformation of models between the MDE's levels. The Translator V1.0 was developed in response to this transformation using ATL (Atlas Transformation Language). In these works, Touzi supposes that business partners can provide and make available the collaborative business process which is the main input of the Translator V1.0. We realized later that this assumption was not reasonable from the user-oriented perspective. But, how can we obtain that explicit collaborative process model?

We answer the above question by applying knowledge-based engineering to the design of a collaborative process model. This is the purpose of the work described in this dissertation. The aim is to accomplish the CIM level and deal with the business level of enterprise interoperability by providing a common collaborative process model. Here, we shall work at a higher abstraction level than the CIM for capturing as much knowledge as possible (e.g. skills, roles and business goals) from business partners. The existing knowledge in reference repositories regarding business services is reused in order to support the design of the collaborative process, and to provide a solution that is intrinsically adapted to an SOA implementation. Thanks to the knowledge from partners and the reference repositories, we can then go down to the CIM level. This approach brings out three crucial topics: knowledge gathering, knowledge representation and reasoning, and collaborative process modelling. Each topic is introduced in the three following paragraphs.

Firstly, we address the characteristics of collaboration in order to define what knowledge needs to be captured. Collaboration is a group of more than two partners who establish relationships and processes among themselves [ICC, 1999] [Kak et al., 2002]. A common goal is shared and defined collectively in order to provide a common understanding and the direction to be taken. Collaboration requires information to be transferred, and needs a collaborative process that business partners carry out together in pursuit of the common goal. Thus, we need to gather knowledge that is relevant to these criteria. We will discuss these criteria and this knowledge in detail in Chapter 2. Most of this knowledge can probably be retrieved from the previous collaboration experiences of business partners. The precision and accuracy of collaboration characterization impacts on the results of the other two topics (knowledge representation and reasoning, and collaborative process modelling). More knowledge captured leads to a more accurate characterization of collaboration. Nevertheless, this knowledge is implicit. So, how can we capture such knowledge and make it more explicit?

Secondly, we focus on representing and reasoning knowledge captured from the first topic. This knowledge may be used for various purposes, such as building a partner knowledge repository, or a collaboration knowledge base, etc. However, in this dissertation, we are interested in the use of knowledge for modelling collaborative process. The first question in this topic is, once the knowledge is collected, how is it stored and reused for collaborative process modelling? This question is not easy to answer due to the many constraints of collaborative process modelling and the behaviour of knowledge. Moreover, the knowledge

captured from business partners concerns the collaboration domain, not the collaborative process field. So, how can we move from the knowledge of the collaboration domain to that of the collaborative process?

Thirdly, we aim to model collaborative process. So, we may need to define more clearly what collaborative process is. The result of the second topic is not yet our expected collaborative process model, but just a pack of knowledge usable for modelling collaborative process. The extraction of such knowledge and the representation of it as collaborative process are the crucial issues to be addressed. As we mentioned previously, we can used this collaborative process model as an input of the Translator V1.0 for defining an SOA-based MIS model [Touzi, 2007]. The crucial questions in this topic are: how do we extract that knowledge?, how can we model collaborative process from that knowledge?, and how do we ensure that our collaborative process model can be used without any problem by the Translator V1.0?

The above discussion and identified questions reveals that each topic covers a wide range of aspects and issues which are somehow related to each other. We need to study them into detail in order to gain a clear understanding of them and find a way to manage them. Thus, the dissertation is organized as follows:

- Chapter 1 focuses on describing the scope and objectives of the dissertation. We are interested in defining the problematic of establishing collaboration between enterprises (the reasons why enterprises collaborate, and the necessity of technologies supporting collaborations). The hypotheses that directly impact the dissertation mainly concern the different collaboration capabilities and the interoperability of enterprises through their information system.

- Chapter 2 is the literature study. We explore the concepts, approaches, and technologies related to the first and third topics discussed previously. It consists of: 1) the notion of inter-enterprise collaboration including the characterization of collaboration and enterprise knowledge, and 2) the definition of collaborative process together with the modelling approaches and languages.

- Chapter 3 presents a knowledge-based system for accomplishing the business level (CIM) of the MDE. We explore the methods and technologies of knowledge representation and reasoning (second topic) and explain globally how the studies in Chapter 2 are adopted to develop the system. We then focus only on an engineering approach for dealing with the second topic which makes up the heart of the system. This approach involves morphing knowledge on collaboration captured from business partners into knowledge concerning collaborative process.

- Chapter 4 focuses on presenting the prototype developed for supporting the knowledge-based system introduced in Chapter 3. Such a prototype is run semi-automatically. It consists of four steps: acquisition of implicit knowledge from business partners, deduction of knowledge, extraction of knowledge, and construction of collaborative process models. The first step concerns the knowledge gathering (first topic), while the last two steps concern the collaborative process modelling (third topic). The second step is discussed in Chapter 3. Each step has specific requirements. We will benchmark several technologies and select an appropriate one for developing

a specific tool to satisfy the requirements of each step of the prototype. The prototype is seen as a chain of several different tools.

- Chapter 5 aims at demonstrating how the prototype works by experimenting with a very simple collaborative situation. Another complex collaboration will also be conceptually presented. The experimentation uses every tool of the prototype to perform every step, from knowledge gathering to the construction of the collaborative process

- Chapter 6 ends the dissertation by giving the conclusions and perspectives of this work. We propose a summary of work carried out during this PhD and the outlook related to the different current studies.

Chapter 1. Scope and Objectives

The aim of this dissertation is to provide a knowledge-based system dedicated to the specification of a relevant collaborative process from a given collaborative situation. The collaborative process generated from this system will be used for developing an information system mediator.

This chapter focuses on describing the scope and objectives of the dissertation. Firstly, we seek to define the problem of collaboration between enterprises including the reasons why enterprises collaborate, and the need for technologies to support collaboration. Then, we present the hypotheses of this dissertation drawn from the previous works carried out in the MISE project. Such hypotheses bring out the different collaboration capabilities and the interoperability of enterprises. The objectives and framework of the dissertation are specified, taking into account the previous works. Finally, we present an overview of the work of this dissertation.

1. Problem of inter-enterprise collaboration

In past decades, enterprises could operate alone in relatively stable and predictable environments. The dispersal of information and the exponential emergence of new technologies have started to erode the stability of this environment. Operating in such environments is becoming increasingly more difficult.

Now the environment has completely changed and the market is more open and globalized. Enterprises, particularly small and medium-sized enterprises (SMEs), are facing competition from large organizations for market share and profits in business. Market trends oblige companies to become involved in many kinds of industrial networks in order to maintain their business efficiency. The capacity of enterprises to collaborate or interact with their partners is a crucial factor for their development and their ability to survive. Indeed, enterprises require the agility to be able to operate under such pressures and thus ensure their survival.

The need for agility has greatly influenced the traditional way enterprises are run. Enterprises have started developing and establishing more and more collaborative projects in response to various challenges. These projects may include group of partners that have complementary skills, groups of businesses in the same market, platforms for group buying, etc. Different forms of networks keep emerging continuously and progressively and their structure is becoming more flexible.

A business network is an open and agile system that evolves and adapts itself regularly along its life cycle. The main challenge of creating such networks is the selection of appropriate business partners and the implementation of appropriate processes. However, once a network is formed its efficiency is determined by the speed and accuracy with which information can be managed and exchanged among the business partners. Its life cycle consists of several

phases: identification, formation, design, operation, development, and dissolution. [D1.1 Synergy, 2008]

Besides this new trend of networked enterprises and the complexity of collaboration, new technological requirements are necessary for supporting collaboration in networks, such as collaborative platforms. In general, business partners are geographically distributed and interact with each other frequently. The diversity of business process categories developed inside the network is as wide as the variety of types of collaboration between those business partners. Thus, collaboration needs to be supported by technologies in order to facilitate the interactions between partners, and maintain the inter-enterprise relationships. The ability to capture and share information seamlessly amongst a suite of software systems is very important as it reduces data-handling errors and facilitates business activities. However, this feature is not always available among the commonly used software applications and it is costly to many globally distributed industries [D1.1 Synergy, 2008].

2. Hypotheses and objectives of the dissertation

This dissertation is a work in collaboration between the Centre de Genie Industriel of the Ecole des Mines d'Albi-Carmaux and EBM WebSourcing. The main purpose of this dissertation is to provide a collaborative process model that describes interactions and information exchanges between business partners. Such a model can be applied to the previous works of the MISE project and the on-going works of EBM WebSourcing. Thus, this dissertation is related both to the MISE project, and to the requirements of EBM WebSourcing.

The aim of this section is to present the principal hypotheses and constraints originating from the previous works, in particular those of the MISE project, which impact directly on this dissertation. Finally, the objectives and scope are also specified based on the previous works.

2.1. Principal hypotheses

Due to the current global and competitive market, the capacity of enterprises to collaborate with their partners is the critical factor for their development and their ability to survive. [Touzi, 2007] defined four levels of capacity: communicating (ability to exchange and share information), open (ability to share business services and functionalities with others), federated (ability to work with others by following collaborative processes in order to pursue a common objective, as well as objectives of the enterprise itself), and interoperable (ability to work with others without a special effort; the enterprises involved are seen as a seamless system). Thus, interoperability can be seen as an alternative to performing the integration of enterprises into a unique system [Vernadat, 2006].

The concept of interoperability first appeared in the domain of computer science in the early 1990s and has been developed continuously and extensively in many domains such as military, medical, transportation, software, etc. [DoD, 2001] [ISO 16100, 2002] [APFA, 2004]. Since then, many definitions related to this concept have been proposed. The most quoted one was given by [IEEE, 1990] which defines interoperability as the ability of two or

more systems or components to exchange information and to use the information that has been exchanged. Furthermore the InterOp NoE defines the interoperability as the ability of a system or a product to work with other systems or products without special effort from the customer or user [Konstantas et al., 2005]. We can deduce from these definitions that the implementation of the interoperability between enterprises is not easy to achieve due to the large number of interoperating components and the semantic heterogeneity of the systems involved.

Four years ago, the MISE project was launched with the aim to conduct experiments in the enterprise interoperability domain. Its investigation was driven by a main issue: to deliver an IT solution for a collaborative network in the context of future developments in the internet and by promoting interoperability concepts.

We realized the fact that collaboration is a complex system which frequently changes. Enterprises involved in collaboration are geographically distributed. Collaboration requires technologies for supporting the interactions between enterprises. The ability of technologies to efficiently manage and exchange information among software systems has become a very important factor.

An information system (IS) is used internally in an enterprise in order to manage, store, and deal with information. It can be considered as a set of data, applications, and processes [Morley, 2002]. IS may also be used to interface with external entities. This offers enterprises the possibility to communicate with their partners through their IS [Touzi, 2007]. The efficiency of this collaboration depends mostly on the interoperability capability of the IS. Nevertheless, the heterogeneity of enterprises in terms of culture, language, business, or technology make the interoperability complex and can lead to the failure of interoperability between systems.

A solution to deal with this issue has been proposed by [Touzi, 2007], namely MIS (Mediation Information System). The MIS was evolved to be used as a mediator (intermediate platform) that connects to the ISs of different partners in order to drive the concept of system of systems [Maier, 1998]. The MIS together with the partners' ISs are seen as a single system. According to [Wiederhold et al., 1997], a mediator comprises a layer of intelligent middleware services in IS, providing intermediary services that link data and application programs without the need to integrate the data bases. Thus, [Bénaben et al., 2008] proposed that the MIS should be able to handle: 1) conversion and delivery of partners' data, 2) management of a repository of partners' services and applications, and 3) orchestration of a collaborative process model that should be run over a workflow engine. Fig.I. 1 depicts the concept of MIS:

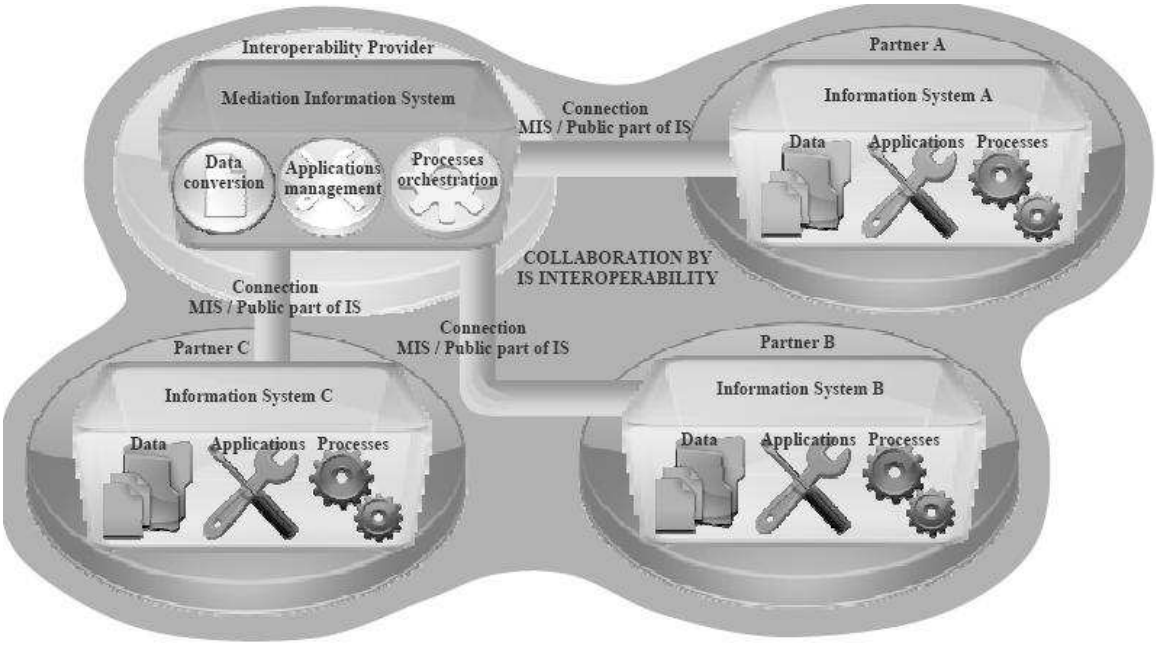


Fig.I. 1 Concept of MIS [Bénaben et al., 2008]

However, defining such a solution is not an easy task. The technical components (e.g. database, ERP, web service, etc.) of partners' ISs should work together in order to meet the business requirements expressed by the partners. Indeed, the configuration of these components is not only a problem at a technological level, but it concerns the conceptual and organizational levels as well. The main investigation of [Touzi, 2007] is to tackle the interoperability problems vertically by going from the organizational level to the technical level according to the MDE (Model Driven Engineering) approach. Thus, the generation of the MIS is based on three levels of the MDE approach: the organizational, logic, and technological levels.

The organizational level addresses the representation of interactions occurring between enterprises, including the exchanged data, resources exposed to others, etc. The collaborative process model written in BPMN (Business Process Modelling Notation) was chosen to represent such interactions. Why BPMN? BPMN is an advanced formalism which can cover mainly the functional view and partially the organizational, informational, and resource views of enterprise modelling. It offers dynamic behaviour (e.g. an event) to business process which is an important aspect of process modelling.

Based on the definitions of interoperability given so far, it allows open access to IS resources. Clearly, it needs to be controlled somehow to protect the privacy of the enterprise. Only the public part of an enterprise's IS should be visible to others. Service-Oriented Architecture (SOA) is a perfect solution to deal with this issue [Touzi, 2007]. SOA contributes to the agility, accessibility, and reduction of the complexity of an IS. According to [Touzi et al., 2008], SOA provides the means to obtain a loosely coupled architecture describing collaboration between autonomous systems, in contrast to classical tightly coupled systems and monolithic architectures. These autonomous systems are represented as services and have independent lifecycles. Indeed, enterprise applications and internal processes can be encapsulated as services. SOA solutions are designed to manage and orchestrate bonds

between applicative services within a process trade. SOA is designed to provide the flexibility to treat elements of business processes and the underlying IT infrastructure as components (or services) that can be reused and combined to address changes of business priorities. In this way, SOA can be used efficiently to facilitate the communication between enterprises when each exposes their services to others. This leads to the main assumption of the MIS regarding the fact that the ISs of enterprises must follow the same conceptual SOA model. [Touzi, 2007] considered the SOA model as a logical level of the MDE. The overview of SOA will be presented in Chapter 2.

The technical level of the MDE concerns ESB (Enterprise Service Bus) architecture. EBM WebSourcing provides an open source PEtALS[1] ESB which was selected to support this level.

EBM WebSourcing is the main industrial partner of this dissertation and a member of OW2 open source consortium. Thus, our principal constraint is that the tools and technologies used in this dissertation should be open source.

2.2. Principal objectives and scope of the dissertation

In this section, we define the scope of the dissertation by taking into account the above hypotheses including the works of the MISE project and EBM WebSourcing's requirements. We start by positioning our work on the MDE approach. Then, because our work deals with the interoperability of enterprises' IS, we need to study the different interoperability frameworks in order to understand the existing standards, architectures, and approaches in this domain. Then, we shall try to position the MISE project, including the work of this dissertation relating to it, for a better understanding of its objectives and background.

2.2.1. Positioning the dissertation on the MDE approach

According to the hypotheses discussed previously, the generation of MIS is based on the MDE approach. MDE was created as a promising approach for dealing with the complexity of platforms [Schmidt, 2006]. The MDE technologies combine: 1) domain-specific modelling languages which formalize the application structure, behaviour, and requirements within particular domains by using meta-models, and 2) transformation engines and generators that analyse certain aspects of models and artefacts, such as source code, XML deployment descriptions, or alternative model representations. MDE technologies focussing on architecture and corresponding automation yield higher levels of abstraction in software development. The OMG[2] (Object Management Group) launched a set of standards called Model-Driven Architecture (MDA) in 2001 for supporting the MDE of software systems.

According to [Millet et al., 2003], MDA is an approach dedicated to the development of software systems. It provides guidelines for structuring specifications expressed as models. MDA distinguishes the business and technological branches. The business branch consists of three abstraction levels:

[1] http://petals.objectweb.org/

[2] OMG™ is an international, open membership, not-for-profit computer industry consortium. OMG Task Forces develop enterprise integration standards for a wide range of technologies, and an even wider range of industries.

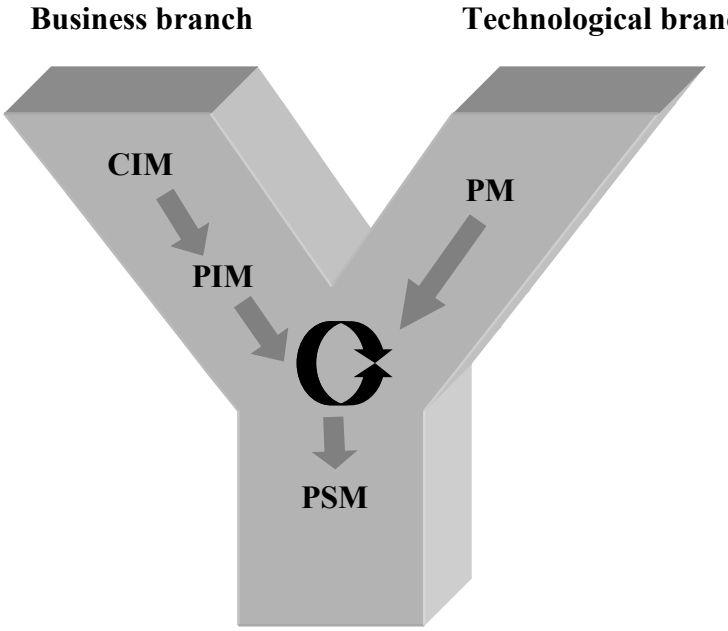


Fig.I. 2 Business and technological branches of MDA

- CIM (Computer Independent Model) is a model of a system that shows the system in the environment where it will operate. It helps in understanding a problem and defining a shared vocabulary for use in other models.

- PIM (Platform Independent Model) consists of enterprise, information, and computational viewpoint specifications.

- PSM (Platform Specific Model) is a view of a system from the platform-specific viewpoint. This level is the combination point of business and technological branches. It describes the realization of software systems by combining the specifications of PIM with the technical architecture of the platform model (PM).

Based on the MDA approach, the transformation can occur in the same (horizontal) or different (vertical) abstraction level. According to [Merilinna, 2005], the horizontal transformation is for example, PIM to PIM transformation when models are enhanced, filtered, and specialised during the design process. However, the vertical transformations can be considered the key transformations, as they push the system under development from specifications all the way to the actual source code.

In the framework of the MISE project, we address only the vertical transformations which concern the change of abstraction levels from CIM to PIM and from PIM to PSM. The works of Touzi start from the collaborative process model with the assumption that business partners are able to provide this model. The model transformation is proposed using model morphism and model-mapping theories. Another assumption used in this transformation phase is about the structure of the logical architecture which is SOA compliant, with respect to the idea to implement the PEtALS ESB. Fig.I. 3 depicts the big picture of the MISE project including the works of Touzi (PhD and post-doc):

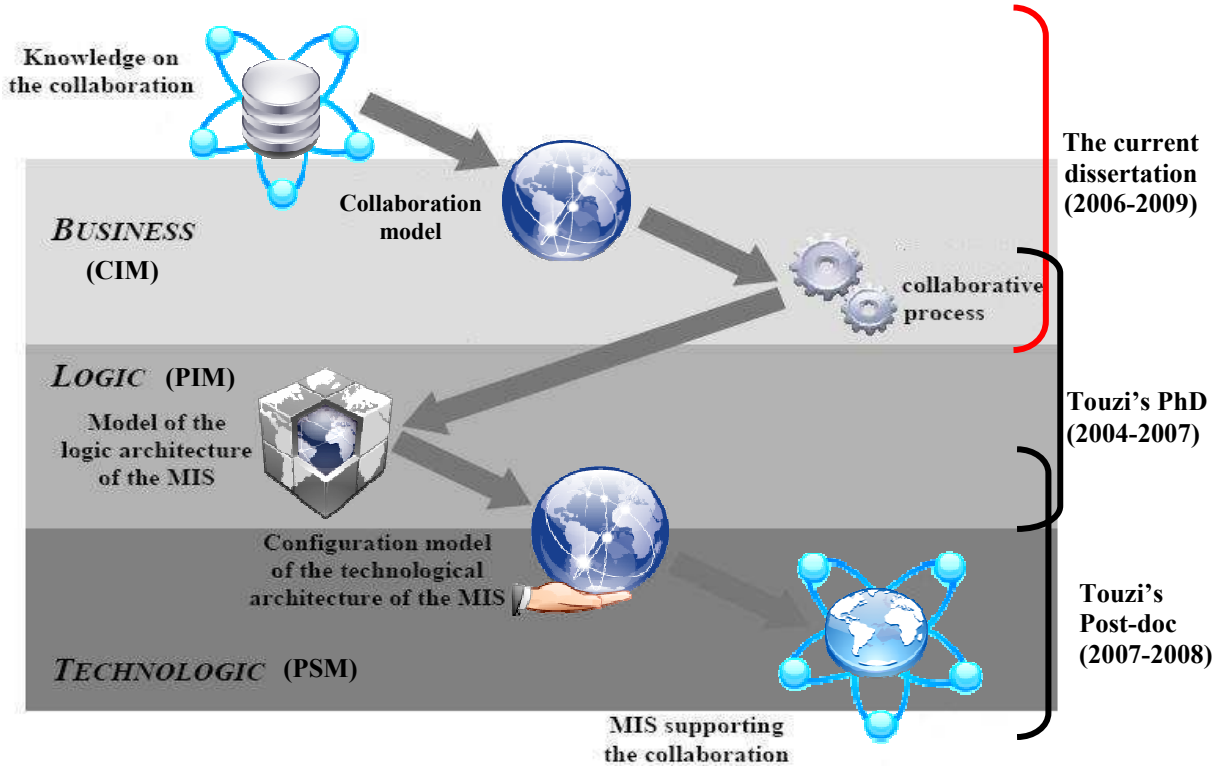


Fig.I. 3 Global principle of MIS design through a Model Driven Approach [Bénaben et al., 2008]

If we position this dissertation on the MDA approach in comparison to the previous works of the MISE project, we realize that:

- Collaborative process model supports the CIM level.
- SOA-based MIS supports the PIM level. This is on the assumption that ISs of the involved partners follow the same SOA conceptual model.
- PEtALS ESB supports the PSM level.

However, the assumption about the collaborative process model that it should be available and provided by partners is not reasonable from the user-oriented perspective. It is unlikely that business partners will have their collaborative process in place before starting the collaboration. Furthermore, this collaborative process should conform perfectly to the requirement (meta-model) defined at the CIM level. Thus, the work developed in this dissertation was originally created to overcome this problem. Our work tries to accomplish the business (CIM) aspect by using knowledge-based engineering. We are working at the upper level of the CIM to capture knowledge on collaboration and exploit this knowledge in order to move down to the CIM. The concept of this work will be introduced theoretically in Section 3.

2.2.2. Interoperability frameworks

The problem of enterprise interoperability relies generally on three levels: data, resources and business processes. Interoperability reinforces the idea that integration has to be prepared by using standards, reference frameworks, specific architectures, and approaches.

As we aim to deal with enterprise interoperability, we should study the standard interoperability frameworks. A number of interoperability frameworks have been developed in various domains: IDEAS (Interoperability Development for Enterprise Application and Software), AIF (ATHENA Interoperability Framework), EIF (Enterprise Interoperability Framework), e-GIF (e-Government Interoperability Framework), EIF (European Interoperability Framework), etc. We have presented only the first three frameworks in this dissertation because they concern the enterprise interoperability domain in which we are interested. The two last frameworks were specifically developed to apply to interoperability between governmental services and administrative services.

2.2.2.1. IDEAS

IDEAS stands for Interoperability Development for Enterprise Application and Software. It is a thematic network project intending to deliver roadmaps in the domain of interoperability of enterprise applications and software. It defines interoperability as an interaction capability between enterprise software applications.

According to [IDEAS, 2003], the IDEAS interoperability framework has three levels: business, knowledge, and ICT (Information and Communication Technology) systems. These three levels are related to each other by a common semantic. They should be considered all together in order to obtain substantial and effective results, as well as pragmatic applications in today's business world. The IDEAS interoperability framework describes how interoperability can be achieved if the interactions can at least take place at three levels between two cooperating enterprises. Fig.I. 4 shows the IDEAS framework:

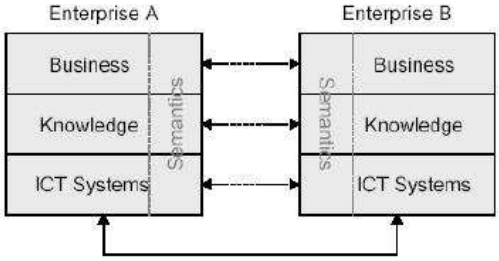


Fig.I. 4 IDEAS interoperability framework [IDEAS, 2003]

- The Business level concerning the problems related to the organization, and business processes. It is divided into three sub-levels: decisional model, business process, and business model.

- The Knowledge level concerns acquiring, structuring, and representing knowledge of enterprises.

- The ICT system level (application and data) concerns the technical solutions for transferring resources from one enterprise to the others.

The semantic barriers concern mutual understanding on all layers, for example, business terms for the business level, dictionaries and ontologies for the knowledge level, and ontology tools and services for the ICT level. This barrier concerns every level of the enterprise when it establishes interoperability with others.

2.2.2.2. AIF

AIF stands for ATHENA Interoperability Framework [ATHENA, 2004]. This framework adopts a holistic perspective on interoperability by integrating the different results and solutions developed in the ATHENA project. It builds upon the thematic network of IDEAS and merges three research areas: 1) architecture and platform to provide implementation frameworks, 2) enterprise modelling to define interoperability requirements and to support solution implementation, and 3) ontology to identify interoperability semantics in the enterprise. The AIF aims to provide approaches to the solution, while the IDEAS framework focuses on structuring the interoperability issues (into business, knowledge, semantic, and technologic issues). According to [Berre et al., 2007], the AIF is structured into three parts as follows:

- Conceptual integration focuses on identifying the concepts, meta-models, languages, and model relationships for developing interoperability. Fig.I. 5 is a simplified view of the reference model that indicates the required and provided artefacts of two collaborating enterprises:

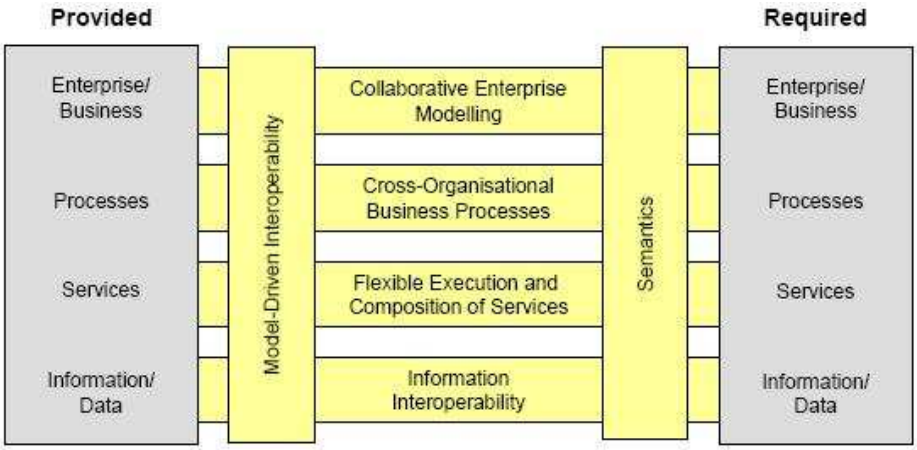


Fig.I. 5 ATHENA reference architecture [Berre et al., 2007]

Interoperations can take place at four levels:

- The Enterprise/business level concerning the organizational and operational ability of an enterprise to cooperate with others. This level requires a collaborative enterprise modelling.
- The Process level focusing on making various processes work together. A cross-organizational business process is needed at this level.

- The Service level concerned with identifying and executing applications. This requires a flexible execution and composition of services which can be supported by PIM4SOA (PIM for SOA).
- The Information level concerning the management and exchange of messages (information interoperability).

For each level, the model-driven interoperability approach is used to formalize and exchange the provided and required artefacts that must be negotiated and agreed upon. The semantic annotation is to give meaning to any kind of resources (e.g. business processes) in terms of the shared reference ontology and reconciliation rules.

- Applicative integration focuses on methodologies, standards, and domain models. This part ensures the establishment of interoperability by providing guidelines, principles, and patterns that can be used to solve interoperability problems.

- Technical integration concerns the technical development, and ICT environments. It provides the ICT tools, and platforms for developing and executing enterprise application, and software systems.

This platform provides a compound framework and associated reference architecture for capturing the research elements and solutions to solve the interoperability problems. It addresses the problem in a holistic way by capturing and inter-relating information from many perspectives covering business, knowledge, technical (ICT), and semantic issues relevant to interoperability.

2.2.2.3.EIF

The EIF stands for Entreprise Interoperability Framework, developed in the InterOp NoE project (Interoperability Research for Networked Enterprise Applications and Software, FP6 508011) [Chen et al., 2006]. This framework aims at defining the research domain of interoperability of enterprise applications. Generally, research on interoperability is an applied and problem-driven type of research. The framework has three basic dimensions: interoperability levels, interoperability barriers, and interoperability approaches.

The interoperability levels concern the interactions that can take place from the various viewpoints of the enterprise. Four levels of interoperability have been defined:

- Interoperability of data aims to make different data models related to particular applications work together. This interoperability allows data coming from heterogeneous bases to be found and shared by different machines, operating systems, and database management systems.

- Interoperability of services concerns identification, combination, and making various applications work together by dealing with syntactic and semantic differences.

- Interoperability of processes focuses on making various processes work together.

- Interoperability of business refers to working in a seamless way at the organizational level in spite of the different modes of decision-making, culture, etc., so that business can be developed and shared between companies.

The interoperability barriers describe an incompatibility which obstructs the sharing of information and exchanging of services. Three categories of barriers have been defined as follows:

- Conceptual barriers relating to the syntactic and semantic differences in information to be exchanged, as well as in the expressivity of the information.

- Technological barriers relating to the incompatibility of information technologies (e.g. architecture, infrastructure, etc.). This kind of barrier prevents collaboration between two or more systems.

- Organizational barriers relating to the definition of responsibility, authority, and organization structure. This kind of barrier particularly concerns human and organization behaviour which can be incompatible with interoperability.

These approaches to interoperability allow knowledge and solutions relating to enterprise interoperability to be categorized according to the ways of removing various interoperability barriers. They have been defined under three categories:

- Integrated approach, referring to the existence of a common format for all models. If the need for interoperability comes from a merger between enterprises, this approach seems the best adapted. In this case, there is only one common format agreed by all partners to elaborate models and build systems.

- Unified approach, referring to the existing of a common format, but only at a meta-level. If the need for interoperability concerns a long-term based collaboration, this approach is a possible solution. A common meta-model provides a means for establishing semantic mapping between models.

- Federated approach, having no common format. Partners do not impose their models, languages, or methods of work. For the need for interoperability originating from the short-term collaboration project, this approach is the most relevant. To interoperate, partners must adapt themselves dynamically by sharing an ontology rather than having a predetermined meta-model.

2.2.2.4. Conclusion

AIF is a solution-oriented framework built upon IDEAS which tries to structure the interoperability problems into business, knowledge, technology, and semantic aspects. According to [Chen et al., 2006], EIF (Enterprise Interoperability Framework) adopts the barriers-driven approach to tackle interoperability problems in a bottom-up way to remove the barriers. IDEAS and AIF do not explicitly represent the barriers or obstacles to interoperability. Neither are aimed at structuring interoperability knowledge with respect to its

contribution to removing various barriers. Three barriers of EIF can be considered as organizational (business), conceptual (semantic), and technological levels of interoperability.

These three frameworks have some similarities. Fig.I. 6 illustrates the comparison of these three frameworks:

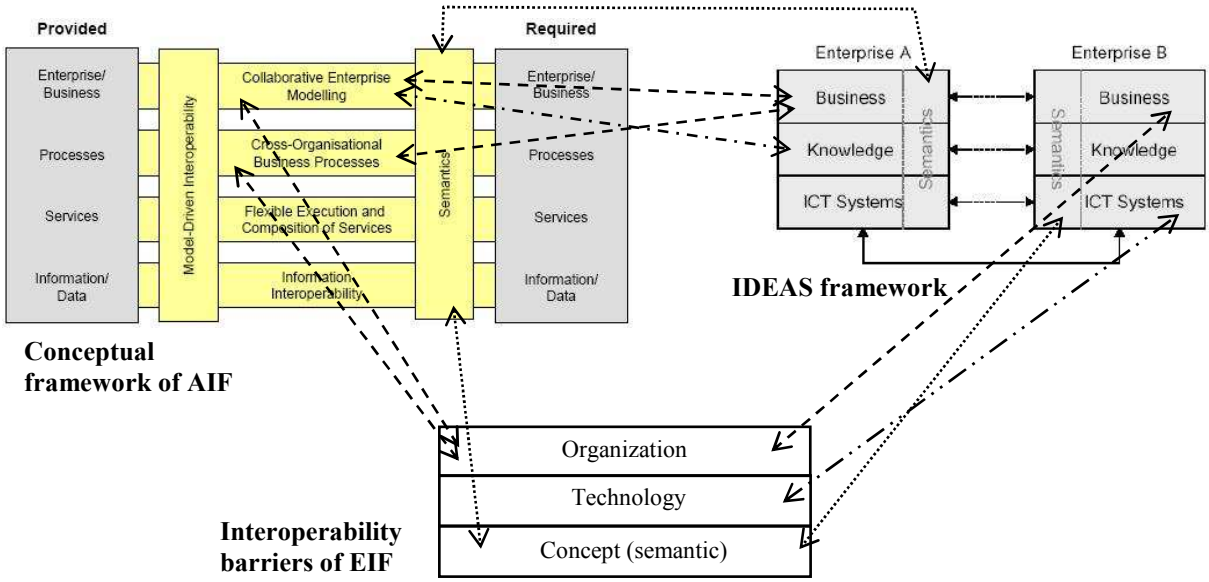


Fig.I. 6 Comparison of IDEAS, AIF, and EIF frameworks

- Firstly, both the organizational level of EIF and the business level of IDEAS define the exchange of objectives, strategies, responsibilities, and the operational modes of the enterprise. This exchange can be based on the use of models and languages understandable by the enterprises as defined at the enterprise and process levels of the conceptual framework of AIF. See the dashed lines.

- Secondly, the knowledge level of IDEAS addresses the enterprise knowledge representation which may be included in the business level of the AIF. See the dash-dot line.

- Thirdly, the technology level of EIF corresponds to the ICT systems of IDEAS because they deal with the incompatibility of information technologies including applications and data. See the dash-dot-dot line.

- Finally, the conceptual level of EIF and the semantics dimension of both AIF and IDEAS concern the signification and the interpretation of the information and knowledge of an enterprise. However, the IDEAS and AIF frameworks deal with the semantic across different levels. See the dot lines.

Even though these three frameworks provide quite similar approaches for tackling enterprise interoperability, EIF also takes into account the two other dimensions of interoperability. Besides, EIF aims at removing the barriers that block and lead to the failure of interoperability. These points make the EIF framework more interesting for positioning and

analysing our work. In the next section, we will define the scope of the work presented in this dissertation on the EIF framework.

2.2.3. Positioning the dissertation on the interoperability framework

According to Fig.I. 3, we defined the scope of this dissertation based on the hypotheses and the previous works of the MISE project. In this section, we intend to position it on the interoperability framework, since we are adressing enterprise interoperability problems in a collaborative network. Fig.I. 7 illustrates the scope of the MISE project in black, and that of this dissertation in white by their positioning on the EIF:

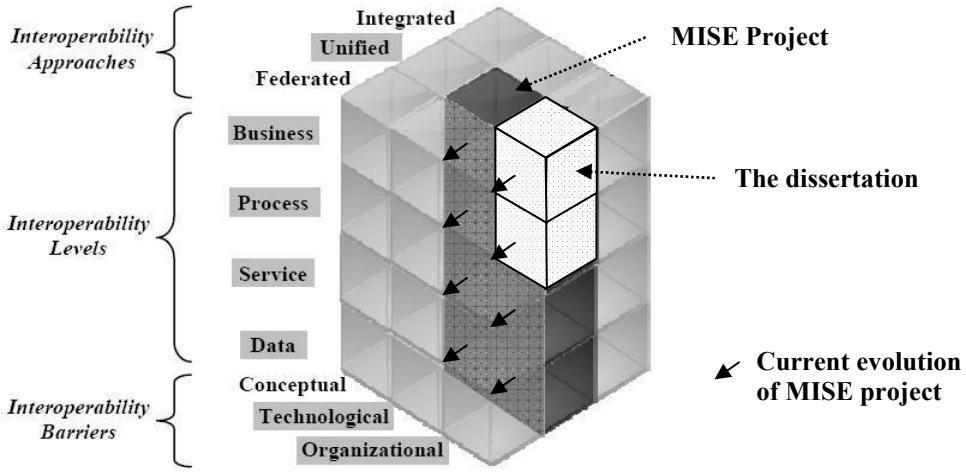


Fig.I. 7 Position of the MISE project and the present work on the EIF of InterOp [Bénaben et al., 2008]

The MISE project copes with the interoperability issues by using a common SOA meta-model of IS. We describe as follows the scope of this project including the work of this dissertation according to these three domains of interoperability:

- Interoperability approaches: The MISE project does not imply any common format for all models, so it does not drive an integrated approach. Neither does it allow complete freedom of models and languages, so it is not federated. But, it is *unified* because we use the SOA meta-model for tackling the interoperability of ISs. Nevertheless, the MISE project is trying to blur the SOA dependency by facilitating and tooling the service presentation of enterprises (the arrows show this evolution towards federated approach).

- Interoperability levels: The project uses collaborative network models to align business goals of involved partners with processes, so it deals with the *business* level. The SOA logic model deals with the other levels: *process*, *service*, and *data*.

- Interoperability barriers: Since the design of MIS follows the MDE approach, it addresses all the organizational (CIM), logic (PIM), and technical (PSM) levels. These three levels concern respectively the *organizational*, *conceptual*, and *technological* barriers of interoperability.

The scope of this dissertation concerns only the *business* and *process* levels and the *organizational* barrier of EIF because we intend to define a collaborative process model for representing a specific collaboration. We take into account the business goals and knowledge captured from the business partners. This collaborative process model contains the information about services and data transfer which come from the MIT Process Handbook. This information does not provide any syntactic and semantic descriptions which are required for the interoperability at service and data levels.

3. Elaboration of a Knowledge-based system for specifying collaborative processes

This dissertation focuses on removing the organizational barrier (EIF) to interoperability and accomplishing the business level (CIM) of the MDE. Thus, our work particularly addresses the design of collaborative process model. This collaborative process model can complement the previous works of the MISE project in order to generate a MIS solution for supporting enterprise interoperability. This process model can also be used afterwards by the collaborative platform solution of EBM WebSourcing.

We propose to work at a higher abstraction level than the CIM for capturing knowledge from collaborative network partners. We shall try to reuse existing knowledge from reference repositories for moving down to the CIM. This idea brings out several crucial topics regarding this work: enterprise knowledge acquisition, knowledge representation and reasoning, and collaborative process modelling. These lead us to develop a knowledge-based system for dealing with the use of knowledge to specify collaborative process models. For constituting such a knowledge-based system, there are several questions concerning each topic to be taken into account:

- Knowledge on collaboration: what knowledge do we intend to use?, where and how can we capture that knowledge?...

- Knowledge representation and reasoning: once we have knowledge, how do we keep or store it? how do we represent it?, how can we reuse the knowledge? The knowledge we captured is about collaboration. So, how do we derive new knowledge about collaborative processes from existing collaboration knowledge?

- Collaborative process modelling: how do we describe the collaborative process? how do we represent the collaborative process model, and by which language?...

To answer the above issues, we need to study and have a clear understanding of each topic. Thus, we propose a framework for studying these three topics as follows:

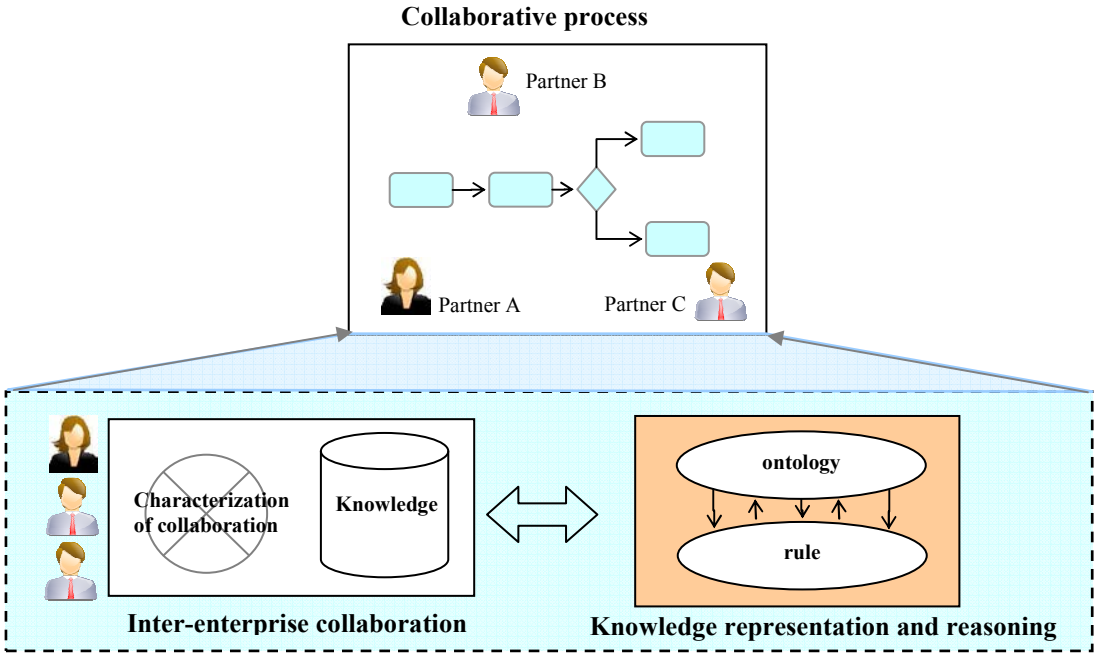


Fig.I. 8 Three-topic framework for defining collaborative process

- Our approach for inter-enterprise collaboration is focused on collaboration in general including its definitions and classifications. The requirements, perspective, as well as the knowledge acquired from experience or the past collaborations of partners should allow us to understand collaboration. The production and exchange of knowledge we can capture during the collaboration are also important. Thus, we shall also discuss the different criteria for characterizing collaborations, enterprise knowledge, and knowledge for collaboration.

- Concepts and technologies for knowledge representation are focused on presenting the different ways for dealing with knowledge. This part also includes the technologies related to the methods for representing knowledge. At the end, we shall select the technologies the most relevant to us for further use.

- Our approach for collaborative process modelling concerns the definition of the collaborative process based on our point of view. The different modelling approaches and languages that we can adapt to the collaboration context are also described.

4. Conclusion of the chapter

This chapter is focused on specifying the scope and objectives of the dissertation. The dissertation can serve the MISE project and EBM WebSourcing. It relies on the organizational level of enterprise interoperability. The main purpose of this dissertation is to provide a collaborative process model corresponding to the specific behaviours found in collaboration. We intend to develop a knowledge-based system for supporting the specification of collaborative processes and guarantee the alignment of these processes with the common goals of business partners.

This dissertation is influenced by the works carried out previously in the MISE project, as well as the requirements of EBM WebSourcing. The principal hypotheses are described in Section 2. They particularly concern the different collaboration capabilities, the interoperability between the ISs of business partners, and the technological aspect, such as the limitations of the collaborative process model, SOA as a logical model of MIS, and PEtALS ESB.

We have highlighted three main topics that are essential for constituting a knowledge-based system: knowledge on collaboration, knowledge representation and reasoning, and collaborative process modelling. We have defined a framework for studying these three topics in detail in order to have a clear understanding of them and to find a way to deal with them. Based on this framework, we have organized the following chapters in the dissertation as follows:

- In Chapter 2, we explore the concepts discussed above in the literature, together with the technologies dedicated to inter-enterprise collaboration (first topic) and collaborative process (third topic). The objective of this chapter is to understand the characteristics of collaboration, the knowledge that drives and supports collaboration, and the modelling approaches that can be applied in a collaborative process context. This understanding offers an overview of our knowledge-based system as it addresses the departure and arrival points of the system.

- In Chapter 3, we explore the methods and technologies for representing and reasoning knowledge (second topic). We describe an overview of our knowledge-based system and how the studies in Chapter 2 can be adopted so as to develop the system. Then, a knowledge engineering approach will be presented in theory, in response to the knowledge representation and reasoning issues which constitute the core of our system.

- In Chapter 4, the prototype of our system will be introduced. This prototype consists of the tools and technologies that support the three main topics presented previously.

- In Chapter 5, a demonstration of how the prototype works will be shown by experimenting with a very simple collaboration case.

Chapter 2. Literature Study

In this literature study, we present the approaches for collaboration and a collaborative process. These two parts concern the departure and arrival points of the knowledge-based system that we intend to create. This system is dedicated to the specification of collaborative processes from our knowledge about collaboration. The first part concerns the definition and characteristics of collaboration. This part provides the knowledge for executing our system. The second part concerns the target collaborative process model that the system has to produce and the approach for modelling collaborative processes.

1. Approach for inter-enterprise collaboration

Enterprises are dynamic systems operating in an environment where markets frequently change. They are faced with the need to reconstruct themselves in order to survive in a competitive global market. A single enterprise model is not adequate to cover all the aspects of customer and market satisfaction due to the complexity of the markets. Therefore enterprises are trying to create collaborative networks with other enterprises. They are developing networked enterprises contributing toward a common business goal. [D1.1 Synergy, 2008]

This section consists of three subsections. Firstly, the collaboration between multi-enterprises will be discussed in relation to the different definitions and classifications in terms of types and levels of collaboration. Secondly, collaboration will be specifically characterized through several criteria. Thirdly, we will address enterprise knowledge, the different types of knowledge in an enterprise, and the knowledge that drives and supports collaborations.

1.1. Inter-enterprise collaboration

1.1.1. Definitions

The literature provides a number of definitions about collaboration in general, as well as specific to different domains.

In a generic context, according to [Jacobs, 2002], a collaboration needs some shared goals, language, and experiences in common, as well as a shared environment. Actors carry out activities in order to achieve the shared goals. The collaboration is composed of three aspects: the joint achievement of tasks, coordination of distributed components involved in the collaboration, and a social component including trust and expertise notions. This definition points out the notions of common goals, and task achievement that partners have to accomplish together.

[Pollard, 2002] defined that collaboration is like tasks that require decisions across the boundary of the enterprise, and the technology in order to attain a mutual competitive

advantage. This definition limits the collaboration to the exchange of information and includes collective decision-making.

[Lambert et al., 1999] defined collaboration as a value network. A network that collaboratively creates value implies a particular type of relation between organizations allowing them to share the risks and the benefits, and to reach a higher level of performance than when the organizations work alone.

Another definition that highlights the notion of process has been given by [Kak et al., 2002]. Collaboration is a process in which many independents coordinate their strategies and decisions.

Finally, [ICC, 1999] said that collaboration is broadly defined as the interaction between two or more individuals and can encompass a variety of behaviours, including communication, information sharing, coordination, cooperation, problem solving, and negotiation. This definition points out the different kinds and levels of collaboration between at least two individuals. We will talk about the different levels of collaboration in the next section.

From the above definitions, an inter-enterprise collaboration has these following characteristics:

- Group of at least two enterprises
- Interactions and relationships among enterprises which include the following elements:
  - Common goals that are shared or decided collectively
  - Collaborative decision-making
  - Process of activities that involved partners carry out in order to achieve the common goals and create value.
  - Exchanges or flows of information between activities

We can summarize that an inter-enterprise collaboration has human and organizational aspects. The human aspect concerns actors who work together in order to accomplish tasks. The organizational aspect is related to an organizational structure and processes, as well as common strategies. However, collaboration is normally supported by dedicated technologies. Thus, the technological aspect also has to be included in the collaboration context.

1.1.2. Different types and levels of collaboration

Nowadays enterprises are moving from a closed enterprise to open networked enterprises in order to survive in globally developing and competitive markets. Collaborations are established among enterprises for specific purposes and aspects. In the definition of [ICC, 1999] mentioned previously the different levels of collaboration are defined in the same way as [Bouzguenda, 2006] [Camarinha-Matos et al., 2006] and [Touzi, 2007]. However, collaboration can be classified into four levels where each level extends its preceding level:

- Communication (data exchange): the enterprises communicate, exchange, and share information in order to optimize their individual functionalities. This level concerns a

simple exchange of data between enterprises. The enterprises use basic communication interfaces like the Internet, and EDI (Electronic Data Integration) that authorize an asynchronous communication. This level of collaboration is called networking by [Camarinha-Matos et al., 2006].

- Coordination (sharing and synchronization of tasks): the enterprises carry out tasks depending on their partners. This level concerns the sharing of competences (applications, functions, and services) that the enterprises share or make available to their potential partners.

- Cooperation (pursuing common goal): with respect to game theory, cooperation is a broader perspective which includes individual and common goals [Pingaud, 2003]. Partners have their own objectives and strategy, but they all pursue a common goal. This level requires a collaborative process that the partners have to follow in order to achieve the common goal. Collaboration at this level is federated and can be supported by the EAI (Enterprise Application Integration) or SOA.

- Integration: the exchange of data, sharing tasks and pursuing common goals are inherent to the fact that the enterprises reunite (virtually) as one entity. Interoperability can be seen as a way of achieving this integration. All the barriers to interoperability are removed. This level is equivalent to the collaborative network level defined by [Camarinha-Matos et al., 2006].

Due to the economic globalization and complexity of the market, [Browne et al., 1999] distinguished two categories of enterprise collaboration called extended enterprise and virtual enterprise. The descriptions of these two categories are summarized as follows:

- Extended enterprise: the enterprises focus on their core competencies and outsource all others. An extended enterprise is a kind of collaboration in which an enterprise extends its boundaries to encompass its suppliers, customers, and other business partners [D1.1 Synergy, 2008]. [Ratchev et al., 2000] stated that the extended enterprise can only be successful if all of the component groups and individuals have the information they need in order to do business effectively. The enterprise develops long term relationship with their partners. This kind of collaboration is generally implemented by manufacturing industries collaborating with other firms for financial services, transportation, and distribution.

- Virtual enterprise: the enterprises form a temporary alliance to share skills, core competencies, and resources in order to better respond to business opportunities and particular goals [Luczak et al., 2005]. The partners are integrated using information and communication technology (ICT) such as online services, the Internet, etc. They have no central office, no hierarchy, and no vertical integration [D1.1 Synergy, 2008].

[Browne et al., 1999] summarized the differences of the characteristics between these two categories in Table II. 1:

S. No.	Criteria	Extended enterprise	Virtual enterprise
1	Strategic issues	Long term objective	Relatively short term
2	Partnership purpose	Business co-operation	Project Collaboration
3	Organization stability	Stable collaboration of firms	Dynamic organization of companies, each with their core competencies
4	Relationships	Trust and mutual dependence	Temporary and dynamic
5	Coordinator	Generally the manufacturer or any other strong body	Managed by broker
6	Information technology (IT)	Facilitated and enabled by IT	Enabler for the cooperation
7	Organization type	Dependent on coordinator	Legally independent bodies cooperating for a particular mission

Table II. 1 Characteristics between extended enterprise and virtual enterprise [Browne et al., 1999]

In comparison with the four levels of collaboration discussed previously, the extended enterprise and the virtual enterprise can be considered as collaboration in the integration and the cooperation levels respectively. The extended enterprise is a long term collaboration in which an enterprise extends its boundaries and maintains stable interoperability with its partners. This can be seen as an integration of enterprises into a single system. The virtual enterprise is created in order to respond to a specific goal in a short period of time. This is the same as the cooperation where the enterprises follow a collaborative process in order to achieve a goal.

1.2. Studies of collaboration characterization

Collaborations self-develop in terms of strategy, functionality, duration, and objectives. However, a study of the systems' characteristics is required. Collaboration has its specific characteristics which can be qualified through its levels, and supporting technologies. Remember the intention of our work is to provide an appropriate MIS that connects to the different information systems of the partners in order to overcome the interoperability problems. According to this scope, we only take into account the characterization based on the different levels of collaboration. The technological aspect is out of our scope.

Establishing collaboration leads to the setting up of a collaborative network. Collaborative networks change in time and have limitations due to the heterogeneities of partners. The definition given by [Camarinha-Matos et al., 2006] refers to a collaborative network as being an alliance constituted of a variety of entities (e.g. organizations and people) that are largely autonomous, geographically distributed, and heterogeneous in terms of their operating environment, culture, social capital and goals, but that collaborate to better achieve common or compatible goals, and whose interactions are supported by a computer network. Most

forms of collaborative networks imply some kind of organization over the activities, identifying roles for the participants and some governance rules.

A network can be defined differently as proposed by [Poulin et al., 1994] who described it in terms of the constituting components. A network is a set of nodes and links between these nodes. The following explains the components of a network:

- Node can be an individual, a tool, a service, a department, an enterprise, or even a group of enterprises. Each node has its own characteristics.
- Links determine the way in which the different nodes are connected and interact between them. These are multiple forms of partnership. Each link can be seen as a flow vector.
- Relations between partners of a network define the scope in which they interact by defining their common goals, type of partnership, and governance rules.
- Flow can be defined as the movement of materials (e.g. information, product, control flows, etc.) between nodes.

The above definitions, the levels of collaboration, as well as the table of [Browne et al., 1999] bring us to the configuration of collaborative networks. Furthermore, we are also interested in how to define and set up collaborative networks, the different relationships established in networks, as well as the structure of networks. The following summarizes these different studies.

1.2.1. Network configuration factors

Different forms of network develop in terms of organization, principles of functionality, duration, and objectives. [Zaidat, 2005] proposed to characterize the networks of organizations by using configuration factors. The definition given by [Zaidat, 2005] refers to a network of organizations or to a set of organizations that exchange products or services through collaboration or cooperation in order to achieve a common objective. He adopted the enterprise modelling approach to define the networks of organizations. This definition matches our point of view in terms of a collaborative network.

Different configuration factors have been distinguished from a number of theoretical studies in this area, such as network of inter-firms [Grandori et al., 1995], network and its function [STRATEGOR, 1997], etc. Here we introduce some configuration factors that we will adopt in our work:

- Common goal: A common goal describes the reason why the network is established in terms of products or services to deliver to customers.

- Relationship: Relationship describes the establishment of strategy between partners. This will be discussed in detail in Section 1.2.2.

- Partner: A partner can be an enterprise, organization, or individual person. It is described by the following attributes: competencies, capacity, culture, motivation

(learning, transferring of knowledge, improving competitiveness, etc.), objectives, localization, and roles in the network.

- Organizational structure: A network can have a high-level structure, or a functional structure. The high-level structure can be described by a topology which will be detailed in Section 1.2.3. The functional structure is represented by a dependency or flow of resources between activities, which will be presented in Section 1.2.4.

- Duration: A network can have a short or long life and this can be predefined or not.

- Stability: A network is static when the same partners form the network throughout its life cycle. It is dynamic when some partners can join or leave the network whenever they want.

According to our perspectives, the above terms are the main characterization criteria necessary for describing and analysing collaborative networks. The common goal and the relationship concern the strategy of the network. The partner and the organization concern the structure of the network. Finally, the duration and the stability concern the behaviour of the network. These characteristics allow us to understand the collaboration and the networks' goals and objectives.

The relationship and the organizational structure are the prominent characteristics and concern the network modelling. They will be described in detail in the three following sections.

1.2.2. Inter-enterprise relationships

The inter-enterprise relationships describe the strategy of the enterprises. They allow us to determine the roles that the partners play in the collaboration. Many classifications of relationship have been proposed.

[Frayret et al., 2003] distinguish six possible relationships of inter-enterprise collaboration, including relationships between an enterprise and its customers, suppliers, competitors, service providers, complementary enterprises, as well as universities.

This classification has a strong correlation with the one proposed by [Fombrun et al., 1982]. [Fombrun et al., 1982] who suggested only three groups which cover the six of [Frayret et al., 2003]. These three groups are:

- Competition or horizontal relationship concerning the collaboration between enterprises in the same business or industry. A network that has this type of relationship is in the situation of substitutability in terms of offers. The partners are currently competing for similar resources, or producing similar products in order to increase negotiation power. The horizontal relationship relies on the strategic management domain.

- Subcontracting, supplier-customer, or vertical relationships concern the collaboration between an enterprise and its suppliers. The enterprises are related to each other by an

essential interdependence. This kind of relationship is mostly found in the manufacturing industry where the enterprises are situated in different hierarchies of the production chain.

- Group interests or transversal relationships concern enterprises which are neither substitutable nor essentially interdependent, but add reciprocal value. The partners provide services that would be a benefit to each other. The partners establish their relationship in order to achieve the same interests, such as shared technology development.

[Gueguen et al., 2006] highlighted the different forms of interdependence between enterprises originally proposed by [Fombrun et al., 1983]: commensalism and symbiotic. Commensalism is about the interdependence between similar entities (concurrence), while the symbiotic is about the interdependence between dissimilar entities (non concurrence). These two forms are equivalent to the three types of relationships listed above. Table II. 2 shows the correspondence between the three types of relationships, the forms of interdependence, and the characteristics of each relationship:

Direction of the relationship	Forms of interdependence Astley and Fombrun (1983)	Situation in terms of offer	Point of analysis	Typeical situation
Horizontal	Commensalism	Substituability	Sector	Competition
Vertical	Symbiotic	Essential complementarity	Subsidiary	Subcontracting
Transversal	Symbiotic	Annexed additivity	Environment	Interest group

Table II. 2 Forms of relationships and their characteristics [Gueguen et al., 2006]

According to [Gueguen et al., 2006], the concept of commensalism concerns cooperative relationships between rivals. It can be considered as a form of horizontal relationship which concerns competition between companies in a situation of substitutability. The concept of symbiosis concerns the vertical and transversal relationships.

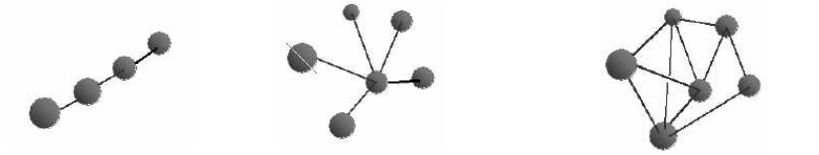
1.2.3. Topologies of network

The concept of topology can be used to describe the structure of networks. It also includes the duration, stability, and decision-making aspects. Several proposals to characterize networks in terms of topologies have been suggested.

[Burn et al., 1999] distinguished six different models of network: virtual faces, co-alliance, star-alliance, value-alliance, market-alliance models, and virtual brokers. These network models range from organizations providing services in the web which do not control any user of the service to dynamic networks of entities collaborating to meet market opportunities.

Another proposal by [Katzy et al., 2000] that we focus on here suggested the characterization of networks be based on three topologies: chain, star, and peer-to-peer. It describes the main

coordination structure that governs the flow of information and material, as well as the decision making within the network. Some of the characteristics of these three topologies are briefly highlighted below:



Chain topology Star topology Peer-to-peer topology

Fig.II. 1 Three basic types of network topologies [Katzy et al., 2000]

- A chain or process oriented topology may be defined as a coordinated system of organizations, people, processes, and resources that moves information or services from one end (e.g. producer, supplier) to another end (e.g. consumer or customer). Chain topology consists of several partners who are collaborating in order to achieve a specific goal. The partners are connected in the right order based on their hierarchy defined by the direction of resource transfer. The architecture of the chain network is mostly fixed and long term. This kind of network is adopted mostly in manufacturing, such as production, and distribution chains. It is designed to deal with for example, inter-organizational coordination and control, rapid change of logistics quantities.

- A star topology or hub-and-spoke network topology consists of one central partner managing the entire network. All other members are directly related to the central partner. This type of topology corresponds to an extended enterprise in which each member provides key functionalities, and a distinguished member who plays the role of leader. The star topology is a good solution for large scale enterprise, for instance consortia in the construction industry, large automobile manufacturing industry. The central partner holds the decision-making power and coordinates the tasks by different methods. It can direct and change the entire network based on its strategies, competencies, and political power. In the automobile industry, which has developed tier supplier networks, the structure of this network is a stable hierarchy of suppliers that are led by one original equipment manufacturer (OEM) [Laubacher et al., 1997]. However the centralized management structure may cause bottleneck problems and failure.

- A peer-to-peer topology is project oriented. It entails mutual relationships between all partners. It is characterized by the lack of hierarchy where any peer may interact directly with any other peer. Their management is usually based on self-organization. The management competencies are distributed within the members and the decision-making power is equal for every member. Such networks seem to be appropriate in industries where access to knowledge and expertise is of primary concern. The biotechnology industry, academic community, and film industry in Hollywood are examples of this kind of network. However, establishing such networks requires careful selection of members, developing and enforcing strong codes of behaviour, as well as investing in building trust amongst each other.

We summarize the trend of characteristics of these three topologies as shown in the Table II. 3:

Topologies	Decision-making power	Duration	Stability
Chain	Hierarchic (chain of command)	Continuous (long term)	Static
Star	Central (one dominant actor)	Continuous (long term)	Static
Peer-to-peer	Equal (no dominant actor)	Discontinuous (short term)	Dynamic

Table II. 3 Main characteristics of topologies

1.2.4. Dependencies and coordination mechanism

The concept of dependency concerns the organizational and functional views of collaboration. It describes interactions within a collaborative network. Dependencies can be referred to as the flows of resources that are transferred between the activities of the enterprises.

A number of past studies have described dependencies and coordination mechanisms only in general terms, without characterising in detail the different kinds of dependencies, the problems that dependencies create, or what coordination mechanisms can deal with those problems. [Crowston, 2003] proposed a more complete theory of coordination and dependency. He distinguishes the different dependencies, and provides some examples of appropriate coordination mechanisms for each kind of dependency.

Dependency and coordination have been recurrent topics in organization studies. These two topics are related because coordination is seen as a response to problems caused by dependencies. According to [Malone et al., 1994], coordination is defined as managing dependencies. They analysed group action in terms of actors performing interdependent activities to achieve goals. These activities might also require or create resources.

[Crowston, 2003] classified three basic kinds of dependencies: flow, sharing, and fit. These three types of dependencies arize from resources that are related to multiple activities. Fig.II.2 shows these three types:

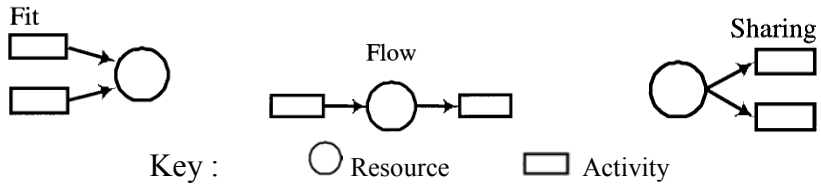


Fig.II. 2 Three types of dependencies [Crowston, 2003]

- Flow dependencies arise whenever one activity produces a resource that is used by another activity. This kind of dependency can be referred to as a supplier-customer relationship. It can occur all the time in almost all processes.

- Sharing dependencies occur whenever multiple activities use the same resource. For example, when two activities need to be done by the same person, or use the same

machine. In addition, the nature (shareability and reusability) of the resource has an impact on this kind of dependency.

- *Shareablity* describes how many activities can use the resource at the same time. Most resources (e.g. raw material, tools, or effort) are non-shareable. An actor may be assigned to multiple activities but work on only one at any instant. Resources are shareable if they are not changed by the activities.
- *Reusability* describes how many activities can use the resource over time. Some resources, such as tools or information, can be used and reused, but others, such as raw materials, can only be used once.

- Fit dependencies arise when multiple activities collectively produce a single resource. For example, when several operators produce the same parts (e.g. motor, body, transmission, etc.) of a car.

Table II.4 shows examples of dependencies between activities and alternative coordination mechanisms for managing them.

Dependency	Examples of coordination mechanisms for managing dependency
Flow	
Prerequisite ('right time')	Make to order vs. make to inventory ('pull' vs. 'push'). Place orders using 'economic order quantity', 'just in time' (kanban system), or detailed advanced planning.
Accessibility ('right place')	Ship by various transportation modes or make at point of use
Usability ('right thing')	Use standards or ask individual users (e.g., by having customer agree to purchase and/or by using participatory design)
Sharing	'First come–first serve', priority order, budgets, managerial decision, marketlike bidding
Fit	Boeing's total simulation vs. Microsoft's daily build

Table II. 4 Examples of elementary dependencies between activities and alternative coordination mechanisms for managing them [Crowston, 2003]

Coordination mechanisms are something we put in between two activities to handle any resource being transferred between them. For example, the flow dependency between the activities delivering and receiving of products can be coordinated by the mechanism of shipping by air. Fig.II.3 illustrates this example of flow dependency:

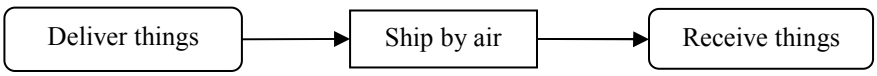


Fig.II. 3 Example of using coordination mechanism to deal with flow dependency

We can see that the dependency approach is process-oriented. It is a business process design approach proposed by the MIT Process Handbook that will be discussed in Section 2.3.2. So, we will talk about this approach again in the collaborative process section (Section 2).

1.2.5. Conclusion

In this section, we aim to define the principal criteria for characterizing any collaboration. Collaboration leads to the setting up of a collaborative network which is generally composed of nodes (partners), links, relations, and flows. Several factors for configuring collaborative network have been defined and adapted from the enterprise modelling approach, such as partners, common goals describing the expectation in terms of result of network, duration, stability, relationships between partners, and organizational structure. Some of these factors have been studied in detail for instance, relationship and organizational structure. The organizational structure concerns the topology and dependency concepts. The topology describes how partners connect to each other through a relationship, while the dependency represents the resource exchanges between services. We realize that the dependency concept is process-oriented as it concerns the characteristics of a process rather than a collaborative network. We therefore decided to take such a concept into account in the collaborative process.

Finally we summarize the prominent elements required for characterising a collaborative network as follows: common goal, relationship, partner (role and competence), topology (duration and decision-making power).

1.3. Enterprise knowledge

In this section, we will discuss knowledge, its definition and types in a generic context, as well as in a collaboration context. Before going into detail, we have to clarify the difference between data, information, and knowledge due to the confusion of these three related terms.

[Kabilan, 2007] described the differences between data, information, and knowledge. Data is raw. It exists and has no significance beyond its existence (in and of itself). It can exist in any form, usable or not. Information is data that has been given meaning by way of a relational connection. It exists when the relationships between data are recognized within a specific context. It can be useful, but is not necessarily so. Knowledge is the appropriate collection of information. It describes what actions to take when certain information exists. It has to be useful.

It follows that knowledge is the most valuable asset of any enterprise. It is a very important resource for learning new things, solving problems, creating core competencies, and initiating new situations for both individuals and enterprises now and in the future [Liao, 2003]. Knowledge describes how and why things are done within an enterprise. It also concerns the production of new facts or new knowledge.

Coming back to the relation between data, information, and knowledge, according to [Aamodt et al., 1995], we can summarize that the role of knowledge is to play the active part in the process of:

- transforming data into information (data interpretation)
- deriving new information from an existing one (elaboration)
- acquiring new knowledge (learning)

Knowledge technologies, according to [Milton, 2008], have emerged during the last two decades in order to deal with knowledge in an enterprise. They are computer-based techniques and tools that provide a richer and more intelligent use of information technology. They are associated with a number of subject areas:

- Knowledge Engineering: emerged from the work in Artificial Intelligence. It concerns the building of computer systems that solve problems in the way humans do.
- Knowledge Based Engineering: emerged from the world of Computer Aided Design (CAD). It concerns the building of computer systems that help engineers to work more efficiently.
- Knowledge Management: emerged from a number of business initiatives. It concerns the use of techniques and tools to make better use of assets in an enterprise. This subject area involves the identification and analysis of required knowledge assets and knowledge asset-related processes, subsequent planning, etc. [D1.1 Synergy, 2008].

1.3.1. Types of enterprise knowledge

Enterprise knowledge can be classified as follows [Spender 1993] [Nonaka & Takeuchi, 1995] [D1.1 Synergy, 2008]:

- Explicit knowledge is both formalized and abstract. It is easily expressed, transferred, and shared in the form of data, facts, figures, rules, or formulas. This kind of knowledge can be transmitted between individuals in formal and systematic ways. The more explicit the knowledge is, the more stable it is. For example, textbooks, software code, etc. Generally explicit knowledge is associated with data through business processes. It can be implemented in an enterprise through creating, reading, updating, and deleting operations.

- Tacit knowledge is often referred to as knowledge-in-practice. It is highly personal knowledge developed from direct experience. It is subjective, and not easily expressible. It can be shared through interactive conversation. Examples of this kind of knowledge are experience, idea, emotions, intuitions, and insights; which are the foundation of innovation and creativity. Tacit knowledge has two dimensions: technical and cognitive. The technical dimension is described as know-how which is dependent on experience. The cognitive dimension is composed of schema, values, and beliefs.

- Social knowledge is shared and may be either explicit or tacit. For example, scientific knowledge which is shared and explicit, communal knowledge which is shared and tacit, etc. Individual knowledge is always tacit.

1.3.2. Knowledge for collaboration

This section focuses on knowledge that can drive and support collaboration. This knowledge encompasses:

- Enterprise information

- Shared and public information of the enterprise network
- Information about how to find, access, and retrieve the above information
- Expertise and procedures to effectively apply the above.

The above knowledge is available within enterprises, as well as in networks to enhance and promote productivity in collaboration. Knowledge users are potentially contributing to the content of knowledge on collaboration. Some knowledge may be freely shared within enterprises or some networks. Some may be commercially valuable knowledge like product information.

[Li et al., 2006] classified enterprise knowledge relevant to the formation and operation of collaborative ventures into four categories: enterprise core competence, VO[3] (Virtual Organization) formation knowledge, partner selection knowledge, and VO operations management knowledge. Fig.II.4 summarizes the different categories of knowledge for collaboration together with their possible sources:

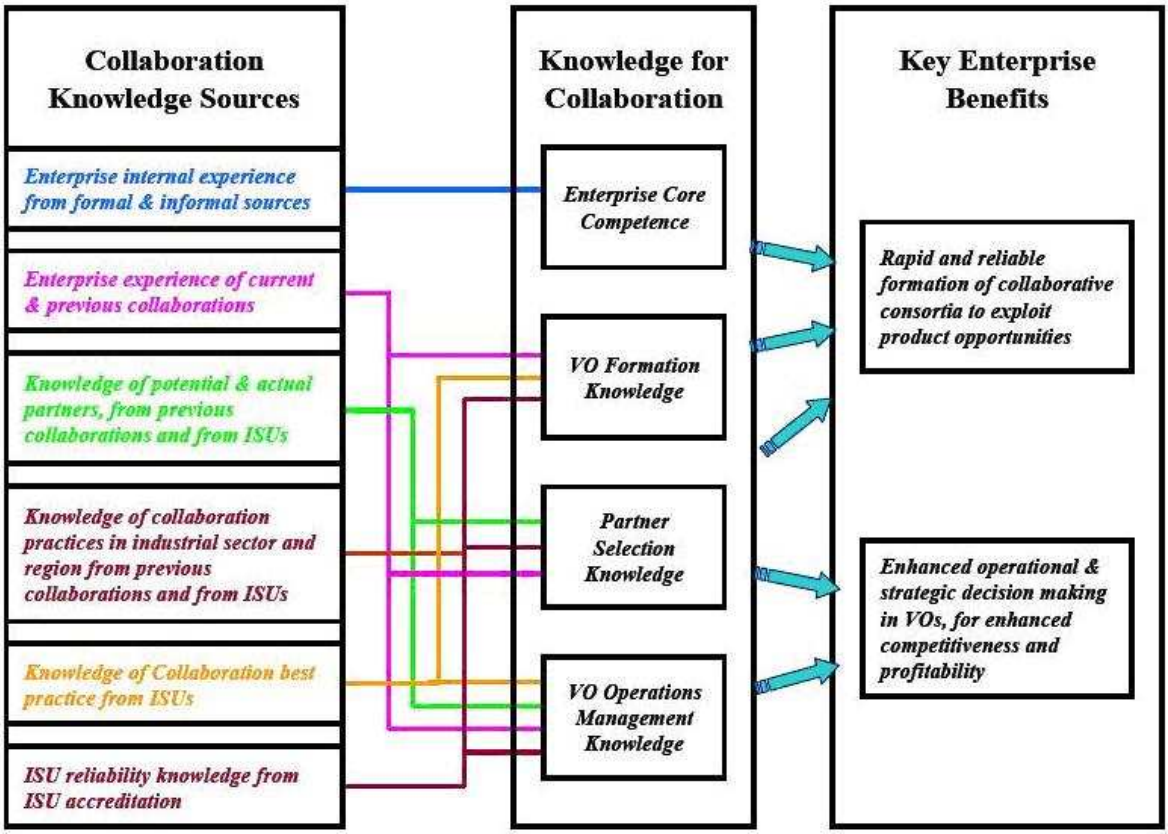


Fig.II. 4 Enterprise benefits of knowledge oriented collaboration [InterOP Roadmap, 2006]

- Enterprise core competence: this is knowledge of the enterprise's own capabilities and capacities, strengths and weaknesses, and technical IPR. This kind of knowledge

[3] A VO is a short-term association with a specific goal of being active in fulfilling a Business Opportunity (BO). VO is similar to virtual enterprise (VE) and is a special type of collaborative network. [Camarinha-Matos et al., 2005]

concerns the enterprise's internal experience which can come from formal and informal sources. We might see a fractal view of the enterprise as a collaboration of its internal functions.

- Process knowledge for VO formation: this is knowledge of best practice in formation of a VO, critical factors in VO development, legal issues, risk analysis, and application of tools such as maturity gate planning. It also includes moderation knowledge about collaboration and interoperability issues likely to be critical to partners. The sources of this kind of knowledge are enterprise experience of current and previous collaboration, as well as knowledge of collaboration practices in the industrial sector from previous collaborations and from ISUs[4] (Interoperable Service Utility).

- Process knowledge for partner selection: this is knowledge of potential partners' core competencies, collaboration and interoperability capability, and reliability in collaboration. The knowledge in this category can be retrieved from knowledge of potential and actual partners from previous collaborations and from ISUs, as well as from the current and previous enterprise experience.

- VO operations management knowledge: this includes the VO enterprise model to support decision making, knowledge of interoperability issues within the VO applied to ensure communication, and moderation knowledge about operational factors likely to be critical to partners. This kind of knowledge can be retrieved from various sources for example, enterprise experience, and best practices from ISUs.

1.4. Conclusion

The aim of Section 1 is to discuss inter-enterprise collaboration and enterprise knowledge in both the generic and the collaboration context.

A number of definitions concerning collaboration have been highlighted which allow us to summarize that collaboration has both human and organizational aspects. The human aspect concerns the actors who accomplish the collaboration tasks. The organizational aspect concerns the strategies, goals and relationships as well as the processes. Collaboration has four levels: communication for exchanging data, coordination for sharing and synchronization of tasks, cooperation if a common goal and process have been established, and integration if the enterprises have become a single entity.

Collaboration leads to setting up a collaborative network which can be configured by several elements (partners, common goals, relationships, and topology including duration and decision-making power). These elements are related to the definition of collaboration. They are the main criteria for characterising collaborations.

Enterprise knowledge can be acquired from experience, practice, conversation, innovation, document, software code, etc. The knowledge that drives and supports collaboration is for

[4] The ISU provides interoperability as a technical, commoditized functionality, delivered as services. The ISU is a basic infrastructure that supports information exchange between diverse knowledge sources, software applications, and Web Services [Li et al., 2006].

example, core competences of the enterprise, experiences from previous collaborations, knowledge of interoperability issues, decision-making support, etc. Normally, the knowledge required for collaboration is retrieved from experiences, and best practices.

The precision of collaboration characterization depends on the knowledge we can retrieve from partners. The capture of more knowledge and better quality knowledge leads to a more accurate characterization of the collaboration and the result will be closer to reality.

2. Approach for collaborative process modelling

According to the definitions of inter-enterprise collaboration discussed in the previous section, collaboration has several characteristics, such as relationships among enterprises, common goals to be achieved collaboratively, shared activities, and processes. [D.A.2.1 Athena, 2006] proposed to define cross-organizational business processes (CBP) for ensuring the interoperability at the process level of the AIF. This aspect refers to multi-actors, interactions, and exchanges of resources between actors.

In this section, we start by introducing the Service-Oriented Architecture (SOA). Then, we focus on defining what a collaborative process is in general and specifying our collaborative process. Next, the languages for process representation are discussed. Finally, we present some modelling approaches which can be adopted for collaborative process modelling.

2.1. SOA

Traditionally, according to [Durvasula et al., 2006a], IT works with the business owners, who are influenced by application vendors. IT often ends up deploying multiple systems that perform the same tasks within an enterprise or business unit. Redundant infrastructure solutions for authentication, single sign-on, and applications (packaged and custom), such as sales, quoting, and order management compound the complexity and cost for IT. It becomes nearly impossible and definitely impractical to modify this portfolio to reflect a change in a business process or accommodate an acquisition.

Service Oriented Architecture (SOA) was introduced as a means of facilitating inter-organizational computing. Its goal is to achieve loose coupling among interacting systems in contrast to traditional tightly coupled systems and monolithic architectures [Touzi, 2008]. SOA provides the flexibility to treat elements of business processes and the underlying IT infrastructure as services that can be reused and combined to address changing business priorities. It encapsulates the functionalities of systems around business processes and packages as interoperable stand-alone services. These services may be executed on geographically distributed computers.

The service producer publishes the service to the service registry which is leveraged by the service consumer for runtime binding. The registry also acts as the record system for the business policies that it enforces at runtime. The service consumer asks a service registry for the service that matches its criteria. If there is such a service in the registry, it gives the

consumer a contract and an endpoint address for the service. The figure below summarizes the relations between service consumer, producer, and registry:

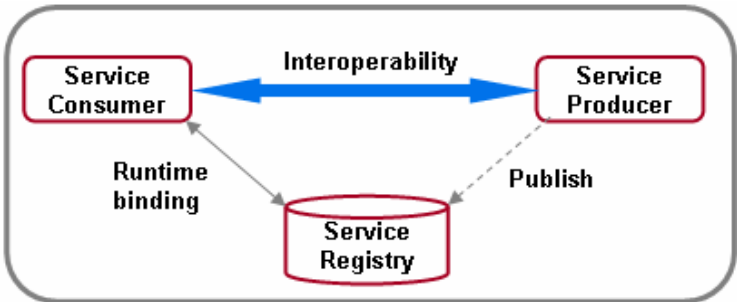


Fig.II. 5 Relations between service consumer, producer, and registry [Durvasula et al., 2006b]

2.2. Collaborative process

2.2.1. Definitions and characteristics of collaborative process

Before going into the definition of collaborative process, we have to understand what a process is and classify its main characteristics. A process in this section is referred to as an internal process in an organization. This aspect is really generic when we talk about a process. A number of definitions of process have been proposed in the literature. Here, we highlight some of them.

According to [Vernadat, 1999], process is defined as a set of activities executed in order to achieve at least one objective. The ISO 9001 (2000) [ISO 9001, 2000] defined a process as a set of activities that transform input into output. [Morley, 2002] followed the previous definitions and adds the organizational aspect regarding actors who carry out the activities.

[Touzi, 2007] defined the main characteristics of a process as follows:

- Activities: an activity describes the transformation of input into output.
- Graph of activities represents the sequence of activities essential for achieving the objective.
- Roles describe the organization or responsibility of actors in the process.
- Objective represents the intention or expectation in terms of output or achievement of process.
- Transition controls could represent the decision of things in the process.
- Resources can be means, information, or tools used by activity.

From the above definitions and characteristics of process, we can focus now on a process that is shared between enterprises or entities, a so-called collaborative process. A cross-organizational business process (CBP) is a special kind of collaborative process.

Many definitions have been proposed in the literature. [Morley et al., 2005] pointed out that the aspect of multi-organizations is essential in a collaborative process because the partners have their specific competencies, so they provide the activities they can perform in order to achieve the objective of process. For example, a process where the customer manages the

activities in the domain of buying, while the supplier manages the activities related to the domain of selling.

[Touzi, 2007] considered the activities provided by partners as their internal process. But, he also stated, based on the point of view of the enterprise, that a collaborative activity can be seen as an internal process and presents an interface dedicated to the collaboration. In the following section, we will introduce the meta-model of collaborative process defined by Touzi.

We can extract some interesting characteristics of a collaborative process from the previous definitions as follows:

- Taking place between multiple independent entities (enterprises, organizations, or individuals).
- Common objectives to be achieved.
- Implying governance between the involved entities.
- Different entities providing a specific competency and playing a specific role.
- Independent entities exchanging resources and collaboratively performing their activities to pursue the objective of the process.

We realize that these characteristics combine those of process and inter-enterprise collaboration. We can see the coherence between them in terms of common goals, roles of partners, exchanges of resources between activities (dependencies), different competencies constituting relationships between partners, and so on.

2.2.2. Meta-model of collaborative process [Touzi, 2007]

From the above definition of a collaborative process given by [Touzi, 2007], in the SOA context, activities are considered as services that partners expose. The work presented in this dissertation is based on this definition. In this section, we introduce the meta-model that describes this definition of a collaborative process.

According to [Touzi, 2007], the model of a collaborative process is BPMN-oriented and based on the SOA. Its meta-model has been defined by referencing the BPMN specification as well as our collaboration aspect. Why BPMN rather than other languages? We will talk about this point in the next section. A set of constraints, precisions, and restrictions has also been defined, for example:

- A pool called *CIS* has to show in the collaborative process model for representing the MIS.
- Each participant has its own pool which contains only tasks (services).
- Types of tasks in the participant's pool can be only *receive*, *invoke* (send), or *response* (two-way). Due to the "SOA by mediation" approach, partners are seen as service consumer or service producer.
- Direct communications between participants are not allowed, but have to pass through the CIS pool according to the "SOA by mediation" approach.
- Each message flow corresponds to at least a data transfer between tasks.

The meta-model of a collaborative process describes the modelling elements that are necessary for constituting a collaborative process model. It also takes into account the constraints listed above. Fig.II.6 shows the meta-model composed of these following elements:

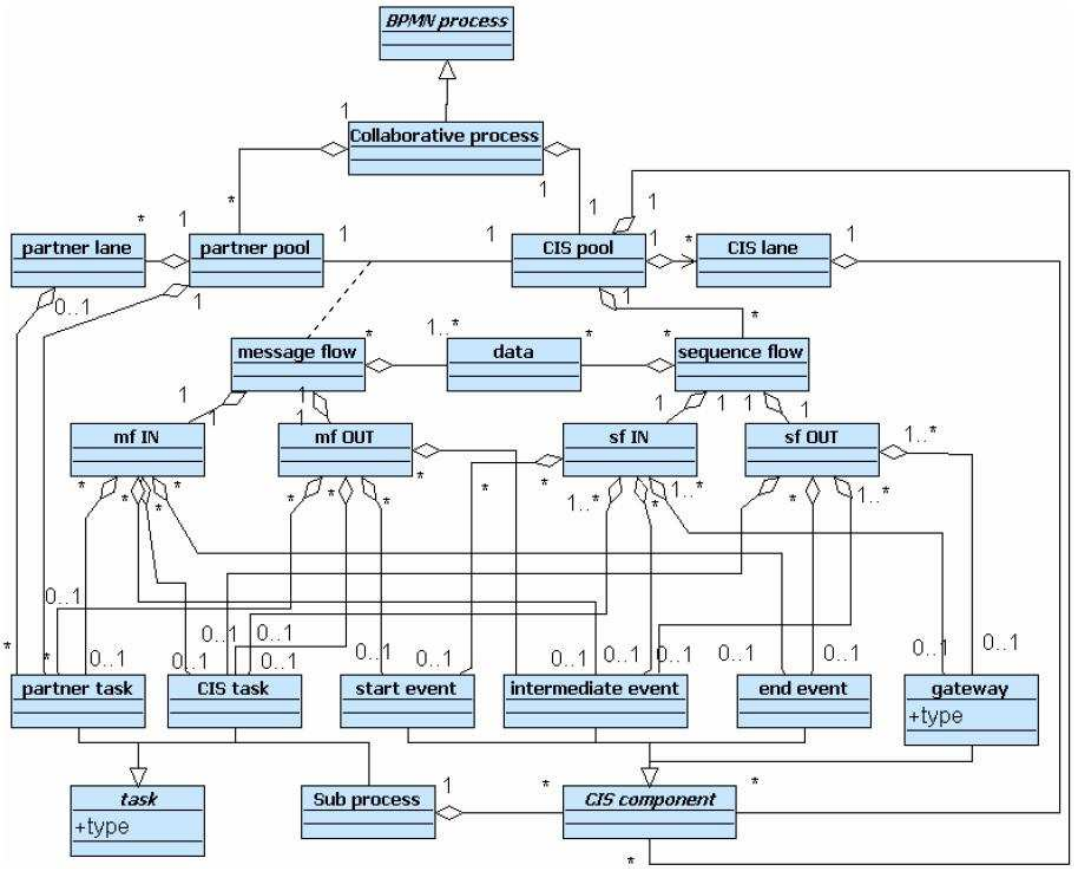


Fig.II. 6 Meta-model of collaborative process [Touzi, 2007]

- *BPMN process* class: an abstract class representing a model oriented BPMN grammar
- *Collaborative process* class inheriting from the abstract *BPMN process* class
- *Partner pool* class representing a participant
- *CIS pool* class representing the MIS, collaborative platform that manages the exchange of data, and applications.
- *Partner lane* class representing a role (or subdivision) of a participant
- *CIS lane* class representing a subdivision of the MIS
- *Message flow* class representing a dependency of resource (or a data transfer) between a *Partner pool* and the *CIS pool* classes
- *Sequence flow* class representing a dependency between elements of the *CIS pool*
- *Partner task* class representing a service of a participant
- *CIS task* class representing a service of the MIS, namely MIS service
- *Start/ intermediate/ end events, gateway, sub-process* classes representing BPMN modelling elements
- *CIS component* class: an abstract class representing elements in the *CIS pool*
- *mf IN* and *mf OUT* classes representing the extremities of a *Message flow*
- *sf IN* and *sf OUT* classes representing the extremities of a *Sequence flow*

2.2.3. BPMN as a collaborative process modelling language

In this section, we talk about BPMN and why we chose BPMN for modelling a collaborative process.

There are many languages for business process modelling, for example, flow chart, Petri net, IDEF0, PCD (Process Chain Diagram) of ARIS, activity diagram of UML (Unified Modelling Language), and BPMN (Business Process Management Notation). The first three languages are elementary languages that take into account only a behavioural representation based on the facts. Process modelling is therefore reduced to just a sequence of activities. The last three languages are advanced languages that include informational (data) and organizational (actors) representations. This category can provide a rich typology of activities, events, flows, etc.

[Touzi, 2007] pointed out that using the advanced formalisms to model a process can cover several aspects of processes including actors (organizational view), and information (informational view). Besides, such formalisms offer a functional view of a process which is also an important factor. The main difference between PCD, UML, and BPMN is that BPMN has a direct connection with an execution language, BPEL. Therefore, BPMN seems to be the best choice for us for modelling a business process. Now, let's see what BPMN really is.

BPMN is a semi-formal language for process modelling defined by the BPMI[5] (Business Process management Initiative) [BPMI, 2004]. BPMI has been established in order to promote and develop the use of Business Process Management (BPM) through the use of standards for process design, deployment, execution, maintenance, and optimization of processes.

The goal of BPMN is to provide an explicit notation that is easy to use, and understandable by all business people who create, implement, or monitor processes. Thus, BPMN closes the gap between process design and process implementation. Another design goal of BPMN is its compatibility with XML-based workflow languages like BPEL. The visualization of processes designed with these formalisms is quite important. Indeed, BPMN has not only been designed to be a modelling formalism for capitalising knowledge, but also a bridge for automated executing processes (under BPEL) via BPMS (Business Process Management System) engine.

The intention of BPMN is to standardize the business process design notation. Since business process design is a complex domain, BPMN covers a variety of different modelling techniques and allows the creation of end-to-end business processes [Vasko et al., 2006]. The elements of BPMN allow the viewers to easily understand the process diagram. The basic elements are classified basing on the three different viewpoints as follows:

[5] www.bpmi.org

- Organizational view is represented by *pool*, and *lane*. A pool is a participant in a process and acts as a graphical container for partitioning a set of activities from other pools. A lane is a sub-partition within a pool. It is generally used to organize activities.

- Functional view is represented by nodes. The nodes are the *activities* as rounded-corner rectangles, *events* as circles, and *gateways* as diamond-shapes. The activities are tasks performed in the business process. When modelling more complex process flows, we need to model more complex business events, such as messages, timers, business rules, and error conditions. These events can initiate a process flow, happen during a process flow, or end a process flow. A gateway is used to control the divergence and convergence of Sequence Flow. It will determine traditional decisions, as well as the forking, merging, and joining of paths.

- Informational view is represented by arcs. The arcs are the flows of information. There are three types: *sequence flow* for internal (in the same pool) communications, *message flow* for inter-pool communications, and *association* for associating data, text, and other Artefacts with flow objects, or showing the inputs, and outputs of an activity. Sequence flow, message flow, and association are drawn as solid lines, dashed lines, and dotted lines. They correspond to the informational view.

[Owen et al., 2003] concluded that by taking processes and placing them in pools or lanes, we are specifying *who* does *what*, for events you specify *where* they occur, and for gateways we specify *where decisions* are made, or *who* makes them. An example of process written in BPMN:

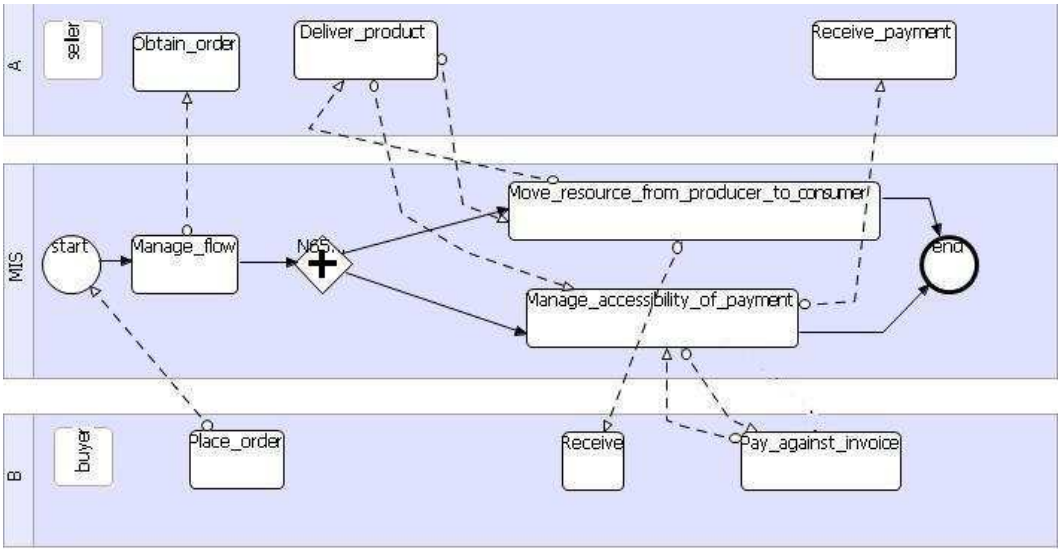


Fig.II. 7 Example of BPMN process

The above diagram conforms to the meta-model of a collaborative process shown in the previous section (Fig.II.6). In this example, there are two partners (A and B) and a MIS pool. Each partner exposes its own services that are necessary for the collaboration. There is no flow inside the partner pools. The services of the MIS are collaborative services that manage

and invoke services exposed by the partners. Thus, each partner is seen as a set of services that interface with the collaborative platform.

2.3. Modelling approaches

2.3.1. Strategic alignment

During the last few years numerous researches have focused on aligning the information system (IS) to the strategy of the enterprise [Henderson et al., 1992] [Etien, 2006] [Gmati et al., 2007]. According to [Gmati et al., 2007], in order to keep competitive, it seems very important that enterprises align their IS on their organizational processes, goals, and strategies. Once the enterprises develop their processes, and strategies in order to adapt themselves to their environment, their IS should change too. They believe that an enterprise will function efficiently if its IS and the objective to be achieved are consistent.

The model of strategic alignment which was originally proposed by [Henderson et al., 1992] is as follows:

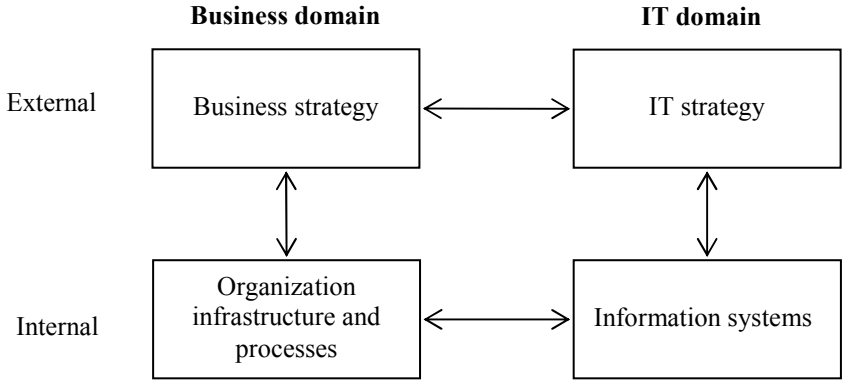


Fig.II. 8 Strategic alignment model [Henderson et al., 1992]

According to the above model, there are two main domains involved in the alignment: the business and the IT domains. Both are split into two sub-domains through the external and the internal perspectives. Thus, the framework is composed of these following elements:

- Business strategy concerns business scope, business competencies, and business governance.
- Organizational infrastructure and processes is composed of administrative infrastructure, skills, and business processes.
- IT strategy concerns technology scope, systemic competencies, and IT governance.
- IS infrastructure is composed of IS architecture, IS skills, and IS processes.

There are four main types of alignments:

- Process/IS alignment concerns the understanding and exploitation of the relations between the organizational processes and the systems to be constituted.

- Strategy/Process alignment concerns the specification of the relations between the objectives of the enterprise and the implemented processes.

- Business strategy/IT strategy alignment represents the connection between the business strategy and the IT (Information Technology) strategy. This is needed to answer how the technology capacity can support the business strategy.

- IT/IS alignment explains how to use the IT strategy to make the system agile.

This approach allows improving the efficiency of the enterprise by dealing with both business and IT domains. The main idea is to define an IT solution in response to the business strategies. The strategies of these two domains should be as coherent as possible. The alignment allows the IT actors to ensure that the solution they propose meets the objective of the enterprise as well as the business actors.

2.3.2. MIT Process Handbook

The Process Handbook (PH) is the result of more than 10 years of development by over 40 researchers and practitioners of the MIT Centre for Coordination Science (CCS). The PH is founded on a set of key concepts that support an innovative methodology for redesigning business processes.

The primary goal of PH is to be a comprehensive framework for organising large amounts of useful knowledge about business in a richly interconnected, consistent, and powerful way [Malone et al., 1999, 2003]. The PH is a large repository of business processes and available to the public over the web at http://ccs.mit.edu/ph. It can be used to help people in redesigning existing business processes, inventing new processes especially those that take advantage of information technology, and organising as well as sharing knowledge about organizational practices. It is also expected to be useful in (semi) automatically generating software to support or analyse business processes.

As the PH is a repository that organizes large amounts of useful knowledge about business, such knowledge came from use cases, alternative business models, textbook models, etc. It is categorized under different concepts. The concepts including in the PH's contents are specialization enabling the creation of a type, subtype hierarchies of business processes, dependencies for capturing the resource relationships between processes, and coordination mechanisms which are processes that manage at least one dependency. Fig.II.5 summarizes the contents of the MIT PH as of July 2002.

At a research level the PH has been shown, as discussed in [Alessandro et al., 2007], to be useful in a variety of domains, such as business process reengineering, business process automation, software design, etc.

In this section, we discuss adopting the PH for modelling business process.

Type of entry	Number of entries	Example entries
Activities		
Generic business activity models		
MIT Business Activity Model	381	Buy, Make, Sell
MIT Business Models Archetypes	30	Produce as a Creator, Produce as a Broker
Comprehensive business process models developed elsewhere	689	International Benchmarking Clearinghouse's Process Classification Framework
Coordination processes	300	Manage by market with bidding
Subtotal	1400	
Case examples		
Supply chain	100	Balance supply chain resources with requirements {Honda}
Hiring	50	Select human resources using agent software {Humana}
e-Business examples	420	Distribute books via electronic store {Amazon}
Subtotal	570	
Classification structure		
Generic verbs and other activity categories	3252	Create, Modify, Preserve, Destroy, . . . , Develop, Make product, Provide service
Total activities	**5232**	
Other kinds of entries		
Dependencies	73	Flow of information
Resources	163	Human agent, software agent, location
Conceptual frameworks for specific research projects		
Exceptions	260	Agent unavailable, resource shortfall
Systems dynamics elements	200	Goal-gap molecule, backlog molecule
Total nonactivity entries	**696**	
Total entries	**5928**	

Table II. 5 Contents of the PH [Malone et al., 2003]

The major kind of content in the PH is the generic models of business activities. These generic models represent important activities (services) that occur in lots of businesses. They can be used in a number of ways [Malone et al., 2003]:

- As a framework for organising and grouping many other kinds of business knowledge: case examples, best practices, software tools, or contact information for knowledge experts.
- Providing a useful starting point for modelling the specific details of a particular enterprise, process, or software module.
- Stimulating new ideas about what is possible that might not have occurred.

The PH includes four primary kinds of generic models of business activities: the MIT business activity model, the MIT business model archetypes, a collection of comprehensive business process models developed elsewhere, and models of basic coordination processes. In

this section we only present the MIT business activity model and models of coordination processes. Both are the keys for modelling business processes.

2.3.2.1. MIT business activity model

The MIT business activity model (BAM) is one of the most important models. It represents a top or abstract level of business activity. Abstract activities are for example, buy, make, sell, etc. Such an activity describes the competence of its provider. Each abstract activity has parts that represent the actions to be performed at a functional level. For example, *buy* includes parts like *identify own needs*, *identify potential sources*, and *select supplier*. The following figure shows an example of the relation between an abstract activity and its corresponding functional level activities:

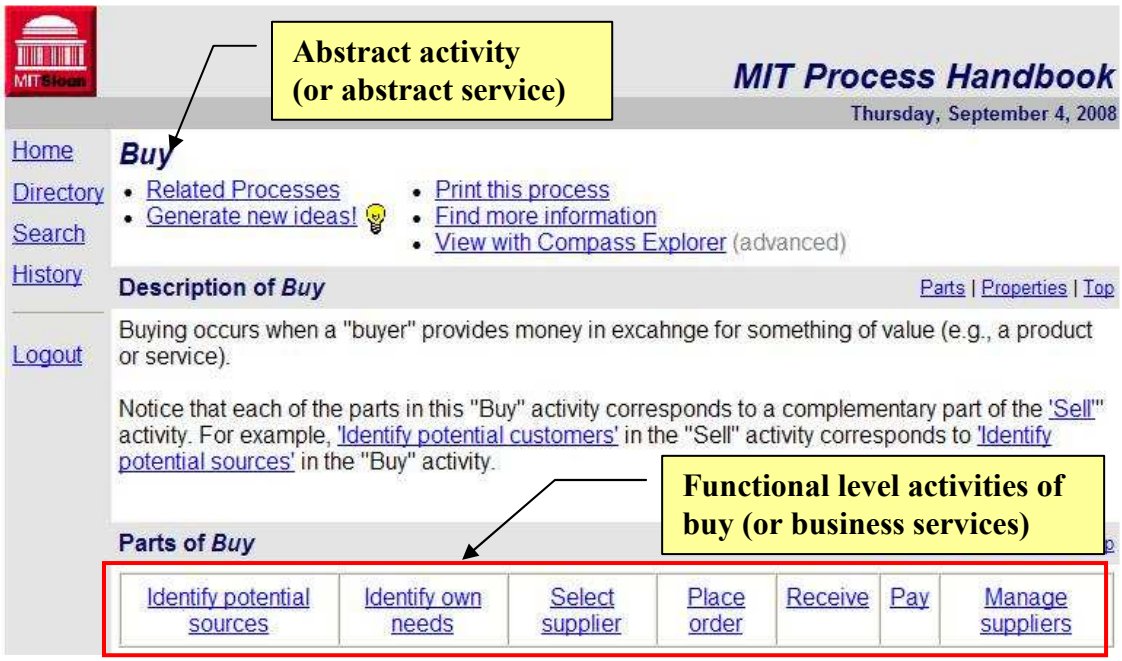


Fig.II. 9 Example of buy process with its parts
(copied from the Process Handbook online [http://process.mit.edu/Activity.asp?ID=1348])

Based on the SOA approach, we consider the abstract activities as abstract services and the functional level activities as business services.

The BAM is based on a theoretical analysis of business from the perspective of coordination theory. This perspective is related to the dependency and coordination mechanisms that have been presented previously in Section 1.2.4. The use of dependencies and their coordination mechanisms describes the interactions of resources between services.

2.3.2.2. Models of coordination processes

The models of coordination processes are the key mechanism of business process modelling. The most generalized coordination process is called the *manage dependency* activity. The first three specializations of this activity are: manage flow, manage sharing, and manage fit; which

are all the three basic types of dependencies presented in Section 1.2.4. An example of flow dependency has also been shown in the dependency section (Fig.II.3).

In some cases, these generic coordination mechanisms include deeper specializations that describe specific examples, such as manage by market, manage time by market bidding, etc. Based on the SOA approach, we consider this kind of process as coordination services. Such services are collaborative services provided by the collaborative platform.

From the earlier applications as well as the developing goal of the PH itself, we realized that the PH is focused generally on specifying process within organizations but we are interested in applying the PH in an inter-organizational context to specify collaborative processes which are shared between different enterprises.

In a context of collaboration and the meta-model of collaborative process presented previously, the BAM allows us to define the services (abstract services and business services) exposed by partners through its abstract and functional level activities. The abstract service describes the competence of the partner. The flows of resources between business services are described by the dependencies which are managed by the coordination processes. The coordination process (coordination service) is the key mechanism for us to succeed in designing a collaborative process. Such a process is seen as collaborative services (or MIS services) implemented on the MIS platform to deal with the interoperability of partners' services and data exchanges between services. Thus, these coordination services invoke or request the business services of partners by sending a message. The coordination service depends on messages to link with a business.

2.3.3. Summary of the modelling approaches

We aim at studying the modelling approaches in order to adapt them to define our own approach in a collaboration context.

The work described in this dissertation only concerns the business level of the MIS design approach. Thus, we are interested in the approaches for modelling processes. Even though the strategic alignment approach does not directly concern process modelling, this approach requires a process to connect the business and IT domains. Our work particularly addresses the strategy/process alignment.

The MIT Process Handbook is a repository of business processes. It contains about 5000 business processes including use cases, alternative business models, and so on. These instances of knowledge are about business processes that we can use and reuse to constitute our knowledge base. The PH can also be considered as a means of modelling and improving business processes in organizations. We adopted the concepts of BAM and coordination process, including dependency, to define our mechanism for collaborative process modelling. The concept of BAM provides the two levels of activity that we call abstract activity and functional level activity. The abstract activity (abstract service) is a high level activity describing the competence of a provider. The functional level activity (business service) is an action that a provider has to perform. The dependency concept describes the relations between business services and resources. When two business services have a common resource, a

dependency exists. The coordination process deals with the dependency by managing that common resource. We consider now that the dependency is a characteristic of collaborative process rather than a characteristic of the collaborative network because it describes the transactions and interactions of services between partners. It provides the functional and organizational views of the collaborative process.

2.4. Collaborative process modelling in a "SOA by mediation" approach

In Section 2, we have presented the approaches for dealing with the interactions, exchanges of information between services, and multi-enterprises. These all reflect the notion of a collaborative process.

Collaborative process in this context refers to a process shared between independent parties in order to achieve objectives. The principal characteristics of a collaborative process have also been highlighted, such as multiple partners, objectives to be achieved, competencies and roles of each partner, activities that all partners have to carry out collaboratively, and so on.

The meta-model of a collaborative process has been stated in this section in order to describe how we define and see a collaborative process. We realized that our collaborative process is described at organizational and informational levels. We are very interested in services, flows of resources between services (dependency), and MIS services. Thus, to represent such processes, we use BPMN which can cover both organizational and informational views. Furthermore, the meta-model of a collaborative process is really BPMN-oriented, in which we can find the BPMN elements (e.g. event, gateway, pool, and lane) defined.

After that the modelling approaches have been studied in order to represent collaborative processes formally. We have focused on strategic alignment, and the MIT Process Handbook. All of them provide different aspects for our collaborative process modelling perspectives. The strategic alignment describes the necessity of strategy in business and IT domains. The MIT PH provides the essential knowledge (instances) and modelling mechanism (dependency and coordination).

In summary, the essential elements for defining a collaborative process are: partner's service, resource, flow of resources between services (dependency), and MIS service (coordination service of the MIT PH).

3. Conclusion of the chapter

In this chapter, we presented two parts which concern the important approaches for defining a knowledge-based system dedicated to the specification of collaborative processes. These two parts provide the elements for constituting the departure and arrival points of this system.

Firstly, the approaches for inter-enterprise collaboration including the definitions, classification, and characteristics have been addressed. Establishing collaboration means setting up a collaborative network of multi-enterprises. We have discussed the main criteria for characterising and configuring a collaborative network, which are partners (including their

competences and roles), common goals to be achieved, relationships, topology defined through duration and decision-making power of the network. The precision of collaboration characterization depends on the knowledge we can retrieve from the partners. Most knowledge can be retrieved from previous collaboration experiences, and best practices. The larger the quantity of knowledge captured the more accurate the characterization of the collaboration will be.

Secondly, we addressed the approach for modeling collaborative processes. A collaborative process in this context is referred to as a process shared between independent parties in order to achieve objectives. Our collaborative processes are defined on the basis of the meta-model of collaborative process given by [Touzi, 2007]. The language we will use to represent our collaborative processes is BPMN because it covers both organizational and information views of process. Moreover, it was created specifically for business process modelling. We used the knowledge provided in the MIT Process Handbook and we also adopted its modelling mechanisms to complete our collaborative process design. Based on these studies, we summarized the principal elements for defining a collaborative process as being: partner's service, resource, flow of resources between services (dependency), and MIS service (coordination service of the MIT PH).

The schema below shows the mapping between the elements of collaborative network and collaborative process via the modelling mechanisms of the Process Handbook:

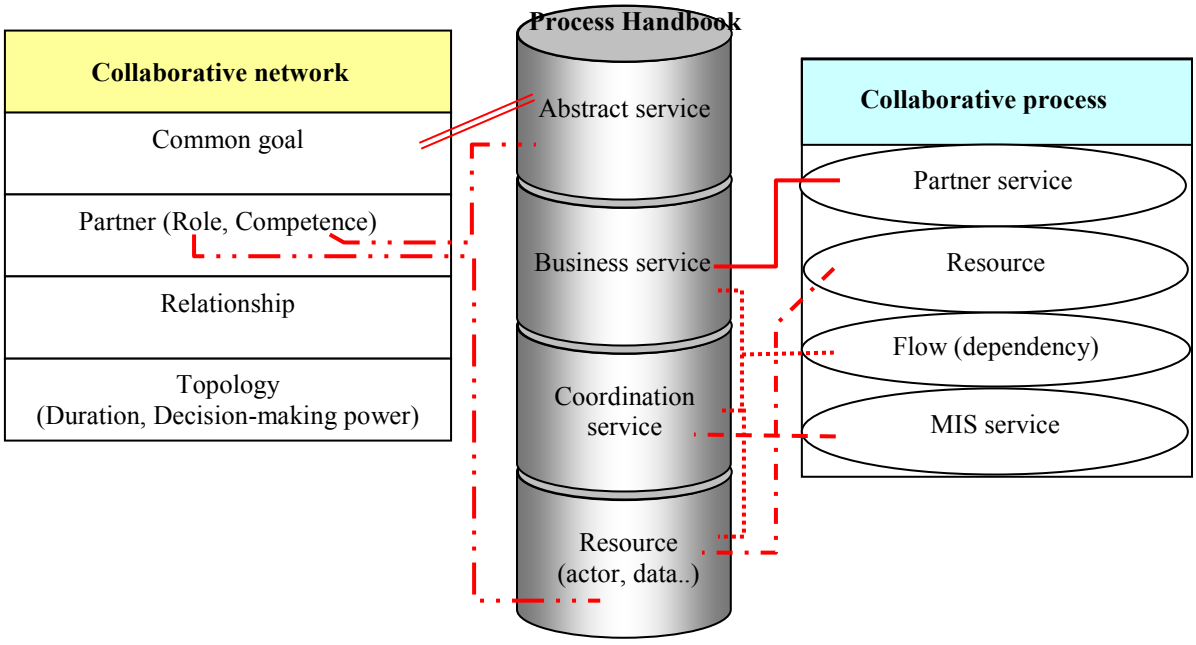


Fig.II. 10 Mapping between the collaborative network and collaborative process worlds via the Process Handbook

From the above schema, we use the knowledge about the common goal and the partners in the collaborative network world to define the elements in the collaborative process world. The knowledge about partners concerns not only the information about the partners themselves,

but also their competences and roles. We realize that this knowledge provides essential information for mapping with abstract service (competence) and resource (role) of the PH (dash-dot-dot lines). We also take into account the strategy/process alignment by mapping the common goal (defined collectively by the partners) with the abstract services in order to find the competences needed in the collaboration (double full line). We inspired from the BAM concept of the PH (Section 2.3.2.1) that every abstract service has its corresponding business services. These business services match with the partner services of the collaborative process. The coordination service, business service and resource of the Process Handbook are all required together to determine the flow of resources between business services (dotted line). This is because a coordination service is seen as a response to a problem caused by resource dependency between services. The coordination service itself matches with the MIS service of collaborative process (dashed line). Finally, the resource of the PH matches with the resource of collaborative process (dash-dot line).

Now we have some idea of how we can cross the worlds of collaborative network and collaborative process. We also know that we need the knowledge from the Process Handbook to make this idea complete. But, how do we make this idea concrete?

The challenge in the following chapters is to adopt the elements studied in this chapter to develop our knowledge-based system. The first part of this Chapter 2 provides the input knowledge for executing the system, while the second part provides the target collaborative process model that the system has to reach as well as the essential knowledge from the Process Handbook to complete the design process. However, as we discussed at the end of Chapter 1, to create our knowledge-based system, we also need to address the knowledge representation and reasoning issue. This issue will deal with the correspondences of elements between the collaborative network and the process worlds. This issue makes up the core of the system. So, we will answer the above question in Chapter 3.

Chapter 3. Knowledge-based System for Collaborative Process Definition

We intend to develop a knowledge-based system in order to support the design of a collaborative process from the knowledge about collaboration. This system allows us to accomplish the CIM level and remove the organizational barrier of interoperability, both of which are within the scope of our work. The system relies on the process-oriented design concept dealing with implicit knowledge acquisition, knowledge representation and reasoning, and collaborative process modelling.

We explored in Chapter 2 the concepts of collaboration and collaborative process which concern the departure and arrival points of our system. From the mapping between these two concepts (Fig.II.10), we realized that to move from collaboration to collaborative process domains, we need some additional knowledge coming from the MIT Process Handbook. The main objective of this chapter is to deal with this mapping by applying the knowledge representation and reasoning issue and the reuse of knowledge.

In this chapter, we first introduce an overview of our knowledge-based system. Then, we shall focus on the approach for representing and reasoning knowledge which is the core of our system. This approach is based on ontology and rule. We will explore the notion of ontology including its definitions and uses. We are also interested in the different languages for editing ontology. Several ontologies related to the enterprise and network modelling domains will be presented. After that, the Collaborative Network Ontology (CNO) that we have developed will be introduced.

1. Presentation of the approach

In Artificial Intelligence, according to [Grimm et al., 2007], knowledge-based systems have a computational model of some domain of interest where symbols serve as replacements for real world domain artefacts. The domain of interest can cover any aspect of which we desire to represent knowledge for computational purposes.

Knowledge representation and reasoning is a fundamental aspect in the construction of knowledge-based systems. It aims to design computer systems that reason about a machine-interpretable representation of the world, in a similar way to human reasoning. Reasoning refers to inferring new statements (conclusions) from a set of given ones (assumptions) which have the property that they are true whenever the assumptions are true [Keller et al., 2005]. In knowledge-based systems, there are three categories of knowledge [Godoy, 2005]:

- Domain knowledge, describing the main information and knowledge for a particular domain of interest. This includes the definition of concepts, their properties and relationships, as well as instances of the concept defined. This category of knowledge captures the static knowledge of the system.

- Inference knowledge, concerning the behaviour of the reasoning process. This is composed of inferences which are primitive reasoning steps operating over the knowledge base by inference engines. This category of knowledge defines the dynamic knowledge of the system.

- Task knowledge, referring to an abstraction over the inference knowledge to promote the reuse of knowledge and a knowledge-based system. It defines the goal to be achieved and the method to be applied. This category of knowledge defines also the dynamic knowledge of the system.

Many languages and formalisms with different levels of expressivity have been proposed for representing knowledge. Some are better suited to represent the static knowledge, and others better used for defining the problem-solving processes. Based on the Semantic Web[6] technologies and use cases, [Grimm et al., 2007] distinguished three forms of representing knowledge:

- A semantic network is a graph whose nodes represent concepts and whose arcs represent relations between these concepts. It is suitable for capturing the taxonomic structure of categories for domain objects and for expressing general statements about the domain of interest.

- Rules reflect the notion of consequence and define reasoning steps. They are represented in the form of *If-Then* expressions. This kind of representation operates on facts and is suitable for reasoning about concrete instance data.

- Logic can formalize both the semantic network and rules in order to give them a precise semantics. The most fundamental logical formalism is first-order logic. The graph of semantic networks can be formalized through description logic which is fragments of first-order logic.

[Grimm et al., 2007] pointed out that, in information systems, ontology is a conceptual model of things in a particular domain, brought into machine-interpretable form by means of knowledge representation techniques. In comparison to the three forms of knowledge representation listed above, ontology appears as the most appropriate formalism for representing domain knowledge applications. It supports reuse of knowledge, and the knowledge base. However, ontology lacks the expressivity for problem-solving. Rules can deal better with the problem-solving and dynamic behaviours of a knowledge-based system.

Hence, a knowledge-based system maintains a knowledge base which stores the symbols of the computational model in the form of statements about the domain, and it performs reasoning by manipulating these symbols. Ontology may be used to declare the structure of a

[6] The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF). [http://www.w3.org/2001/sw/]

knowledge base, whereas a knowledge base is constituted with ontology, instances, and rules [Kabilan, 2007].

Our knowledge-based system is created on the basis of the discussion above. We use ontologies and rules for dealing with knowledge and for the purpose of defining collaborative processes. Fig.III. 1 depicts our knowledge-based system:

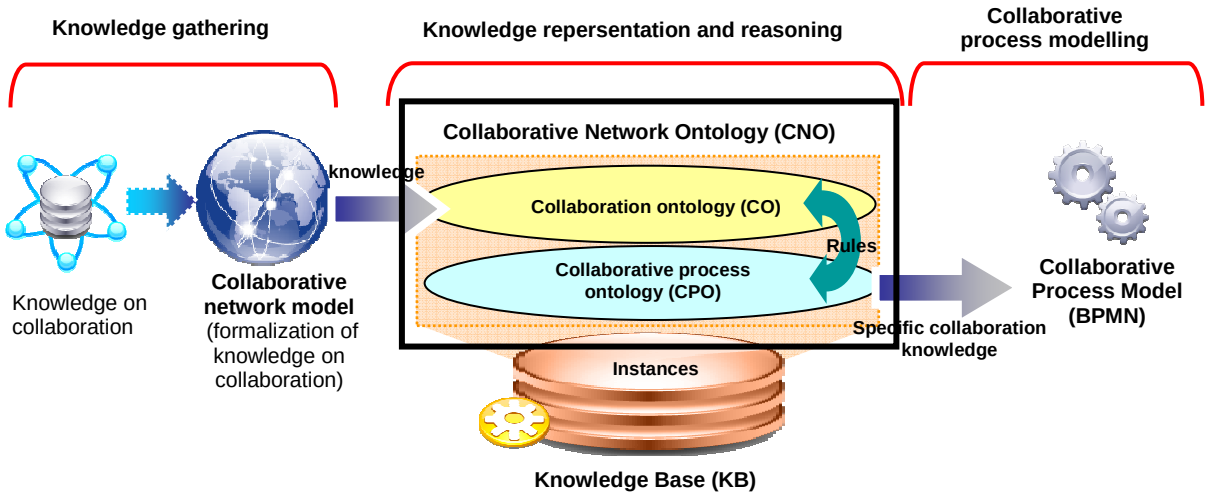


Fig.III. 1 Knowledge-based system for generating BPMN collaborative process

According to the above figure, our knowledge-based system is composed mainly of three parts: knowledge gathering (left), knowledge representation and reasoning (middle), and collaborative process modelling (right):

- The left part concerns the acquisition and formalization of implicit knowledge. We assume that business partners are able to express informally and partially their knowledge on collaboration based on their experiences and perspectives. This implicit knowledge is about the characteristics of the collaboration that is to take place. The knowledge is interpreted and represented formally in the form of a collaborative network model. Based on the mapping schema (Fig.II.10) at the end of Chapter 2, this part defines the collaborative network world.

- The middle part concerns the knowledge representation and reasoning on a knowledge base. This knowledge base is built on an ontology covering the collaborative network and process domains. This ontology is called Collaborative Network Ontology (CNO). According to the mapping schema (Fig.II.10), this part is in charge of creating the connections between the collaborative network and process worlds.

- The right part concerns the collaborative process modelling. This part includes also the extraction of knowledge from the knowledge base. It also guarantees the conformity of the collaborative process model generated for the logic level of the MDE (Fig.I.3). Based on the mapping schema (Fig.II.10), this part defines the collaborative process world.

To be able to understand how our knowledge-based system works, a detailed and theoretical explanation of the middle part is needed. Thus, in this chapter, we focus on this middle part which concerns knowledge representation and reasoning. This is the core part of our system. This part is an ontology-based approach using ontologies and deduction rules for dealing with static and dynamic behaviours of knowledge. Such an approach will be explained in theory in this chapter. The whole system will be explained technically in Chapter 4 including the left and right parts.

Our knowledge representation and reasoning approach starts by receiving the knowledge on collaboration from the left part. This knowledge is captured from the partners involved in a given network. Once the knowledge is collected, this knowledge is stored in the Knowledge Base (KB). But how can we reuse this knowledge for modelling collaborative process? We realize that the collected knowledge concerns the collaboration domain, not the collaborative process. This question came up at the end of Chapter 2 on the mapping between collaborative network and collaborative process domains (Fig.II.10). So we need to morph this collaboration knowledge into collaborative process knowledge. We have developed the CNO ontology consisting of two ontologies which each deals with a specific domain: Collaboration ontology (CO) for collaboration features, and Collaborative process ontology (CPO) for collaborative process features. These two ontologies are connected to each other by the semantics and structural links (represented by the deduction rules). The rules offer the possibility to morph collaboration knowledge into collaborative process knowledge by reasoning with the instance data (from the Process Handbook) in the KB. Thus, they should cover the connections between collaborative network and process domains defined in Fig.II.10. They are also created as a part of the CNO. Finally, the knowledge about collaborative process specific to the given network will be extracted, and will be transformed into an appropriate collaborative process.

To be able to develop the CNO, we have to explore the concepts and technologies regarding ontology and the related ontologies in the enterprise and network modelling domains which are our primary domains of interest.

2. Concepts and technologies for developing ontology

This section is aims to present the ontology as a means for representing knowledge. First of all, we would like to introduce the origin and history of the word ontology and Ontological Engineering. According to [Gomez-Perez et al., 2004], ontology first appeared in ancient Greece where it was applied to extracting the essence of things. In the middle ages, one of the key issues in ontology was universalality. The counterparts of universals in knowledge modelling are classes or concepts in contrast with individuals. At the end of the middle ages, ontology was much more concerned with how to codify characteristics of things using symbols. Until now, a large number of ontologies have been developed by different groups, using different approaches, and with different methods and techniques. Ontological Engineering originates in the context of the new science that codifies features of things, and on the other hand, ontologists are devoted to extracting the essence of things. The more the essence of things is captured, the easier it is for the ontology to be shared.

In this section, we first present the definitions of ontology. Then, we talk about the applications and categories of ontology. Next, we deal with the representation of ontology in terms of languages. Finally, we present some ontologies related to our domains of interest.

2.1. Definitions of ontology

In its original meaning in philosophy, ontology is a branch of metaphysics and denotes the philosophical investigation of existence [Grimm et al., 2007]. It means a systematic explanation of being. It studies being or existence and its basic categories and relationships, to determine what entities and what types of entities exist. In the last decade, ontology has become a relevant word for the Knowledge Engineering community. [Guarino et al., 1994] propose to use the words "Ontology" (with capital 'o') and "ontology" (uncapitalized) to refer to the philosophical and Knowledge Engineering senses respectively.

Many definitions of an ontology have been given over recent years. The first definition was stated in [Neches et al., 1991]: an ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms, and relations to define extensions to the vocabulary.

From this definition, an ontology includes not only the terms that are explicitly defined in it, but also the knowledge that can be inferred from it.

A few years later, [Gruber, 1993] gave another definition of an ontology: a formal explicit specification of a shared conceptualisation for a domain of interest.

This definition became the most quoted by the ontology community. It encompasses several interesting aspects [Gruber, 1995] [D8.1 InterOp, 2004] [Grimm et al., 2007] which are:

- A formal aspect: an ontology is expressed in a knowledge representation language that provides formal semantics. The ontology should be machine understandable.

- An explicit aspect: the type of concepts used and the constraints on their use are explicitly defined. Ontology states knowledge explicitly to make it accessible for machines. Notions that are not explicitly included in the ontology are not part of the machine-interpretable conceptualisation it captures, although humans take them for granted using common sense.

- A sharing aspect: since ontology captures consensual knowledge and reflects an agreement on a domain conceptualization, it cannot be created by a unique person. This means there is some kind of agreement among people in a community or systems regarding the ontology.

- A conceptualisation aspect: an ontology specifies knowledge in a conceptual way in terms of symbols representing concepts and their relations. On the other hand, it describes a conceptualization in general terms in order to cover as many situations that can potentially occur as possible.

- A domain specificity: the specifications in an ontology are limited to knowledge about a particular domain of interest. The narrower the scope of the domain for the ontology, the more an ontology engineer can focus on axiomatizing the details in the domain rather than covering a broad range of related topics.

[Missikoff et al., 2003] summarized that an ontology is a formal and explicit description of concepts of a particular domain, together with characteristics of these concepts and relations between them. Ontology is referred to as a representation of knowledge that can be used and reused in order to facilitate the comprehension of concepts and relations as well as the communication between different domain actors.

2.2. Uses of ontologies

In recent years, ontologies have been widely used in Knowledge Engineering, Artificial Intelligence, and Computer Science, in applications related to knowledge management, intelligent integration information, database design, and in the Semantic Web.

The idea of the Semantic Web is to annotate web content by machine-interpretable meta-data so that computers are able to process this content on a semantic level. [Grimm et al., 2007] discussed the application of ontologies in this context. Ontologies offer a semantic approach to electronic information management and exchange. They provide the domain vocabulary in terms of which semantic annotation is formulated. Meta statements about web content in such annotations refer to a commonly used domain model by including the concepts, relations, and instances of a domain ontology. The formality of ontology languages makes it possible to reason about semantic annotation from different sources, connected to background knowledge in the domain of interest.

In Artificial Intelligence research some typical types of applications have evolved that make use of ontologies in different ways. The followings are examples listed by [Grimm et al., 2007]:

- Information integration: ontologies are often applied to handling heterogeneous information sources on the schematic level. Different databases store the same kind of information conforming to different data models. Ontologies are therefore used to mediate between database schemas for integrating information from different sources and to interpret data from one source under the schema of another.

- Information retrieval: information retrieval on web documents is a major field of application for ontologies. The idea behind ontology-based information retrieval is to increase the precision of retrieval results by taking into account the semantic information contained in queries and documents and by lifting keywords to ontological concepts and relations.

- Semantically enhanced content management: the data that is actually computed is annotated with meta-data. Ontologies provide the domain-specific vocabulary for

annotating data with meta-data. The formality of ontology languages allows for an automated processing of this meta-data and facilitates machine-interpretability.

- Knowledge management and community portals: in enterprises, knowledge management refers to individual knowledge to be shared and systematically maintained. Ontologies provide means to unify knowledge management efforts under a shared conceptual domain model, connecting technical systems for navigating, storing, searching, and exchanging community knowledge.

- Expert systems: it is desirable to simulate a domain expert who can be asked questions specific to the domains of interest. This can be achieved by developing a domain ontology for formalising the expert knowledge. The domain-specific questions can then be answered by reasoning over this highly specialised knowledge.

We can summarize the general reasons of building ontologies as follows:

- Sharing common understanding of the structure of information among people or software agents
- Enabling reuse of domain knowledge
- Making explicit domain assumptions: explicit specifications of domain knowledge are useful for new users who must learn what the terms in the domain mean.
- Separating the domain knowledge from the operational knowledge
- Analyzing domain knowledge: formal analysis of terms is extremely valuable both when attempting to reuse existing ontologies and when extending them [McGuinness et al. 2000].

2.3. Different categories of ontologies

Ontologies are considered as a means to foster reuse of knowledge within knowledge-based system engineering, and it turns out that different types of ontologies exhibit a different potential for reuse [Grimm et al., 2007]. A categorization of ontologies can be determined according to their subject of conceptualization, as well as to the richness of their internal structure. Much recent research in this area [Gomez-Perez et al., 2004] [Grimm et al., 2007] has provided a state-of-the art categorization of ontologies'. Here we briefly introduce some of the more interesting categorizations:

[van Heijst et al., 1996] classified ontologies according to two principle dimensions:

- The first dimension has three categories: terminological (lexicon), information (database schemata), and knowledge modelling ontologies.

- The second dimension is identified through four categories: representation, generic, domain and application ontologies. The representation ontology concerns conceptualization of knowledge representation formalism. The generic ontology is reusable across domains. The domain ontology tries to cover all the aspects of one domain and is reusable. Finally, the application ontology is a non-reusable ontology

because its knowledge can be limited to the minimum required to fulfil the needs for the application.

Another interesting categorization proposed by [Guarino, 1998], which classified types of ontologies according to their level of dependence on a particular task or point of view. This classification gives the most prominent insights, which are summarized in Fig.III. 2:

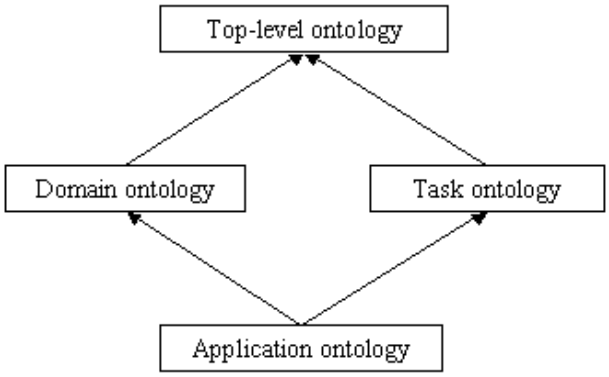


Fig.III. 2 Different types of ontologies [Guarino, 1998]

- Top-level ontologies (also called upper or foundational ontologies) attempt to describe very abstract and general concepts that can be shared across different domains and applications. They adopt the philosophical notions for describing the top-level concepts for all things that exist, such as physical objects or abstract objects, as well as generic notions of common sense knowledge about phenomena like time, space, processes, etc. Due to their generality, they are typically not directly used in applications but rather for other ontologies to be aligned to. Prominent examples of top-level ontologies are DOLCE [Gangemi et al., 2002] and SUMO [Niles et al., 2001].

- Domain ontologies capture the knowledge within a specific domain (e.g. medicine, geography), or the knowledge about a particular task (e.g. diagnosing, configuring). They are narrower and more specific in scope than the top-level ontologies.

- Task ontologies are described with respect to a domain ontology, while the conceptualization in a domain ontology is kept strictly away from the task. The task ontologies were invented for scheduling and planning tasks, monitoring in a scientific domain, intelligent computer-based tutoring, missile tracking, execution of clinical guidelines, etc.

- Application ontologies provide the specific vocabulary required to describe a certain task enactment in a particular application context. They typically make use of both domain and task ontologies.

Finally, [Lassila et al., 2001] classified different types of lightweight (on the left) and heavyweight (on the right) ontologies in a continuous line, as shown in the Fig.III. 3:

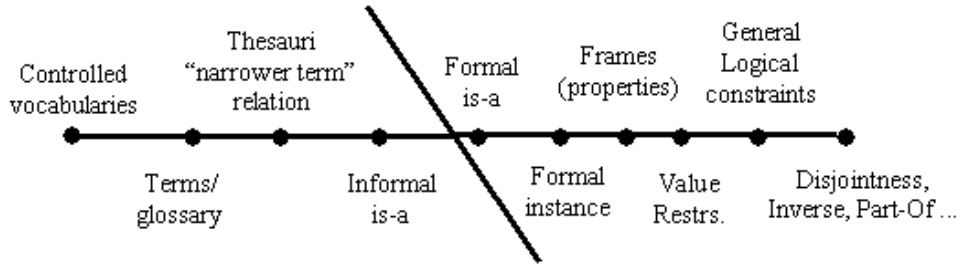


Fig.III. 3 Different types of ontology based on lightweight and heavyweight classification [Lassila et al., 2001]

A lightweight ontology is a structured representation of knowledge. It ranges from a simple enumeration of terms to a graph or taxonomy where the concepts are arranged in a hierarchy with a simple (is–a) relationship between them. It is usually informal and sufficient to define concepts and basic relationships between them. The lightweight ontology can be classified as follows:

- Controlled vocabularies. For example a catalogue.
- Glossaries are a list of terms with their meanings specified as natural language statements.
- Thesauri provide some additional semantics between terms. They give information such as synonym relationships, but do not supply an explicit hierarchy.
- Informal *is-a* hierarchies are taken from specifications of term hierarchies. Such a hierarchy is not a strict subclass or *is-a* hierarchy. For instance, the terms car rental and hotel are not kinds of travel but they could be modelled in informal *is-a* hierarchies below the concept travel.

A heavyweight ontology adds more meaning to the structure by providing axioms and broader descriptions of knowledge. Axioms and constraints tend to reduce the ambiguity in the knowledge base by restricting and constraining the usage of information. It is a formal ontology because it can support more complex queries and delivers comprehensive answers. The heavyweight ontology can be classified as follows:

- Formal *is-a* hierarchies are necessary to exploit inheritance.
- Formal instance hierarchies include instances (individuals) of the domain.
- Frames: the ontology includes classes and their properties, which can be inherited by classes of the lower levels of the formal *is-a* taxonomy.
- Ontologies express value restriction. These are ontologies that may place restrictions on the values that can fill a property. For instance, the type of the property *name* is a *string*.
- Ontologies express general logic constraints. These are the most expressive ontologies
- First-order logic constraints: very expressive ontology languages such as those seen in Ontolingua or CycL allow first-order logic constraints between terms and more detailed relationships such as disjoint classes, disjoint coverings, inverse relationships, part-whole relationships, etc…

2.4. Ontology languages

A set of AI-based ontology implementation languages was created at the beginning of the 1990s. [Gomez-Perez et al., 2004] stated that basically the knowledge representation paradigm underlying such ontology languages was based on first-order logic (e.g. KIF), on frames combined with first order logic (e.g. Ontolingua, OCML, and FLogic), or on DL (e.g. LOOM).

The boom of the Internet led to the creation of ontology languages that exploit the characteristics of the web. Such languages are usually called web-based ontology languages, or ontology mark-up languages. These languages are shown below:

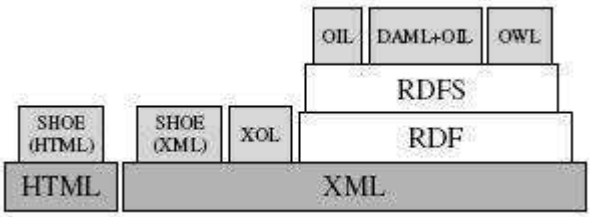


Fig.III. 4 Stack of ontology mark-up languages [Corcho et al., 2002]

The history of ontology mark-up languages has developed since 1996 when SHOE [Luke et al, 2000] was built as an extension of HTML in the University of Maryland. It combines frames and rules, and allows ontologies to be inserted in HTML documents. Then XML [Bray et al., 2004] was created and widely adopted as a language for exchanging information on the web. Consequently, SHOE was modified to use XML. Other ontology languages were built on the XML syntax after that. RDF [Lassila et al., 1999] and RDF Schema were developed by the W3C as a semantic network-based language to describe web resources. These languages established the foundations of the Semantic Web. RDF(S) is the combination of both RDF and RDFS. RDF(S) provides a simple ontology language for conceptual modelling with some basic inferencing capabilities. New languages have been developed as extensions to RDF(S), such as OIL, DAML+OIL, and OWL.

In this section, we detail only RDF(S) and OWL which are important languages, widely adopted and extended. For a more complete state-of-the art description of ontology languages see [Corcho et al., 2002] [Gomez-Perez et al., 2004] [D8.1 InterOp, 2004] [Godoy, 2005] [Grimm et al., 2007].

2.4.1. RDF and RDF Schema

The Resource Description Framework (RDF) is an XML application. XML provides outstanding support for defining vocabularies but it fails to provide proper semantics of the data it holds [Godoy, 2005]. To overcome this limitation, the RDF has been defined.

The syntax of RDF is defined in XML [Klyne et al., 2003]. It is a W3C recommendation that defines a general-purpose language for defining meta-data in the web. RDF is particularly intended for representing meta-data about web resources (e.g. title, author, copyright, etc.). It

does not require that resources be retrievable on the web and is therefore suitable for representing any kind of meta-data.

The approach for representing meta-data about resources in RDF is based on a few main ideas:

- Identification through URI: The identification of entities is based on their URI (Uniform Resource Identifier), which is a compact string of characters. The URI exhibits some naming convention that allows for partitioning of name into namespace. For modelling ontologies in RDF, URI may be used to identify these kinds of entities: individuals, things, properties of those things, and values of those properties.

- Sentences with subject, predicate, and object: Any statement in RDF is in the form of subject-predicate-object expressions, called triples in RDF terminology. The subject denotes the resource, and the predicate expresses a relationship between the subject and the object. For example, the statement *Netty works at EBM WS* is represented in RDF as the triple: a subject denoting *Netty*, a predicate denoting *works at*, and an object denoting *EBM WS*.

- Graph representation: Several triples taken together form an RDF graph whose nodes are resource URIs, and whose arcs are properties. A node in an object position can be either a resource, or an RDF literal (representing a data value). Moreover, RDF graphs support blank nodes which represent anonymous resources.

- XML serialization: The RDF recommendation defines an XML-based syntax called RDF/XML in which RDF graphs are encoded for machine processing. Fig.III. 5 illustrates an example of RDF/XML:

```
<?xml version="1.0" encoding="UTF-8"?>
        <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xml:p1="http://ebmwebsourcing.com">
                 <rdf:employees rdf:ID="EBMWS_001">
                          <p1:firstName>Netty</p1:firstName>
                          <p1:lastName>Rajsiri</p1:lastName>
                 </rdf:employees>
        </rdf:RDF>
```

Fig.III. 5 RDF/XML Example

According to [D8.1 InterOp, 2004], RDF does not impose any interpretation on the kinds of resources involved in a statement beyond the roles of subject, predicate, and object. It has no way of imposing agreed meaning on the roles or the relationships between them. The RDF Schema is a way of imposing a simple ontology on the RDF by introducing a system of simple types.

The RDF Schema (RDFS) [Berners-Lee et al., 2001] is an extension to RDF. It provides vocabularies for RDF. While RDF is used to relate resources by means of properties, RDFS introduces the notions of resource classes, subclasses, and properties. It can also impose restrictions on the domain and range of properties. Classes (*rdfs:Class*) are resources representing a collection of resources, such as books, cars, persons, etc. A resource can be

identified as a member of a class by means of the properties (*rdf:type*). RDFS also supports restriction of the values of a property to be members of a particular class using the *rdfs:range* property, and restricts the subjects of a property (*rdfs:domain*).

RDFS has been contrived with limited semantics [Klein et al., 2003] [D8.1 InterOp, 2004] [Godoy, 2005] because it was designed to be a simple, formal, and extensible ground for defining shared vocabularies for the web. RDFS lacks many ontological constructs such as equivalence, inverse, symmetric, or transitive relationships. Because of these limitations of expressiveness, the W3C created a more expressive language, OWL.

2.4.2. OWL

The Web Ontology Language (OWL), endorsed by the W3C, is a family of knowledge representation languages for authoring ontologies. Intuitively, OWL can represent information about categories of objects and how objects are interrelated. It can also represent information about objects themselves.

OWL was created in order to overcome the limitations of expressiveness caused by RDF and RDFS. OWL ontologies are therefore richer and more expressive than those provided by RDF and RDFS. OWL is derived from the DAML+OIL Web Ontology Language. It builds upon RDF and RDFS and is defined as a vocabulary extension of RDF. The modelling primitives of OWL include those from RDF, and the majority of the RDFS constructs. Thus, every OWL document is a valid RDF document. OWL was designed to support the use, modification, and integration of ontologies over the web. An important issue for the design of OWL was the balance between expressivity of the language and scalability of reasoning.

OWL allows classes to be described by specifying relevant properties belonging to them. Properties can be described by defining their domains (classes) and ranges (classes, string or integer datatypes). It is possible to declare properties as transitive, symmetric, functional or inverse of other properties. OWL also provides restrictions on how properties behave. OWL can be used to restrict the models to meaningful ones by organising classes in a subclass hierarchy, as well as properties in a subproperty hierarchy.

[D8.1 InterOp, 2004] mentioned that OWL is quite a sophisticated language. OWL semantics is formalized by means of a DL (Description Logic) style model theory. In addition, OWL's formal specification is a frame-like syntax which makes OWL easier to understand and to use. Its axioms are easily expressible by means of a set of RDF triples (subject-predicate-object expression). This property is essential for connecting with the Semantic Web.

OWL has three different sublanguages with increasing expressivity: OWL Lite, OWL DL, and OWL Full. These three languages will be introduced in the next sections in order of increasing expressivity.

- OWL Lite was originally intended to support those users primarily needing a classification hierarchy and simple constraints. OWL Lite is the most restrictive flavour of OWL. It is the least expressive one, but supports the most efficient reasoning. It is expected to promote a wider adoption of semantic technologies in the

W3C. OWL Lite does not allow the use of nominals, but it allows only for unqualified number restrictions in the form ≤ 1 R. All OWL DL constructs can be captured in OWL Lite, except those containing either individual names or cardinalities greater than 1 [D8.1 InterOp, 2004]. Development of OWL Lite tools has thus proven almost as difficult as development of tools for OWL DL, and OWL Lite is not widely used [Godoy, 2005].

- OWL DL was designed to provide as much expressiveness as possible while retaining computational completeness (all conclusions are guaranteed to be computed), decidability (all computations will finish in finite time), and automated reasoning. OWL DL includes all OWL language constructs, but can only be used under certain restrictions. For example number restrictions may not be placed upon properties which are declared to be transitive. OWL DL is based on the description logic $\mathcal{SHOIN}$(D) [Horrocks et al., 2003].

- OWL Full is different from OWL Lite or OWL DL. It was originally designed to preserve some compatibility with RDF Schema. OWL Full allows classes to be used as individuals and the language constructors to be applied to the language itself. [D8.1 InterOp, 2004] mentioned that OWL Full goes beyond OWL DL, but makes reasoning undecidable because it needs expressively all the OWL, RDF, and XML. It is unlikely that any reasoning software will be able to support complete reasoning for OWL Full.

2.4.3. Conclusion

In the context of the Semantic Web, languages based on XML are widely used for exchanging ontologies between applications. We have presented RDF(S) and OWL in this section.

RDF was developed as an extension of XML to overcome the XML limitations. RDF is a representation language and has model-theoretic formally defined semantics. But it is still too weak to express many properties. So, RDFS was developed as a complement to RDF by adding meanings to some vocabularies used within RDF. However, RDF and RDFS were still not expressive enough. OWL was thus developed upon RDFS and became the most expressive language. [Gomez-Perez, 2004] pointed out some differences between RDF(S) and OWL. For example, RDF(S) cannot represent the cardinality constraints, while OWL can. So, OWL seems to be the better language for modelling ontology.

Now we have to choose the sublanguage of OWL. [Horridge et al., 2004] defined some simple rules for deciding the appropriate sublanguage as follows:

- The choice between OWL Lite and OWL DL may be based upon whether the simple constructs of OWL Lite are sufficient or not. OWL Lite has a limited notion of cardinality (0 or 1).

- The choice between OWL DL and OWL Full may be based upon whether it is important to be able to carry out automated reasoning on the ontology or if it is important to be able to use highly expressive and powerful modelling facilities such as meta-classes.

For our case, we intend to carry out automated reasoning and we may need to define more than one cardinality for some concepts in our ontology. So, OWL DL is the one we chose.

2.5. Related ontologies

In the last decade, many ontologies have been developed for different purposes. They cover different domains of interest such as medicine, tourism, knowledge, etc. Before developing our own ontology, we explored the related ontologies in the business process and enterprise modelling domains. The following paragraphs present some of them:

2.5.1. AIAI Enterprise Ontology

An enterprise ontology is a collection of terms and definitions used in organizations. The AIAI (Artificial Intelligence Applications Institute) enterprise ontology [Uschold et al., 1998] was developed in the scope of the enterprise project whose goal was to provide a set of tools for enterprise modelling. The available tool set contains a procedure builder for capturing process models, an agent toolkit for supporting agent development, and a task manager for integration and visualization. The ontology was used in order to ensure a consistent communication between agents, either human or software. The enterprise ontology built within the enterprise project is not meant to be a complete ontology describing the enterprise domain. It only presents the most frequent terms used in this field. Thus, the ontology has to be enriched for each specific business case.

The enterprise ontology is divided into five top-level concepts: activities-processes, organization, strategy, marketing, and time:

- The organization part contains the terms representing the actors that play a role in an enterprise. They can have legal responsibilities or not, and can be human or machine.

- The activity-process part includes the concept of resources and skills. It contains the concept of input/output.

- The strategy part describes the concept of purpose. Purpose captures two related notions. One is the intended reason for executing an activity specification (what a PLAN is for). The other is something that an organization unit can be responsible for (defined in the organization part).

- The marketing part describes sales. A sale is an agreement between two legal entities for the exchange of a product for a sale price.

- The time part is not specific to enterprises, but is used by them. Normally, a time interval is required to refer to when activities are performed. A time interval is defined in terms of time points which in turn make up a time line.

2.5.2. TOronto Virtual Enterprise Ontology

The TOronto Virtual Enterprise (TOVE) ontology was developed in the scope of the TOVE project [Fox, 1992]. The TOVE ontology is a formal representation of the enterprise domain. It intends to 1) provide a shared terminology for the enterprise that each agent can jointly understand and use, 2) define the meaning of each term in as precise and unambiguous a manner as possible, 3) implement the semantics in a set of axioms that will enable TOVE to automatically deduce the answer to many common-sense questions about the enterprise, and 4) define a symbol for depicting a term or the concept constructed therefrom in a graphical context.

The TOVE ontology was developed in cooperation with several companies. It has been applied to the design and analysis of enterprise models within supply chain management, project management, and business-process engineering. The TOVE ontology is divided into several top-level concepts to segment the enterprise into general categories: activity, states, causality, time, resources, and organizational structure. Here we introduce some categories:

- Activity and states ontologies: an activity is the basic transformational action primitive with which processes and operations can be represented. An enabling state defines what has to be true for the activity to be performed. A caused state defines what is true once the activity has been completed.

- Resource ontology comprises two sets of terms or assertions. The resources are defined in terms of knowledge, role, mobility, and division of the resource. The role of the resource represents its nature, such as product, tool, or work area. Mobility specifies the possibility of moving the resource from one place to another. Divisibility of the resource specifies if the resource can be divided into several resources, without affecting its role in an activity.

- Organization ontology describes an organization entity which can be an individual, or a group denoting several people (e.g., board of directors, teams, etc.). Each organization has properties such as role, skill, constraint, etc. The role specifies the goal that the organization has to achieve. Each role attaches skills, processes, policies, etc., which are necessary to complete the goal.

The concepts encoded in the ontology are also enriched by a set of axioms that define and constrain the interpretation of these concepts. The ontology is formalized using first-order logic, allowing questions to be answered by using the TOVE reasoning engine.

2.5.3. The Business Process Management Ontology

The primary goal of the Business Process Management Ontology[7] (BPMO) is to provide a stable platform for the semantically rich definition of business processes, in order to better align information technology (IT) with business. The BPMO makes it possible to define private and public processes, business entities, business objects, and services that implement

[7] http://www.bpiresearch.com/Resources/RE_OSSOnt/re_ossont.htm

process activities. Currently, it comprises approximately 650 classes. The ontology is available in the OWL format.

It follows the UN/CEFACT modelling methodology (UMM) for business process and information modelling. Business entities are defined, according to the UN/CEFACT glossary, as something that is accessed, inspected, manipulated, and produced in the business. Once these entities have been defined, they are generalized under new concepts called business objects. For instance, the business entities customer and supplier may be represented by a business object named party, which is a generalization of customer and supplier [BPMO Tutorial].

The BPMO also introduces the concept of the process task. It describes which role performs a task, which business entities and business documents it is related with, and which resources it consumes. Every task represents a defined context which includes the following items:

- Role: A common responsibility or position of one or more physical actors. An actor may be a member of one or more roles. For example: buyer.

- Business document: A set of information components that are interchanged as part of a task. A business document may participate in a message flow. For example: purchase order form.

- Durable information entity: An information entity that a task needs to perform its function. It may be composed of multiple business objects. For example: purchase order information.

- Resource: A real object that can be identified. For example: fax.

2.5.4. Process Specification Language Ontology

According to [Schlenoff et al., 1999], the Process Specification Language (PSL) is an attempt to create a formalism for the representation of processes that is common to all manufacturing applications. The PSL ontology is formally defined using first-order logic and the KIF[8] language to encode axioms. The ontology is divided into two layers:

- A PSL-Core comprising concepts which are common to all manufacturing applications. It is composed of four classes: activities, activities occurrences, timepoints, and objects. A process can be defined as one or more activities that occur over a period of time in which objects participate.

- A set of extensions providing the resources in order to express other concepts which are not present in PSL-Core. For example: activity extensions, resource roles, resource sets.

[8] http://ksl.stanford.edu/knowledge-sharing/kif/

2.5.5. Collaborative Networked Organization Ontology

The Collaborative Networked Organization (CNO) ontology was developed in the scope of the ECOLEAD project [ECOLEAD]. It aims to create strong foundations and mechanisms needed to establish the most advanced collaborative and network-based industry society in Europe.

A collaborative network is dynamic in the sense that organizations or individuals may join or leave the network whenever they want. This ontology focuses on a particular type of collaborative network called the Virtual Breeding Environment[9] (VBE) [Camarinha-Matos et al., 2005]. It can be organized into four sub-ontologies:

- Top-Level of the CNO ontology (domain ontology): The two top-most concepts are CNO and organization. A *CNO* is a special type of collaborative network comprising only organized collaborations. Special types of *organizations* are *VO*[10] *(Virtual Organization)*, *VBE,* and *VBE Participant*. *VO* and *VBE* are also special types of *CNO* since they represent alliances of companies, and individuals. The partners of the *VO* are selected from the *VBE participants* according to their competencies and availability to deliver products or services required to fulfil a *Business Opportunity* (BO).

- Organization-related ontology: this includes the concepts describing an organization's capabilities such as profile, competency, capacity, process, resource, and product/service. The concept *Profile* is a set of structured information describing the organization, such as name, contact information, description etc. Each organization covers one or more *Competencies* which define its *Capability* to perform processes. A *Process* is a structured, managed and controlled set of interrelated activities that uses *Resources* to transform inputs into specified outputs.

- VBE-Role-related ontology: A *Participant* is an organization within a VBE and can participate in different manners. Each participant in the VBE can have one or more specific *Roles*: *members of the VBE*, *support institutions*, and *public entities*. All these roles have in common a relation with the concept *Task*. *Public entity* is the role taken by a participant which is not registered in the VBE. *Member role* can be *VO Partner*, or *VO Support provider*. *Support institutions* can provide a broad range of services (e.g. training, research, consulting, information system, etc.) and tend to propose a solution based on their expertise for the client.

[9] A Virtual Breeding Environment (VBE) is an association (also known as a cluster) of organizations and their related supporting institutions that have both the potential and the will to cooperate with each other through the establishment of a long-term cooperation agreement and interoperable infrastructure. The VBE responds to business opportunities by forming VOs. [Camarinha-Matos et al., 2005]

[10] A Virtual Organization (VO) is a short-term association with a specific goal of being active in fulfilling a Business Opportunity (BO). A BO is a time or occasion with a favourable combination of circumstances that is suitable to start a business. VO represents a temporary alliance of diverse organizations that form a collaboration network, sharing knowledge, skills and resources in order to respond to a specific BO. [Camarinha-Matos et al., 2005]

- VBE bag of assets-related ontology: The main purpose of the VBE related assets is to speed up and improve the process of a VO creation. In a VBE, the potential assets are , for example, lessons learned, sample contracts, general legal issues, information of interest, FAQs, etc. According to these potential assets, the concept *VBE_Asset* was introduced. Each asset belongs to a VBE encoded as the property *hasAsset* of the concept VBE.

The complete CNO ontology is shown below:

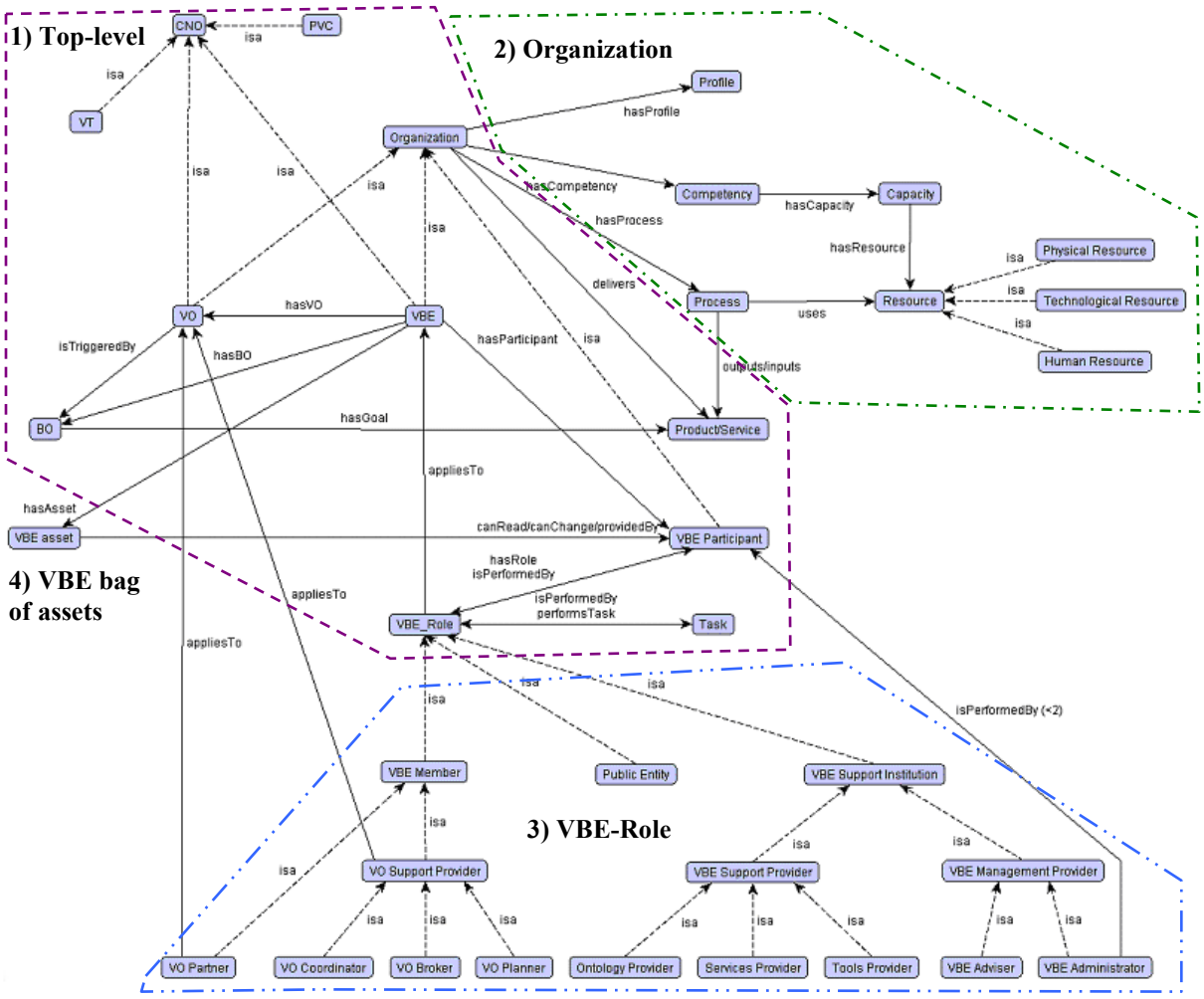


Fig.III. 6 The complete Collaborative Networked Organization ontology [Putnik et al., 2008]

2.5.6. MIT Process Handbook Ontology

The Process Handbook (PH) has been under development at the MIT Center for Coordination Science (CCS) for over ten years [Malone et al., 1999] [Malone et al., 2003]. As discussed in [Alessandro et al., 2007], the PH has been shown to be useful at a research level in a variety of domains such as business process reengineering [Bernstein, 1998] [Malone et al., 1999], business process automation [Bernstein, 2000], and software design [Dellarocas, 1996], etc.

The researchers within the dynamic and distributed information systems group of the University of Zurich have participated in the MIT PH project since it began. They decided to bootstrap an OWL ontology based on the PH in order to evaluate their Semantic Web applications. The OWL PH is provided within the dataset, an OWLized version of the MIT PH. The dataset contains not only the OWL PH schema file, but also approximately 5000 business processes which are stored in their own OWL files. Fig.III. 7 illustrates the PH ontology:

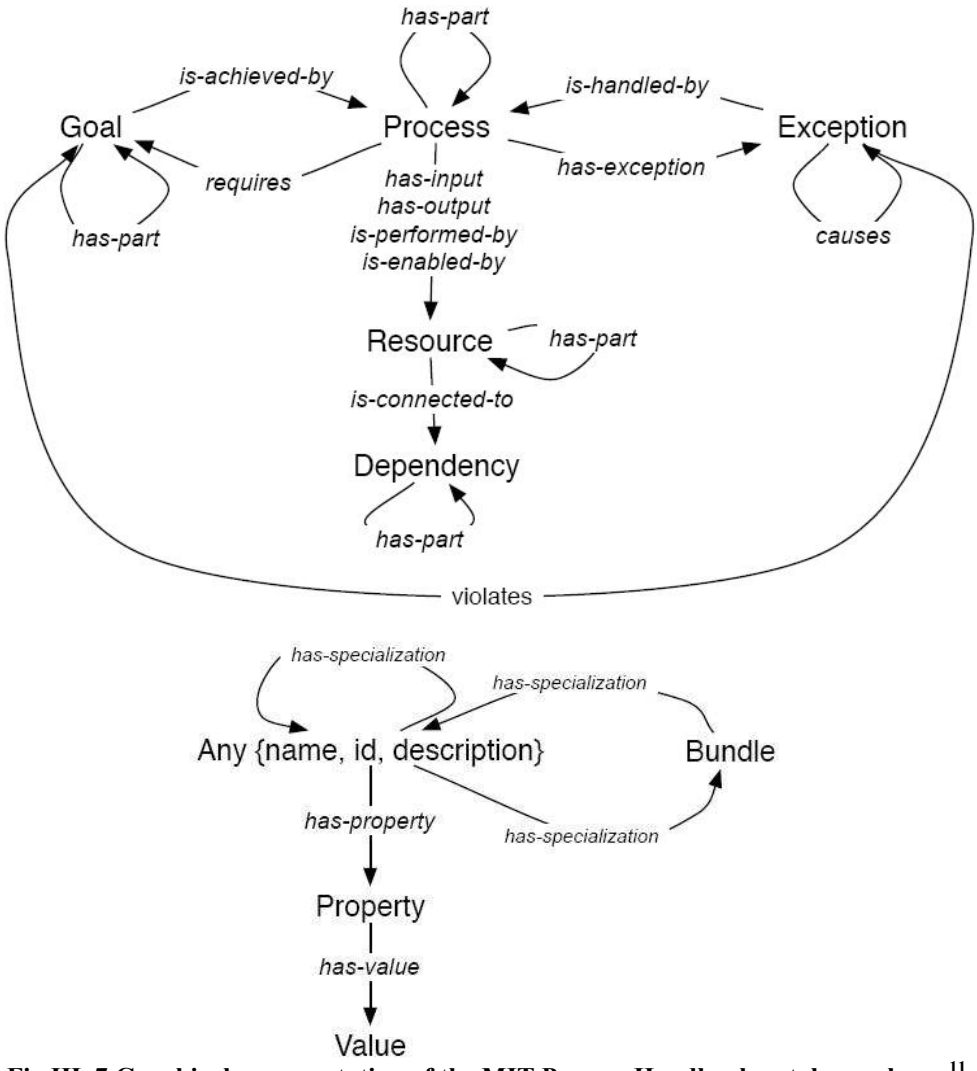


Fig.III. 7 Graphical representation of the MIT Process Handbook ontology schema[11]

The PH ontology provides a specialization hierarchy of processes and their inter-relationships in the form of properties which connect the process to its attributes, parts, exceptions, and dependencies to other processes. Specialization in the PH is non-monotonic. In other words, it is possible for a child process to overwrite or delete an inherited property. The PH thus has the advantage of being a sizable data set that was developed in a real-world setting.

[11] http://www.ifi.uzh.ch/ddis/fileadmin/ph/ProcessHandbook-Schema-16-10-06.pdf

All major parts of the PH, such as Process, Bundle, Goal, Exception, Resource, Dependency, and Trade-offs are represented as OWL classes (Fig.III. 7). The 5000 business processes in the PH have been written in OWL and stored as instances in their own files. The key elements of the PH ontology are:

- Process: Like most process-modelling techniques, the PH allows processes to be annotated with attributes that capture such information as a textual description, typical performance values (e.g. how long a process takes to execute), as well as conditions. A process is modelled as a collection of activities that can in turn be broken down into sub-activities.

- Resource: A process consumes and produces resources. In other word, resources describe input and output of processes they are related to.

- Dependencies: Another key concept is that coordination can be viewed as the management of *dependencies* between processes [Malone et al., 1999]. Every dependency can include an associated *coordination mechanism* which is simply the process that manages the *resource* flow and thereby coordinates the activities connected by the dependency.

- Goal: allows business processes to be composed, or monitors their execution.

- Exceptions: It is possible that processes can fail because of exceptions. Therefore we have to anticipate, avoid, or detect and resolve them.

- Bundle: this is a group of related specializations. In general, it is often very useful to create bundles based on the basic questions for asking about any activity: how? what? who? when? where?, and why?.

2.5.7. Conclusion

We have presented the existing ontologies related to the business process and enterprise modelling domains in this section: AIAI, TOVE, BPMO, PSL, CNO of ECOLEAD, and MIT Process Handbook ontologies.

The AIAI ontology is focused particularly on intra-enterprise modelling in order to ensure a consistent communication between humans or software applications. The TOVE and CNO of ECOLEAD ontologies are focused more on virtual enterprise modelling. Both are conceptualized in the organizational level, taking into account such concepts as participant, role, and activity. The BPMO, PSL, and MIT Process Handbook are oriented to process modelling in the functional level. The BPMO ontology provides a stable platform for the definition of business processes in order to better align information technology (IT) with business, which is the same intention as our work. The PSL ontology was originally created to represent processes for manufacturing applications. The MIT Process Handbook ontology is more generic and can be applicable to any industries and business domains.

Since we intend to develop an ontology that covers the inter-enterprise collaboration and collaborative business process domains, the nearest to our intention is the MIT Process Handbook ontology. Even though the MIT Process Handbook is not directly dedicated to the inter-enterprise context, its modelling approach, as discussed in Section 2.3.2 of Chapter 2, shows that it can be applied to such a context. Besides, we can reuse the knowledge (instances) from the MIT Process Handbook to define collaborative processes. We will present our ontology in detail in the next section including its concepts, and the relations between them.

3. Collaborative Network Ontology (CNO)

Our ontology is called *Collaborative Network Ontology* since it deals with collaboration as well as providing common definitions in collaboration, and network domains. Our Collaborative Network Ontology has the same abbreviation, CNO, as the Collaborative Networked Organization ontology of the ECOLEAD. Note that we are talking about a collaborative network, not a collaborative networked organization in particular. The main difference is that a collaborative network can be an organized collaboration or a collaboration between individuals, while the ECOLEAD ontology can only apply to organized collaborations.

Note that we developed our CNO on the basis of the characteristics of collaboration and collaborative process studied in Chapter 2. We were also inspired by the related ontologies in the enterprise and modelling domains, particularly the MIT Process Handbook ontology. This ontology is oriented to business process modelling which corresponds to the scope of our work.

As we have discussed previously, knowledge has static (e.g. domain knowledge) and dynamic (e.g. inference, task knowledge) behaviours. Ontologies are an appropriate means for defining static knowledge, reusing knowledge bases, and integrating pre-existing knowledge-based components. Dynamic behaviour is usually defined as a mixture between traditional software procedures and inference rules that declaratively define reasoning steps [Godoy, 2005].

Thus, we developed our CNO consisting not only of ontologies, including their concepts, relations and properties, but also rules. The ontologies in the CNO are: 1) the Collaboration Ontology (CO) representing the elements of a collaborative network, and 2) the Collaborative Process Ontology (CPO) representing the elements of a collaborative process. The deduction rules connect these two ontologies together by the semantics and structural links. They also reflect the notion of consequence and allow us to carry out deductive reasoning in the knowledge-based system. Hence, these two ontologies refer to the concepts defined in the left and right worlds of Fig.II.10, while the rules refer to the connections between these two worlds.

In this section, we present our ontology-based approach, which concerns the CNO and the deduction mechanisms. The ontologies and the rules will be presented separately. Firstly, we introduce the conceptualization, and the references on which the CO and CPO are based. Then, the deduction rules are discussed with some examples.

3.1. Ontologies

An ontology can be represented by a graph whose nodes represent concepts and whose links are relations between concepts. Our full CNO ontology is shown in Fig.III. 8 whose upper and lower parts represent the CO and CPO respectively:

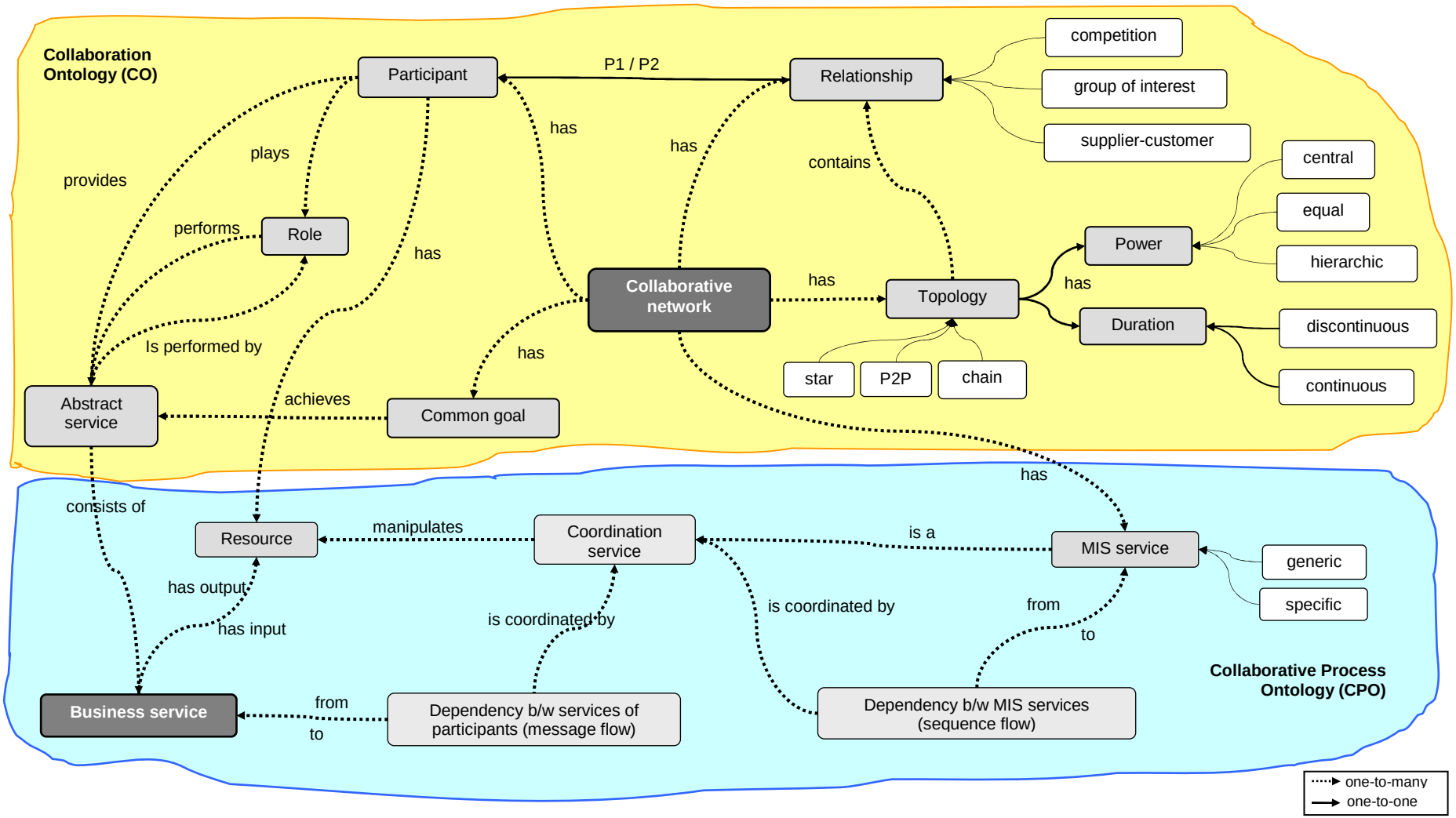


Fig.III. 8 CNO composed of Collaboration and Collaborative process ontologies

3.1.1. Collaboration Ontology (CO)

The Collaboration Ontology or CO refers to the conceptualization of enterprise collaboration, and the collaborative network characteristics. In the literature study (Chapter 2), we talked about the different approaches concerning network elements, and collaboration characterization for constituting a network for collaborating. Thus, the CO has mostly been defined on the basis of these approaches.

3.1.1.1. Definitions of concepts and meta-model of CO

We have organized the CO into two main categories: participant and collaboration. The following paragraphs detail the concepts of these two categories, together with relations between concepts.

a. Participant category

The participant category involves descriptions of each individual in the scope of the collaboration. This category has three concepts described as follows:

- A Participant can be a physical actor or an enterprise that joins the network in order to achieve a common goal collaboratively with other participants.
- A Role defines the responsibility of a participant in the network. For example: seller, buyer or producer. Role refers to a resource as defined in the MIT Process Handbook.
- The Abstract service is a high-level service that explains the competencies or the know-how of the participant. For example: marketing and sales, procurement. This concept comes from the BAM concept of the MIT Process Handbook (Section 2.3.2.1 of Chapter 2).

Fig.III. 9 illustrates the relations between the concepts, which can be described in terms of every participant playing roles and providing abstract services corresponding to the roles played.

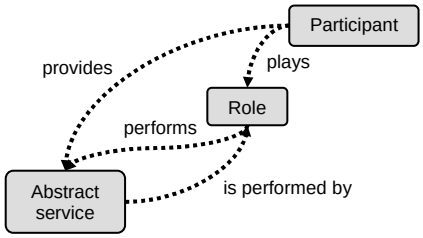


Fig.III. 9 Relations between participant, role, and abstract service

From the above figure, deductive reasoning can occur, for example between role and service. The related abstract services will be derived from a given role and vice-versa. For example, if the role is *computer maker* then its services are *making screen*, *making keyboard* etc.

b. Collaboration category

The collaboration category concerns the characterization criteria of collaboration: common goal, participant, relationship, and topology. The definitions of these concepts are described as follows:

- A Collaborative network is a group of at least two participants who would like to work together in response to one or multiple common goals and a set of relationships between the participants.

- A Common goal describes the reason why the network is established in terms of products or services to deliver to customers [Zaidat, 2005]. It gives the direction the partners have to head for and achieve.

- A Relationship defines the interaction between two participants. It describes how partners connect to each other. As discussed in Section 1.2.2 of Chapter 2, three types of relationship have been classified [Fombrun et al., 1982]: competition - if enterprises are in the same sector of business, supplier-customer - if enterprises collaborate with their partners who supply them with complementary services, and group of interest - if enterprises are neither in substitutability nor vital complementary, but annexed additivity.

- Topology describes the relationships between partners at high level and the overall structure of the network. As discussed in Section 1.2.3 of Chapter 2, three basic forms of topology based on the circulation flow in the network have been presented in [Katzy et al., 2000]: chain, star, and peer-to-peer. The form of topology can be distinguished by the orientation of decision-making power and duration of collaboration in the network (see the Table II.3)

- Decision-making power describes the behavior and the orientation of decision-making in the network. Three decision-making powers are distinguished: central, equal or hierarchic. These three kinds are inspired from the topology characteristics.

- Duration describes the frequency of interactions that occur during the collaboration in the network. [Zaidat, 2005] distinguished two kinds of duration: continuous or discontinuous.

For readability reason, we have cut the ontology into several small parts. The following figures explain the relations between concepts in the collaboration category.

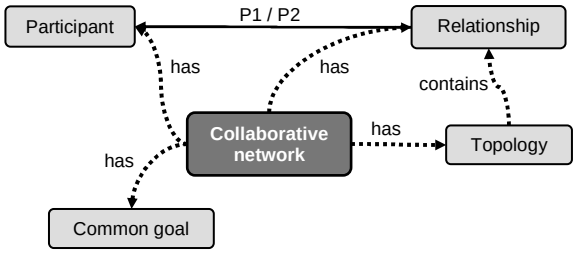


Fig.III. 10 Characteristics of a collaborative network

Every collaborative network has a group of participants who work together in response to some goals. It also has a set of relationships, each of which represents the connection between two participants, and has topologies containing some relationships that all have the same properties.

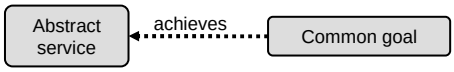


Fig.III. 11 Relation between common goal and abstract service

Every common goal achieves some abstract services provided by participants. The relation between participant and abstract service concepts has been discussed in the previous category.

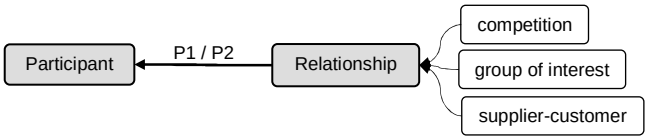


Fig.III. 12 Types of relationship and relation between relationship and participant

Every relationship links two participants (P1 (participant 1) and P2 (participant 2)). Group of interest, supplier-customer, or competition is a special kind of relationship.

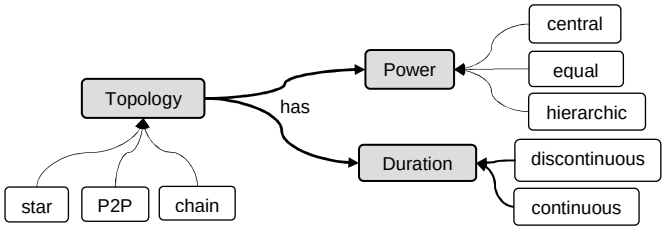


Fig.III. 13 Types and characteristics of topology

Every topology has duration and decision-making power concepts. Central, equal, or hierarchic are special kinds of decision-making power, while continuous, or discontinuous is a special kind of duration concept. Star, peer-to-peer, or chain is a special kind of topology.

3.1.2. Collaborative Process Ontology (CPO)

The Collaborative Process Ontology or CPO refers to the conceptualization of a collaborative process. We studied the characteristics of collaborative process and already defined what our collaborative process should be in Section 2 of Chapter 2. The CPO is an extension of the concepts developed by the MIT Process Handbook project [Malone et al., 2003] and the collaborative process meta-model [Touzi, 2007].

3.1.2.1. Definitions of concepts and meta-model of CPO

The CPO addresses business service, flow of resources between services and management of flows. It covers the concepts of business service, resource, dependency, coordination service, and MIS service. Only the MIS service concept comes from the meta-model of collaborative process, while the others are inspired from the OWL PH schema. In fact, the dependency concept of the PH schema can be considered as the message and sequence flows of the meta-

model of collaborative process. The coordination service concept is the main point for connecting the PH schema to the MIS service concept of the collaborative process meta-model. The definitions of these concepts are described as follows:

- Business service explains the task at functional level. An abstract service is composed of some business services. For example: assemble components of computer, obtain order. This concept is inspired from the functional level activity described in the BAM concept of the MIT Process Handbook (Section 2.3.2.1 of Chapter 2).

- Resource can be data, machine, software, tool or material used or produced by business service. For example: message, order, machine, container, technology. Resource concerns the resource concept defined in the MIT Process Handbook.

- Coordination service is in charge of managing the dependency of resources. For example: manage flow of material, manage accessibility of documents. This concept comes from the model of collaborative process concept of the MIT Process Handbook (Section 2.3.2.2 of Chapter 2).

- MIS service is defined in the meta-model of collaborative process (Section 2.2.2 of Chapter 2). We consider a coordination service as a MIS service because both are collaborative services provided by the collaborative platform (or MIS).

- Dependency between business services (message flow) is a flow from one business service to another when they have a resource in common. The two business services linked by this kind of flow do not belong to the same participant. It can be seen as a movement of a resource between business services.

- Dependency between MIS services (sequence flow) is a flow from one MIS service to another when they have a resource in common. It can be seen as a movement of a resource between MIS services.

The concept of dependency defined in the MIT Process Handbook does not distinguish the special kinds of dependency as we do here. The main difference between these two dependencies concerns the services they are dealing with. If it concerns business services, it is message flow. If not, it is sequence flow.

The following paragraphs detail these concepts, and the relations between them.

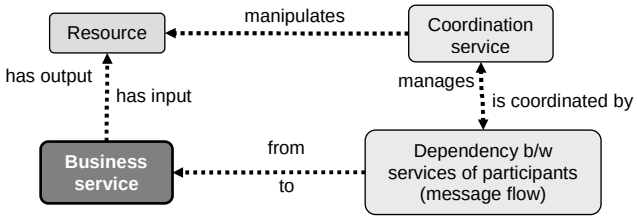


Fig.III. 14 Concepts of business service, resource, coordination service, and dependency

According to [Crowston, 1994], the business service explains the task at functional level. Every business service has input and output resources. Two business services are dependent

on each other when they have a common resource according to the concept of dependencies between resources.

To derive a dependency, we consider possible combinations of services using resources. Every dependency is associated with (at least) one coordination service. The concepts of dependency and coordination are related since coordination is seen as a response to problems caused by dependencies. This means a coordination service is in charge of managing a dependency. For example, if the *placing order service* of a *buyer* produces a *purchase order* as output and the *obtaining order service* of a *seller* also uses a *purchase order* as input then there is a dependency of resource between these two services and we can use the *forwarding document coordination service* to manage such a dependency.

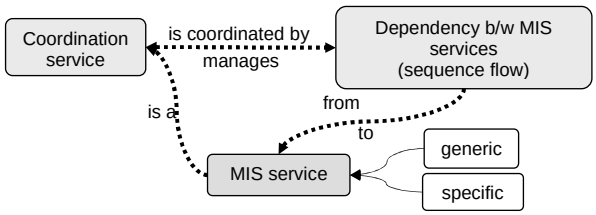


Fig.III. 15 Concepts of MIS, coordination service, and dependency

Every MIS service is a coordination service, as discussed in [Touzi, 2007], since MIS is defined as a mediation system that manages the collaboration and deals with the data and applications of participants. A collaborative network can have only one MIS. The MIS has its own MIS services which are generic (e.g., send documents/mails) or specific (e.g., select supplier service). The generic MIS service is a service that can be used by any collaborative process. This kind of MIS service already exists in a service repository of the MIS platform. The specific MIS service is a service newly created for a particular collaborative process. Such an MIS service can be defined from the coordination service and is an added-value service of the MIS.

3.1.3. Relations between the concepts of CO and CPO

According to the CO and CPO ontologies, we need to connect the concepts of these two ontologies in order to make the deduction rules operational. The connections are defined thanks to relations. These relations are marked with dotted lines in black, as shown in Fig.III. 16:

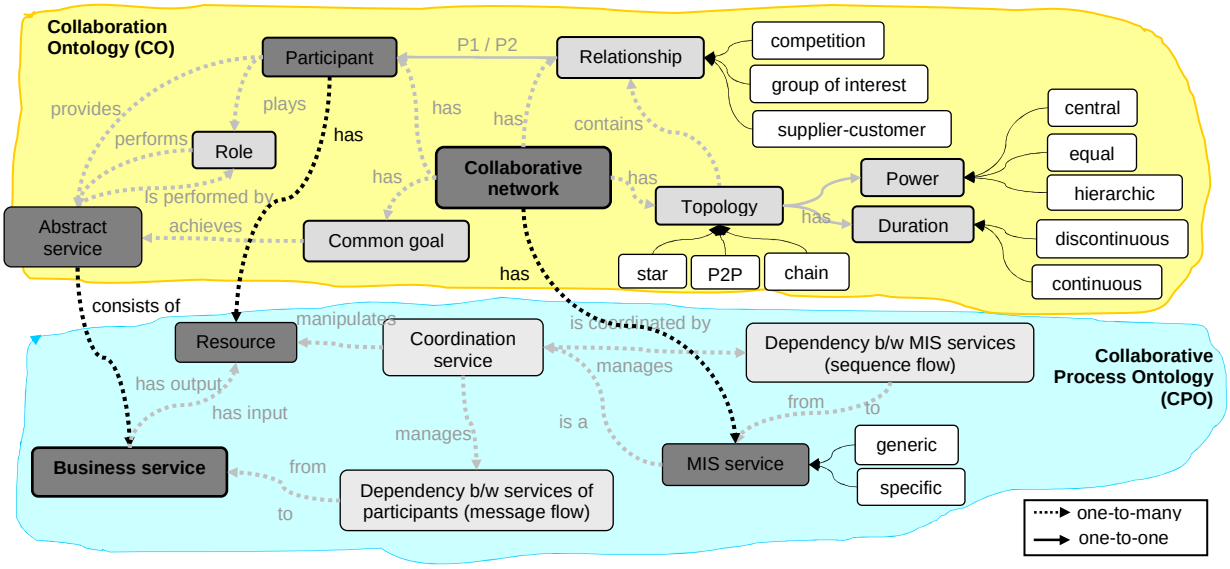


Fig.III. 16 CNO ontology and relations between CO and CPO in red

- Connection between collaborative network and MIS service via *has* relation:
  Every collaborative network has several MIS services which are coordination services. Each coordination service can manage several dependencies.

- Connection between participant and resource via *has* relation:
  Every participant has some resources which can be inputs or outputs of business services.

- Connection between abstract and business services via *consists of* relation:
  Every abstract service consists of some business services. This connection is the most significant one since it makes the deductions realisable. The idea of separating services into abstract services and their related business services comes from the BAM concept discussed in Section 2.3.2.1 of Chapter 2. This concerns the generalization and specialization concepts discussed in [Malone et al., 2003]. For example, if service is *making keyboard* then business services are *assembling circuit board*, *testing board*, etc.

3.2. Deduction rules

Rules are widely used in business applications including computer-aided training, diagnostic fact finding, compliance monitoring, and process control. Besides, rules are adopted for different purposes: not only reasoning instances, but also querying, as well as connecting rules for reasoning across domains, etc.

In our case, the implementation of deduction rules is vital since they establish the semantic connections between the CO and CPO. This means we intend to apply rules to reason across the collaboration and collaborative process domains. According to Fig.III. 1, the deduction rules will be executed once the collaborative network model has been imported into the CO as a new instance. The execution creates new knowledge in the CPO and completes knowledge in the CO. It is realisable since the reasoning is done over the generic knowledge (instances of

the Process Handbook) stored in the KB. Thus, we can say that the deduction rules can deal with the relations between the collaborative network and process domains that we defined in Fig.II.10.

Indeed, when defining such rules we should probably take into account the relations between the concepts of both ontologies. A key correspondence that makes the deductions concrete is that between the abstract service and the business service concepts defined in the CO and the CPO respectively. The rules defined in our system are mostly based on expertise and references found in the literature.

We shall start this section by selecting an appropriate language for editing rules. Then, we shall present the deduction rules written in the selected language. The possible deductions that can occur in our knowledge-based system will be also presented.

3.2.1. Selection of technology

Many different languages have been created to support rule definition, such as RuleML, SWRL, ISO Prolog, WSML, SWSL, etc. Such languages are built upon the principles of logic (e.g. first-order logic, horn logic program). Rules support logical reasoning of new knowledge from a pre-existing knowledge (precondition) [Russell et al., 2003].

In the next section, we introduce only RuleML and SWRL, which are the most prominent languages for supporting the integration into the Semantic Web architecture [Berners-Lee et al., 2001] [Godoy, 2005]. At the end of this part, we shall select an appropriate rule language for editing our deduction rules.

3.2.1.1. RuleML

The Rule Markup Language[12] (RuleML) is a standardization initiative that was started in 2000 [Boley et al., 2001] [RuleML Initiative, 2005] [Godoy, 2005]. It is defined by the Rule Markup Initiative. The Rule Markup Initiative is an open network of individuals and groups from both industry and academia formed to develop a canonical Web language for rules using XML markup and transformations from/to other rule standards or systems.

RuleML was developed to express both forward (bottom-up) and backward (top-down) rules in XML for deduction, rewriting, and further inferential transformational tasks. Its goal is to provide an open XML/RDF-based rule language that allows rules to be exchanged between different systems and components across the web. According to [Boley et al., 2002], RuleML encompasses a hierarchy of rules:

[12] http://www.ruleml.org/

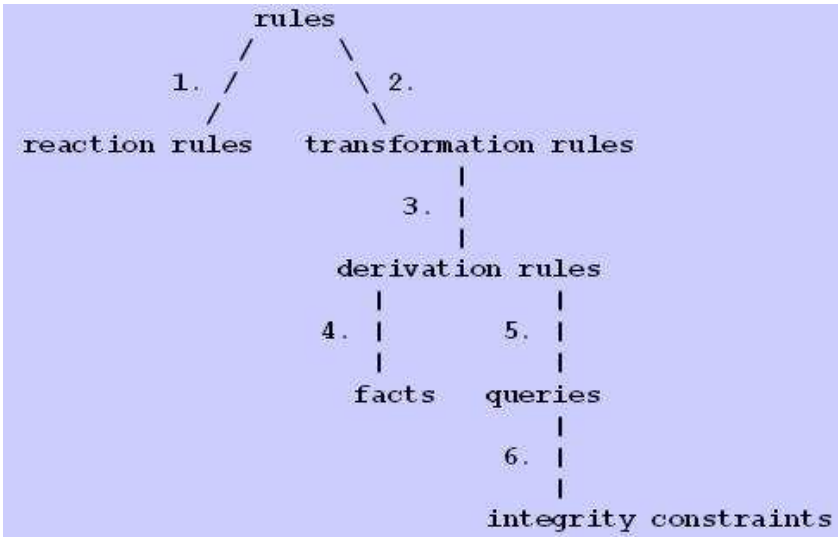


Fig.III. 17 Hierarchy of rules [Boley et al., 2002]

- Reaction rules (event-condition-action rules) are rules that return no value. They state triggering conditions, pre-conditions, and effects. They define the behaviour of a system in response to a particular condition or event.

- Transformation rules (functional-equational rules) are rules whose event trigger is always activated. They could be reduced to derivation rules over special relations that have an extra argument for the transformation values.

- Derivation rules (implicational-inference rules) are transformation rules that have a set of events/conditions (premises) and whose action only asserts a new fact or conclusion. Such rules can also be applied forwards for deriving new facts or backwards for proving a conclusion from premises.

- Facts are derivation rules that have an empty conjunction of premises.

- Queries are derivation rules that have an empty disjunction of conclusions or a conclusion that captures the derived variable bindings.

- Integrity constraints are queries whose action is to signal inconsistency when some conditions are fulfilled. They are usually applied forward upon update.

As we mentioned above, RuleML intends to be a common markup language for exchanging any kind of rules in any kind of language. So RuleML is also dedicated to inference rules. RuleML has a concrete XML syntax and it is divided in a hierarchy of sublanguages corresponding to well-known rule systems, such as SWSL, FOL+, Negation Datalog, etc. Since the syntax of RuleML is based on XML, XSL Transformations (XSLT) can be used to perform transformation into specific rule languages, for example N3, and Jess [Godoy, 2005].

As of September 2008, the latest version of RuleML is 0.91. (http://www.ruleml.org/0.91/). This version is the revised XML Schema specification (including Schematron annotations) for RuleML.

3.2.1.2. SWRL

The Semantic Web Rule Language[13] (SWRL) was proposed to the W3C in May 2004. SWRL is based on a combination of OWL DL and the Unary/Binary Datalog sublanguage of RuleML. According to [Parsia et al., 2005], SWRL is roughly the union of horn logic (RuleML) and $\mathcal{SHOIN}$(D) (OWL DL). It allows users to write Horn-like rules expressed in terms of OWL concepts to reason about OWL individuals. The rules can be used to infer new knowledge from existing the OWL knowledge base.

The XML concrete syntax combines the OWL XML syntax with the RuleML XML syntax in order to simplify the integration of OWL and RuleML. This syntax mainly facilitates mixing rules and OWL statements. Also it simplifies reusing previously developed RuleML tools for SWRL. The second syntax is a RDF-based syntax. The main rationale behind providing such syntax is to support automated reasoning over rules such as applying more specific versions of a rule to increase efficiency. The SWRL specification does not impose any restrictions on how reasoning should be performed with SWRL rules. But, SWRL rules reason about OWL individuals primarily in terms of OWL classes and properties.

SWRL rules are written as antecedent (body)-consequent (head) pair. The head and body consist of a conjunction of one or more atoms. However, SWRL does not support more complex logical combinations of atoms. Atoms can be A(?x), P(?x, ?y), sameAs(?x, ?y), or differentFrom(?x, ?y). A(?x) is an OWL class description. P(?x, ?y) is OWL property attached to OWL class. ?x is variable representing OWL description, while ?y is either variable, OWL individual, or OWL data value.

[O'Cornnor et al., 2005] gave an example of a SWRL rule for expressing that a person who has a child whose gender is male, has a son. This rule requires capturing the concepts of *person*, *gender*, *child* and *son* in OWL ontology. The concept of person can be captured using an OWL class called *Person*, while the *gender*, *child*, and *son* can be expressed using OWL properties *hasGender*, *hasChild*, and *hasSon* which are attached to the *Person* class. The rule in SWRL would then be:

Person(?x) ^ hasChild(?x, ?y) ^ hasGender(?y, "man") → hasSon(?x, ?y)

Fig.III. 18 Example of SWRL rule

SWRL rules make it possible to manipulate the instances by variables (?x, ?y, etc.). It does not create the concepts or the relations, but it adds the relations following the values of the variables and the satisfaction of the rule.

In addition, SWRL supports a range of built-in predicates. The built-ins are based on XPath and XQuery. Among these built-ins, we can find several kinds of operators for comparisons

[13] http://www.w3.org/Submission/SWRL/

(e.g. swrlb:equal, swrlb:lessThan), arithmetics (e.g. swrlb:divide, swrlb:mod), or string manipulation (e.g. swrlb:substring, swrlb:contains). These built-ins are described in detail in the SWRL Built-in Specification[14].

3.2.1.3. Conclusion

Rule languages have been developed in order to overcome the expressivity limitation of ontologies. Rules address the dynamic behaviour of knowledge by providing deductive reasoning of new knowledge or new facts to the knowledge-based systems.

Two rules have been presented in this section: RuleML, and SWRL. RuleML has existed for a number of years. It was designed to be an XML-based interchange language for rules on the web. RuleML can thus be considered as a standardized rule representation language. SWRL is more recent than RuleML. It integrates OWL DL and RuleML. Many constructs and namespaces of RuleML were incorporated into SWRL. Thus, SWRL rules could be mapped to an equivalent RuleML representation. The expressivity of SWRL is probably close to RuleML.

[Matheus, 2004] argued that whereas RuleML is designed to be all encompassing and quite flexible for implementing rules, SWRL is designed to be used in the OWL DL context and thereby inherit important semantic characteristics that make automated reasoning more tractable.

SWRL is OWL-specific, while RuleML focuses on dealing with the rule interchange problem. Since our ontology will be developed in OWL DL and automated reasoning is significant for us, it seems to be better if we use SWRL with OWL directly, not RuleML.

3.2.2. Specification of deduction rules

Since we intend to use the deduction rules to reason across the collaborative network and process domains, these rules should cover the relations defined on the mapping schema between these two domains (Fig.II.10).

The deduction rules defined in our knowledge-based system are mostly based on expertise and references found in the literature, as well as the mapping relations shown in Fig.II.10. We specify five groups of rules (GRs) as follows:

- GR1: Role and abstract service
- GR2: Business service
- GR3: Dependency and MIS service
- GR4: Common goal
- GR5: Topology

Some of these five GRs have several rules defined. We explain in the following sections only one rule for each group, and also give an example of instance deduction that can be made by

[14] http://www.daml.org/rules/proposal/builtins.html

this rule. All the rules in the same group work on the basis of the same concept. They are all shown in Annex A.

3.2.2.1. GR.1: Role and abstract service (Role ← → Abstract service)

This group is intended to derive abstract service and role when each is provided. There are two rules in this group:

- Deduction of abstract service when role is recognised (Role → Abstract service).
- Deduction of role when abstract service is recognised (Abstract service → Role).

According to [Petersen, 2005], each virtual enterprise is represented by its goals, the activities to achieve the goals, the roles that perform the activities and the skills that are required to fill the roles. This definition refers to the relation between role and activity. We consider this activity as abstract service which describes the competence of its provider.

Fig.III. 19 shows the first rule of this group written in SWRL:

Participant(?x) ∧ playRole(?x, ?y) ∧ performAService(?y, ?z) → provideAService(?x, ?z)

Fig.III. 19 Deducing abstract service from role

This rule starts at retrieving the roles of the participant and finding abstract services that can be performed by these roles. Then, the rule will return the list of abstract services that correspond to the roles the participant plays.

The second rule of this group works vice-versa. This means the rule derives role when abstract service is provided.

The following figure illustrates how the above rule works by showing instances with respect to their corresponding concepts in the ontology:

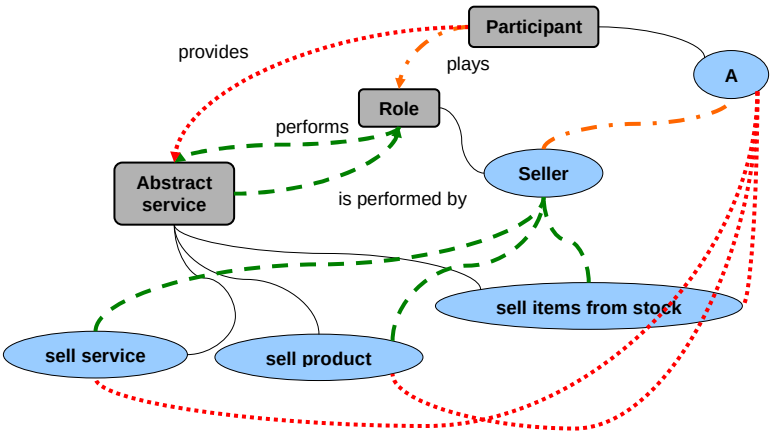


Fig.III. 20 Example of deduction by the rule in Fig.III. 19

The instances are represented as ellipses. The instance linked with dash-dot line is the one we are defining. Those linked with dotted lines are what the rule derives. Those linked with

dashed lines already existed in the Knowledge Base (KB) before running this rule. Otherwise, the rule does not function as it should. The figure explains that if the network has participant *A* who plays the role of *seller*, then participant *A* provides the *sell service*, *sell product*, and *sell items from stock* abstract services.

3.2.2.2. GR.2: Business service (Abstract service → Business service)

This group concerns the deduction of business services when an abstract service is recognised. We define this rule on the basis of the BAM concept discussed in Section 2.3.2.1 of Chapter 2. This concept states that every abstract activity has its corresponding functional-level activities. We consider these functional activities as business services.

There is only one rule in this group. Fig.III. 21 shows the SWRL rule deducing business services from abstract services:

Participant(?x) ∧ provideAService(?x, ?y) ∧ hasBusinessService(?y, ?a) → provideBusinessService(?x, ?a)

Fig.III. 21 Deducing business service from abstract service

If participant X provides the abstract services Y which have the business services A, then participant X provides the business services A. This rule starts at retrieving business services that correspond to the abstract services provided by the participant. Then, the rule will return the list of business services that the participant should expose.

This rule is the key that creates the semantic connections between the CO and CPO. The figure below shows the instances created respective to their corresponding concepts:

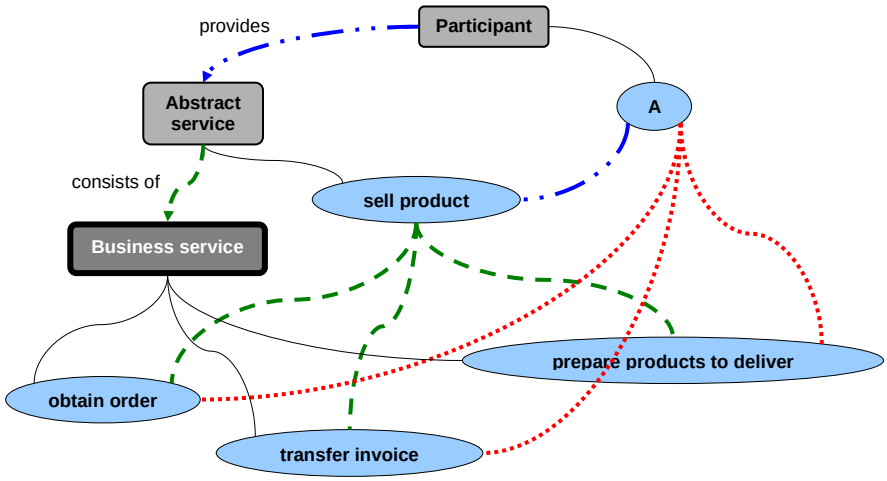


Fig.III. 22 Example of deduction by the rule in Fig.III. 21

In the same way as the first rule, the instances are always represented as ellipses. The instance linked with dash-dot-dot line has been derived from the first rule. The dashed lines mean the instances already existed in the KB. The dotted lines mean the instances are derived from this current rule. We continue the example of the first rule that deduced a list of abstract services provided by participant *A*. For readability reasons, we illustrate only the *sell product* abstract

service in the figure. The figure explains that if participant *A* provides *sell product* abstract service then participant *A* also provides the *obtain order*, *prepare products to deliver*, and *transfer invoice* business services.

3.2.2.3. GR.3: Dependency and MIS service (Resource → Dependency → Coordination service → MIS service)

This third group concerns the deductions of dependencies for both message and sequence flows, coordination services, and MIS services. The rules defined in this group are the most complicated in comparison to those of the other groups because they take into account several concepts at the same time.

Here, we present an example of the rules defined in this group. The rule shown in Fig.III. 23 allows us to deduce dependencies when two business services belonging to different participants have at least one resource in common. The concept of dependency of resource has been defined by [Malone et al., 2003] (Section 2.3.2.2 of Chapter 2). Coordination services can be deduced from dependencies by taking into consideration the exploitation of resources. [Crowston, 1994] affirmed the relation between dependency and coordination service, whereas [Touzi, 2007] supported the idea that the coordination service can be considered as the MIS service.

CNetwork(?a) ∧ hasRelationship(?a, ?z) ∧
P1(?z, ?y) ∧ provideBusinessService(?y, ?c) ∧ hasOutput(?c, ?d) ∧
P2(?z, ?x) ∧ provideBusinessService(?x, ?b) ∧ hasInput(?b, ?d) ∧
CoordinationService(?f) ∧ manipulateResource(?f, ?d) ∧
Dependency_between_BusinessServices(?e) →
fromBusinessService(?e, ?c) ∧ toBusinessService(?e, ?b) ∧ containResource(?e, ?d) ∧
isCoordinatedBy(?e, ?f) ∧ hasMISservice(?a, ?f) ∧ MISservice(?f)

Fig.III. 23 Deducing dependency, coordination service, and MIS service

This rule starts by finding a relationship between two participants via P1 (participant 1) and P2 (participant 2) relations. Each participant provides its own business services which have input and output resources. The rule verifies whether the output of a business service is the same as the input of another business service or not. If so, the rule finds a coordination service that can manipulates such a resource and creates dependency between these two business services. It also defines this coordination service as a MIS service.

The figure below shows the instances created respective to their corresponding concepts:

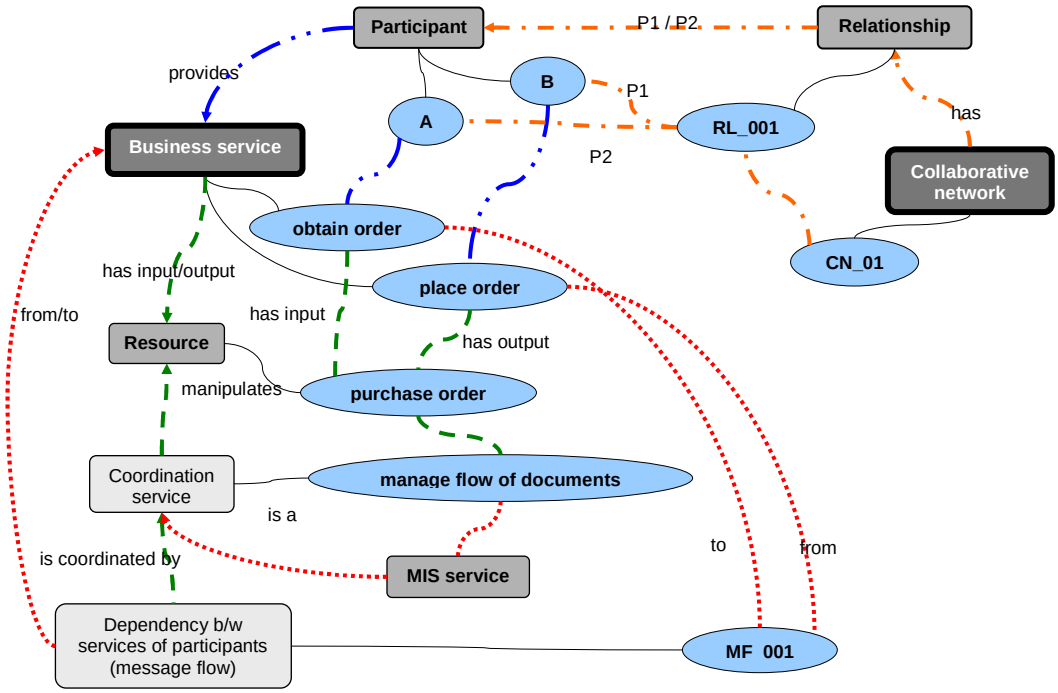


Fig.III. 24 Example of deduction by the rule in Fig.III. 23

This rule includes the concepts found in both CO and CPO ontologies. We still represent the instances and concepts in the same way as the two previous rules. The dash-dot lines have been defined by us since the beginning. The dash-dot-dot lines have been deduced from the second rule. The dashed lines already existed in the KB. The dotted lines are deduced from this current rule.

The example is continued from the results deduced by the previous rule. For readability purposes, we have kept only the *obtain order* business service provided by participant *A*. However, we add another participant, namely *B*, into the network since we are dealing with the dependency between business services belonging to different participants in the same network, *CN 01*. We assume that the participants *A* and *B* are related with a *RL 001* relationship. From the second rule, we obtained that the participants *A* and *B* provide *obtain order* and *place order* business services respectively. The *place order* service has a *purchase order* resource as output, while the *obtain order* service has the same resource as input. The current rule deduces a *MF 001* dependency of *purchase order* between these two business services. The *manage flow of document* coordination service can manipulate the *purchase order* resource. This coordination service is also created as the MIS service.

The rule is described by considering the direction of resource flows between business services. This means for each service we have to consider both its inputs and outputs. The rule shown in Fig.III. 23 is presented by the *dependency 1* in Fig.III. 25:

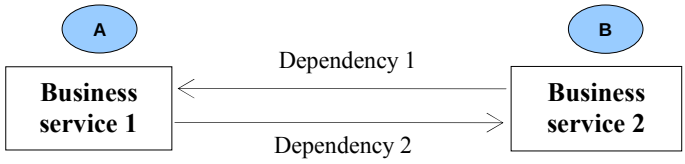


Fig.III. 25 Two-way dependency consideration

Therefore, there is another rule dealing with the direction of the *dependency 2*. The dependency between business services belonging to different participants is called message flow.

Another important rule in this group concerns the dependency between MIS services (sequence flow). Such a rule is also based on the same input-output concept as the dependency between business services (message flow). The description of this rule written in SWRL is shown in Annex A.

Finally, we can summarize that there are three rules defined in this group:

- Deduction of dependency between business services (Dependency 1)
- Deduction of dependency between business services (Dependency 2)
- Deduction of dependency between MIS services

3.2.2.4. GR.4: Common goal (Common goal → Abstract service)

This group is dedicated to deducing a list of abstract services to be included in the network. The abstract services deduced by the first rule are the ones that the involved partners provide to the others. They are a subset of the abstract services obtained by this actual rule. There is only one rule in this group. Fig.III. 26 shows the SWRL rule that derives abstract services from goal:

CommonGoal(?x) ∧ description(?x, ?a) ∧ swrlb:substringBefore(?y, ?a, " ") ∧ AbstractService(?b) ∧ name(?b, ?c) ∧ swrlb:containsIgnoreCase(?c, ?y) → achievesAService(?x, ?b)

Fig.III. 26 Deducing abstract service from common goal

The rule starts by segmenting the description of common goal and keeping only the first word found. Then the rule searches in the KB for abstract services whose name contains this word. The abstract services obtained are the services that all involved partners and the network itself have to provide.

We adopted the concept of goal from [Tawbi, 2001], which defines a goal as consisting of verb and parameters (e.g. profit, direction, result). However, the limitation of SWRL built-ins (identified using the prefix *swrlb*) does not allow us to analyse goal as discussed in the original concept since some built-ins have not yet implemented it. Consequently the rule has not yet been completed. Furthermore, there are restrictions in terms of expressing description of a common goal since it is required to start with a verb. The full implementation should take the whole phase of description into account and analyse every segment of it. Fig.III. 27 illustrates an example of deduction by this rule:

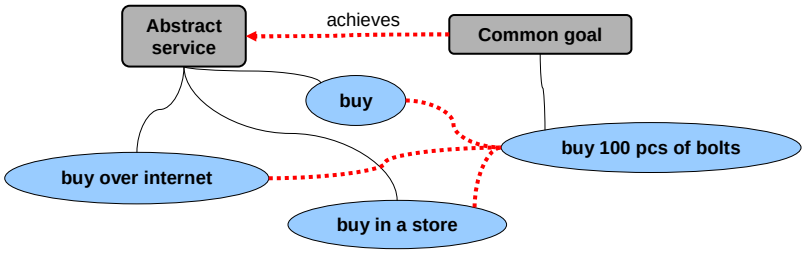


Fig.III. 27 Example of deduction by the rule in Fig.III. 26

We continue to represent the instances and concepts in the same way as in the previous rules. The figure shows that for the *buy 100 pcs of bolts* common goal, the rule deduces, for example, *buy*, *buy over internet*, and *buy in a store* abstract services. All of these abstract services contain the *buy* which is the first word of the description of the common goal.

3.2.2.5. GR.5: Topology (Power and Duration → Topology)

The rules in this group are dedicated to deducing the type of topology when the orientation of decision-making power and the duration of communications are provided. There are three rules defined in this group which are all shown in Fig.III. 28:

Topology(?x) ∧ hasPower(?x, central) ∧ hasDuration(?x, continuous) → hasType(?x, star)
Topology(?x) ∧ hasPower(?x, equal) ∧ hasDuration(?x, discontinuous) → hasType(?x, P2P)
Topology(?x) ∧ hasPower(?x, hierarchic) ∧ hasDuration(?x, continuous) → hasType(?x, chain)

Fig.III. 28 Deducing type of topology

These rules are specified on the basis of the characteristics of topology (chain, star, and P2P) [Katzy et al., 2003]. We summarized the characteristics of these three topologies in terms of decision-making power and duration in Table II.3 (Section 1.2.3 of Chapter 2). These three rules can be represented graphically as follows:

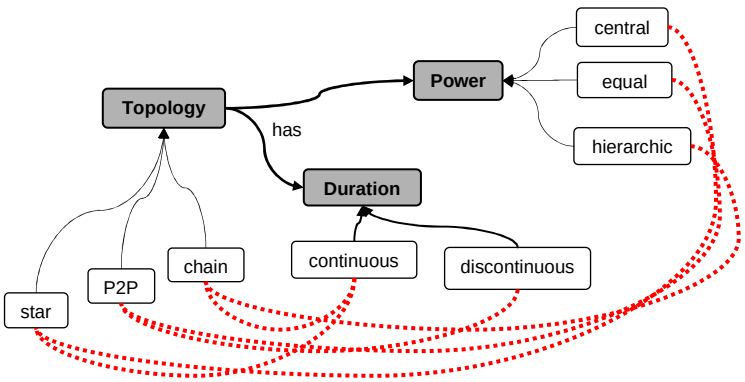


Fig.III. 29 Graphical representation of the rules in Fig.III. 28

The way we describe the rules in this group is different from the others. We specify the instances of concepts directly in the SWRL rules. If the concepts meet the instances defined, the rules return the instance result. We can describe the rules as below:

Topology is star if it has central power and continuous duration. Topology is P2P if it has equal power and discontinuous duration. Topology is chain if it has hierarchic power and continuous duration.

4. Conclusion of the chapter

Our knowledge-based system consists of three main parts: knowledge gathering, knowledge representation and reasoning, and collaborative process modelling.

In this chapter, we focused only on the second part. We developed an ontology-based approach by taking into account the mapping between collaborative network and process domains (Fig.II.10). This approach makes up the core part of our knowledge-based system. It is a process-oriented approach using ontologies and deduction rules to constitute a knowledge base. We realized that to move across these domains, we needed some additional knowledge coming from the Process Handbook. Such knowledge will be stored in the knowledge base.

We have developed an ontology called a Collaborative Network Ontology (CNO). The CNO is a domain ontology and classified as heavyweight ontology because it defines some restrictions and constraints (see Section 2.1.2 of Chapter 4). The CNO is composed of two ontologies (CO and CPO), and deduction rules establishing connections between these two ontologies. The CO represents collaboration features, while the CPO represents collaborative business process features. The ontologies and rules are written in OWL DL and SWRL respectively.

Now, we position the CO, CPO and rules on the mapping schema created at the end of Chapter 2 (Fig.II.10). The CO, CPO and deduction rules can be positioned respectively on the left, right, and links as shown below:

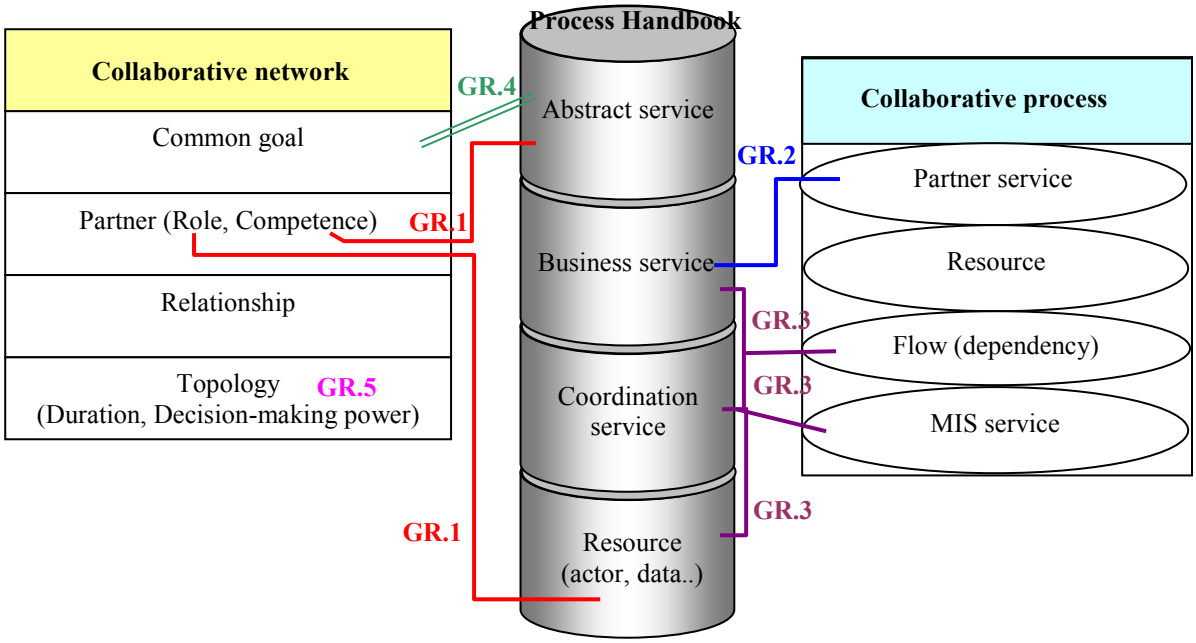


Fig.III. 30 CO, CPO and rules on the mapping between collaborative network and process worlds

The CO describes the elements of the collaborative network (left). The CPO describes the elements of the collaborative process (right). Five groups of rules (GR links) presented in the previous section establish the semantics and structural links between these two ontologies. These links replace the initial relations proposed originally in Fig.II.10. These links allow the knowledge-based system to use the knowledge in the collaborative network world and the instances (Process Handbook) in the knowledge base to derive new knowledge for the collaborative process world.

The other two parts of our knowledge-based system (Fig.III.1): knowledge gathering (left), and collaborative process modelling (right), will be presented in the next chapter. We will describe the essential functionalities of the whole system and its prototype. A demonstration of how this prototype works will be introduced in Chapter 5.

Chapter 4. Prototype Development

We stated in the previous chapter that our knowledge-based system (Fig III.1) is composed of three parts: knowledge gathering (left), knowledge representation and reasoning (middle), and collaborative process modelling (right). The main objective of Chapter 4 is to technically discuss the prototype of the whole system. However, the theoretical explanation of the middle part of the system which is required in order to be able to understand the prototype has already been done in the previous chapter. The left and right parts of the system will be explained technically in this chapter.

In this chapter, we will first of all present a brief introduction to the prototype including its objective, and functionalities. Then, the technical architecture of the prototype will be discussed, together with the knowledge gathering (left) and the collaborative process modelling (right). The principal components and some other related development technologies will be presented.

1. Presentation of the prototype

1.1. Objective

The prototype is aimed at demonstrating the possibility of using tools to enable the transition from knowledge on collaboration to a collaborative process model. Another expectation of the prototype is to be able to automate this transition process as much as possible

According to Fig.I.3, our knowledge-based system relies on the business level of the MIS design concept. The output of this system will be used to generate the SOA logical architecture of the MIS (logic model of MDE). As discussed in Chapter 1, the principal constraints and hypotheses that impact the development of this system mainly concern the succession of the model from the CIM to PIM levels. This model is the BPMN collaborative process model that our system produces. The BPMN model becomes the important factor that we need to take into account when developing the prototype in order to guarantee the interoperability between the CIM and PIM levels.

1.2. Functionalities

Our knowledge-based system consists of three main parts: knowledge gathering, knowledge representation and reasoning, and collaborative process modelling. The first part concerns the acquisition of implicit knowledge expressed by business partners to characterize the behaviours of a collaborative network. The knowledge representation and reasoning part is based on ontologies and deduction rules as discussed in Chapter 3. This part is the heart of the system since it offers the possibilities to morph knowledge about collaboration into a collaborative process. Finally, the collaborative process modelling part concerns the

extraction of the collaborative process knowledge, and the construction of a collaborative process model from the extracted knowledge.

Fig. IV. 1 shows the big picture of our knowledge-based system, the principal functionalities related to the three parts of the system, and the movement of knowledge inside the system:

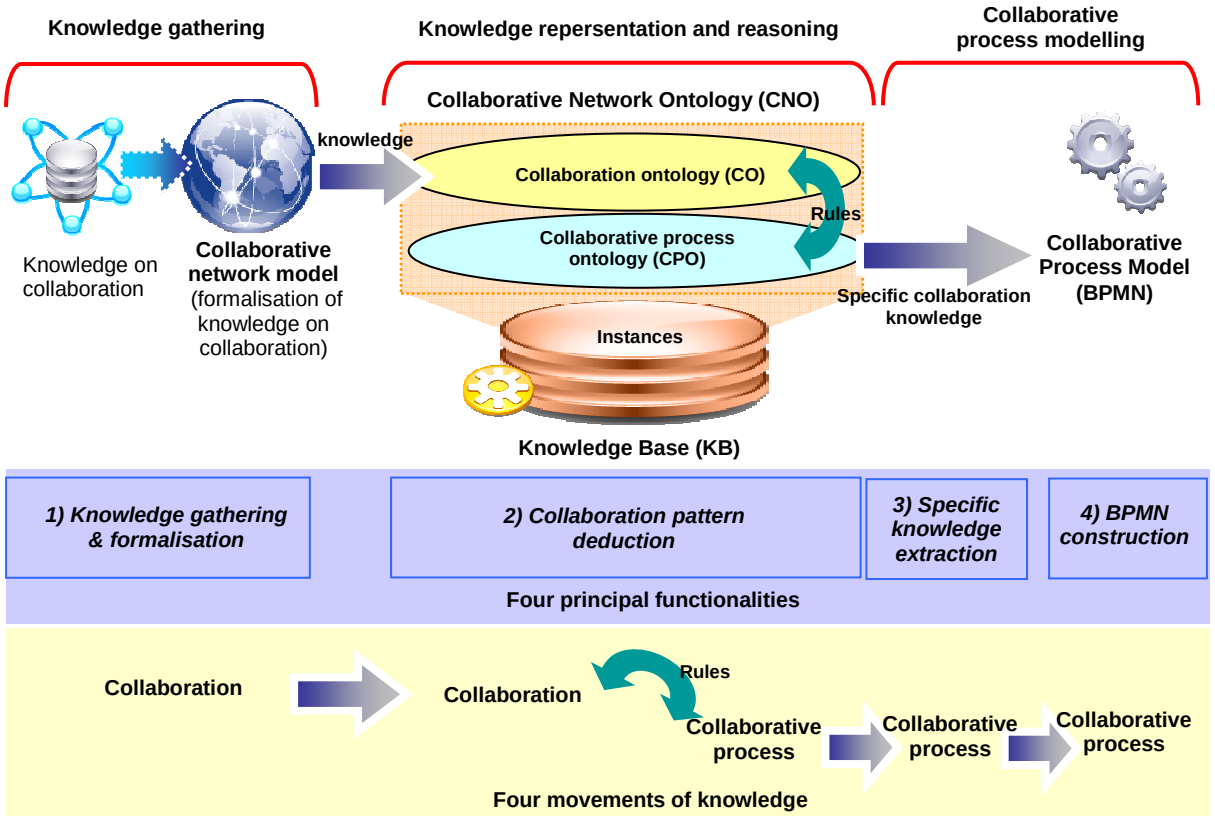


Fig. IV. 1. Big picture of the development concept including four functionalities and movements of knowledge

The prototype requires the full implementation of these four principal functionalities:

- **Knowledge gathering and formalization** functionality focuses on capturing all necessary knowledge concerning the collaboration that is going to take place. Users of this functionality are in charge of collecting that knowledge by interviewing the partners of a studied network. This knowledge concerns the network's characteristics (relationships between each pair of participants, and common goals), and participants' details (roles, and services). From this knowledge, the users can design relevant collaborative network models.

- **Collaboration pattern deduction** functionality supports the ontology-based approach for representing and reasoning knowledge (presented in Chapter 3). It takes as input the collaborative network model defined from the first functionality and automatically deduces new knowledge as output. The deduced knowledge concerns globally the

knowledge about the collaborative process, such as business services, resource dependencies, and collaborative services.

- **Specific collaborative process extraction and visualization** functionality is a sub part of the collaborative process modelling. This functionality consists of four actions: 1) extraction of knowledge, 2) representation of the queried knowledge in the form of collaborative process, 3) verification of the collaborative process generated from the previous action with the involved partners (manually deleting the useless objects from the process), and 4) generation of a new complete collaborative process composed only of relevant objects, gateways, and events. The partners have to agree on this complete collaborative process model before passing it through the BPMN construction. The compliance of the BPMN process models between business and logic levels (Fig I.3) is ensured by this functionality, not the BPMN construction functionality. Thus, the collaborative process models obtained from this functionality do totally conform to the meta-model of collaborative process (Fig.II.6) defined by [Touzi, 2007] in order to have the appropriate BPMN process models at the end of the fourth functionality.

- **BPMN construction** functionality is another sub part of the collaborative process modelling. This last functionality focuses on representing the collaborative process model obtained from the previous functionality in the form of a BPMN model. It concerns the transformation of collaborative process models into a BPMN relevant one. [BPMS Watch] mentioned BPMN as an important factor driving business-IT alignment since it is intuitive enough to be used by business analysts, yet rich enough to generate powerful service-oriented implementations.

We can see the movements of knowledge between functionalities and even inside the functionality itself. The most important one is the one inside the second functionality. This movement concerns the transformation of knowledge from collaboration into the collaborative process. This transformation is done by the ontology-based approach through deductions. These deductions make it possible to morph the collaboration knowledge (in CO) into the collaborative process knowledge (in CPO) (see Chapter 3 for more detail). The other movements only concern the transformation within the same domain of knowledge. Such movements will be presented in Section 2.2.3 of this chapter.

2. Technical architecture

The four functionalities listed previously are what the prototype should be able to provide in order to attain the objective. We developed and implemented some tools to support these functionalities, as shown below:

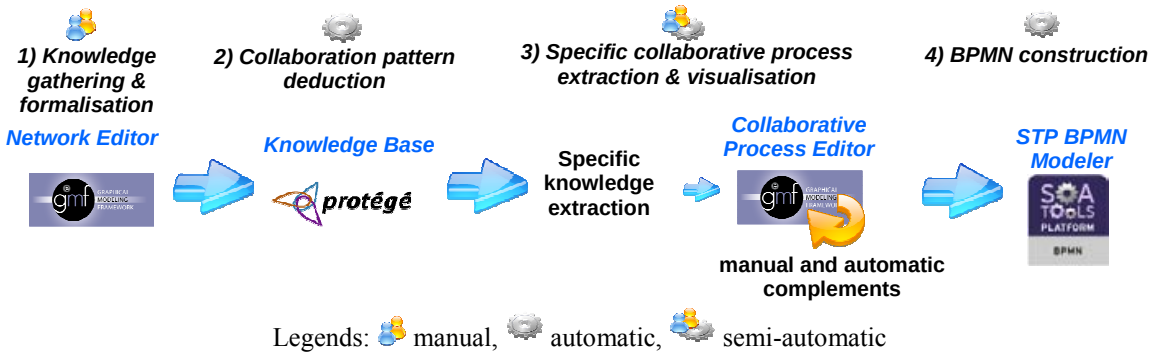


Fig. IV.2. Four functionalities of the prototype and development technologies

The figure above represents the technical architecture of the prototype. The prototype has been developed on the Eclipse platform with Java application. It is mainly composed of four components which support four different functionalities:

- Network Editor (NE) supporting the knowledge gathering and formalization functionality.
- Knowledge Base (KB) supporting the collaboration pattern deduction functionality.
- Collaborative Process Editor (CPE) supporting only the collaborative process visualization of the third functionality.
- STP BPMN Modeller supporting the BPMN construction and visualization.

Besides the main components, there are also some complementary concepts and technologies that complete the construction of the whole prototype:

- The technologies for extracting specific knowledge (query) which supports the specific knowledge extraction of the third functionality.
- The technologies for connecting the four principal components together (transformation of models).
- The complementary concept for completing the collaborative process model generated from the CPE: generation of gateways, events, etc.

In this section, we will first discuss the principal components, along with their development concept, and technology. Secondly, the complementary technologies will be presented.
.

2.1. Principal components

The principal components are Network Editor, Knowledge Base, Collaborative Process Editor, and STP BPMN Modeller. These components work consecutively which means that the output of the preceding one is the input of the following one. For each component, we will talk about the goals and scope of development, technology, and functionalities.

2.1.1. Network Editor

2.1.1.1. Goals and scope

The main reason for developing the Network Editor (NE) is to support the knowledge gathering and formalization functionality for collecting the essential knowledge, and modelling the collaborative network. The following figure shows the use of NE in the prototype:

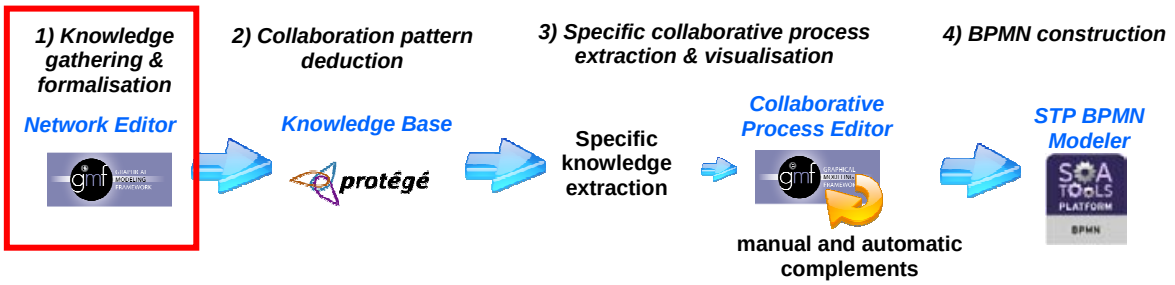


Fig. IV. 3 Use of Network Editor in the prototype

The idea is to provide an aided design tool in the collaborative network domain. The NE should provide a sort of design space with some tools which allow users to create, and characterize their collaborative networks in a graphic way. This editor requires some expertise, as well as effective and efficient communication between the NE's user and the network business partners.

The input of this tool is knowledge on collaboration expressed by all the partners involved in a studied network. The output is a collaborative network model describing the collaboration characteristics, including partners' details. This output will be imported into the Knowledge Base as a new instance. Thus, the NE should somehow be connected to the Knowledge Base.

2.1.1.2. Selection of technology

One of the first difficulties regarding the development is to select a competent technology. This technology should meet the above goals which address the Domain-Specific Modelling (DSM). DSM is a software engineering methodology for designing and developing particularly IT systems. It involves the systematic use of a graphical domain-specific language (DSL) to represent the various facets of a system. Interest in DSM has grown tremendously with a lot of DSM editors coming out over the last few years, for instance, EMF, GEF, GMF [GMF], and MetaEdit+ [MetaCase]. MetaEdit+ is eliminated from the list since it is not open source, neither can it be used with Eclipse. The other three are based on the Generic Eclipse Modelling System (GEMS). The GEMS is an open source project that has been developed in conjunction with Siemens CTSE2, IBM, and Prismtech. It is able to generate the implementation of a graphical modelling tool from a visual language specification (meta-model).

From [White et al., 2007], the EMF (Eclipse Modelling Framework) contains various APIs (Application Programming Interface) for building object graphs for a model, serialising and de-serialising object graphs, and enforcing basic constraints on the graphs. The GEF (Graphical Editor Framework) provides inversion of control for loading a graphical model, creating controllers for each model element, and constructing and associating views with the controllers. The GMF (Graphical Modelling Framework) allows developers to develop

complex controller logic from specifications that map a model to the different elements of a view.

Finally the GMF seems to be the best choice since it is based on both EMF and GEF. It is tending to become a keystone framework for the rapid development of standardized Eclipse graphical modelling editors.

2.1.1.3. Development concept

We start this section by giving a brief overview about GMF in order to understand how it works. Then we will go into the detail of the development by presenting the meta-model and the graphical definition model that our editor is based on.

a. Overview of GMF

GMF uses both EMF and GEF for building Eclipse-based functionality. It is divided into two main components: runtime and tooling used to generate editors capable of leveraging the runtime. Fig. IV. 4 shows the dependencies between the generated graphical editor, the GMF Runtime, EMF, GEF, and the Eclipse platform:

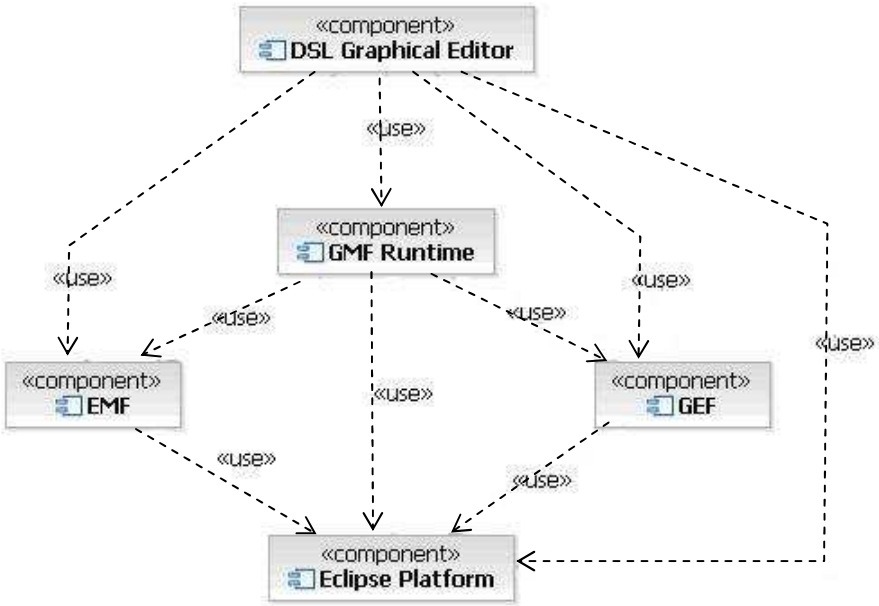


Fig. IV. 4. Dependencies between the generated graphical editor, GMF runtime, EMF, GEF and Eclipse platform [Plante, 2006]

The development of a GMF-based editor requires these following core models:

- The Domain model is where we define concepts, attributes, and relations between concepts. It is a meta-model describing a domain of interest. It is the first model to be developed. However, this is not mandatory as the diagram definition is maintained separate from the domain.

- The Graphical definition model contains information related to the graphical elements that will appear in a GEF-based runtime. However, this model does not have any connection to the domain model for which it will provide the representation and editing.

- The Tooling definition model is used to design the palette and other peripherals (menus, toolbars, etc.).

Once these three models have been defined, we have to create a mapping model in which we specify the relations between these three models. After that, the GMF provides a generator model that allows defining the implementation details for the generation phase. An editor plug-in based on the generator model automatically generates a runtime model diagram. Fig. IV. 5 illustrates the main components and models used during GMF-based development:

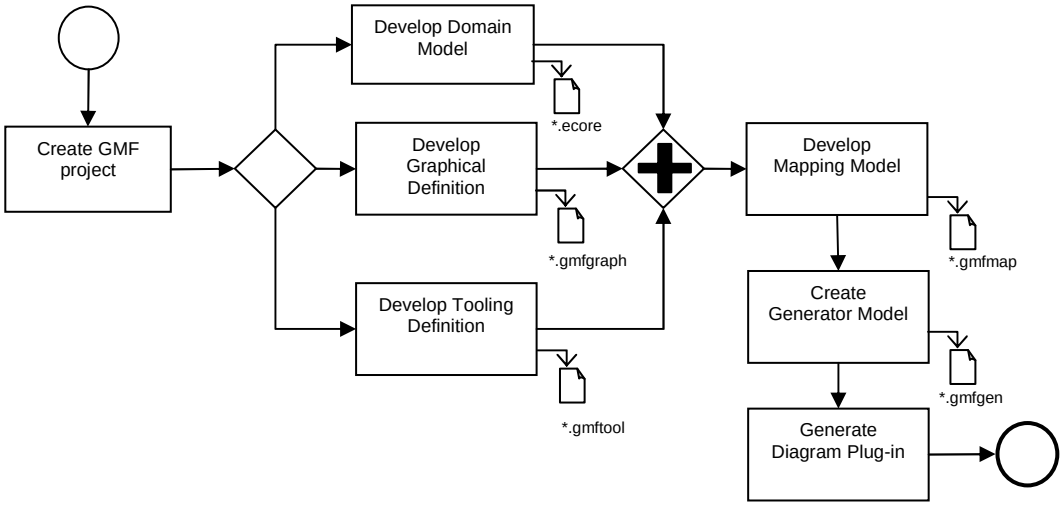


Fig. IV. 5. Main components and models used during GMF-based development[15]

b. Meta-modelling

The NE is aimed at facilitating users to collect and formalize knowledge about collaboration. Its domain model should describe the collaboration context. In addition, an output model of the NE has to be imported into the KB. The concepts and attributes defined in this domain model should also correspond to those of KB. Therefore, the domain model mostly relates to the CO (corresponding to the collaborative network side of Fig.II.10) from which we can find the modelling elements of collaborative network. Fig. IV. 6 illustrates the domain model (meta-model) of the NE:

[15] http://wiki.eclipse.org/index.php/GMF_Tutorial

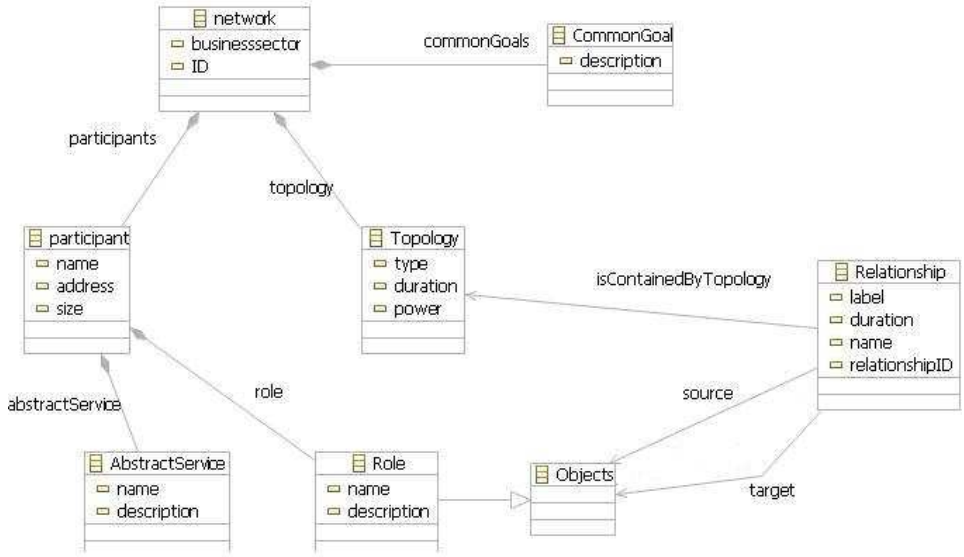


Fig. IV. 6. Domain model of NE (diagram view)

From the above figure, a network is composed of several participants, topologies, and common goals. Each participant is composed of several services, and roles. Topology contains the relationship which links two participants together at the role level. The appropriate attributes are also defined for each concept. They can be character string, numeric value, Boolean, or enumeration. For example, the *Role* concept has *name* and *description* attributes defined as enumeration, and string respectively. The enumeration literals of the role name attribute are retrieved from the instances of the role concept stored in the KB. Fig. IV. 7 shows the enumeration literals related to role name, power type, and duration type:

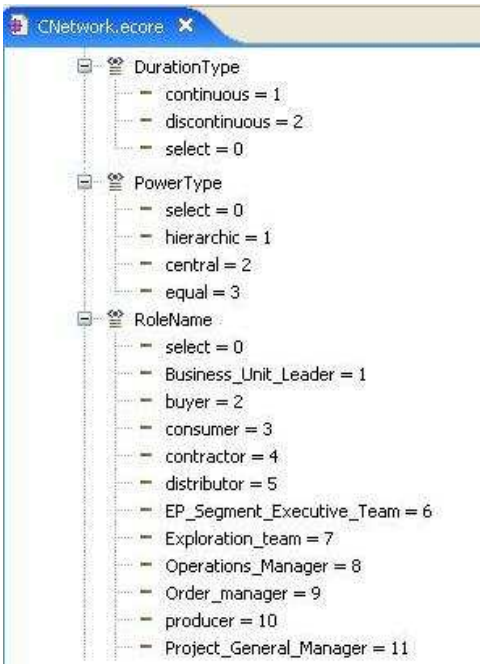


Fig. IV. 7. Enumeration classes in the domain model (Ecore view)

c. Graphical and tooling definition models

A graphical definition model is used to define the figures, nodes, diagrams, connections, etc. that will be displayed on the diagram. From the domain model discussed in the previous section, the graphical description for each concept is defined as shown in Fig. IV.8, for example:

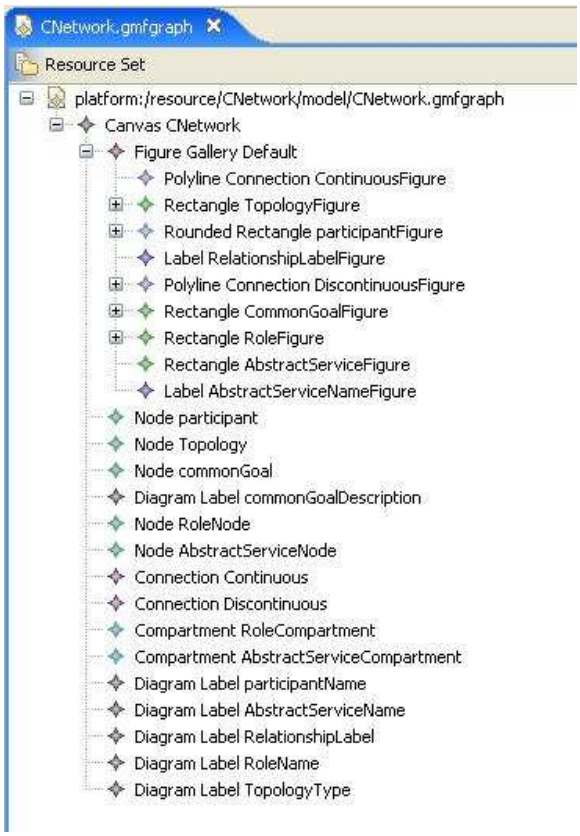


Fig. IV.8. Graphical definition model of NE

- participant as node in yellow rounded rectangle figure
- role, topology, and common goal as node in rectangle figure
- abstract service as node in text label figure
- continuous relationship as connection in full line figure
- discontinuous relationship as connection in dashed line figure

The illustration of these elements is shown in Fig.IV.10.

To be able to create the above concepts graphically we need a tooling definition model. We classify the tools under two groups: tools for creating participants' details, and for creating the characteristics of the collaborative network. Fig.IV.9 shows the tooling definition model of the NE:

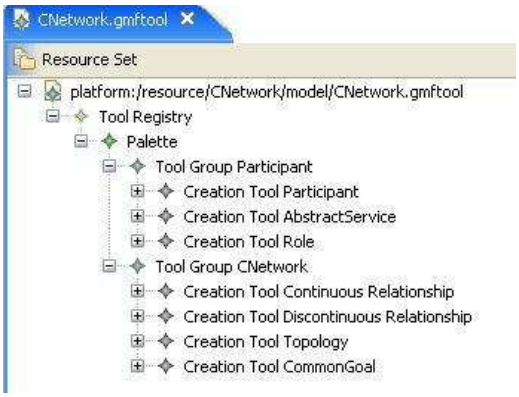


Fig. IV. 9 Tooling definition model of NE

These two models will be used to generate the runtime diagram of NE. We show in the following section the runtime diagram.

2.1.1.4.Functionalities

The aim of NE is to be an aided design editor for collaborative networks. The runtime diagram of NE consists of three main parts as follows:

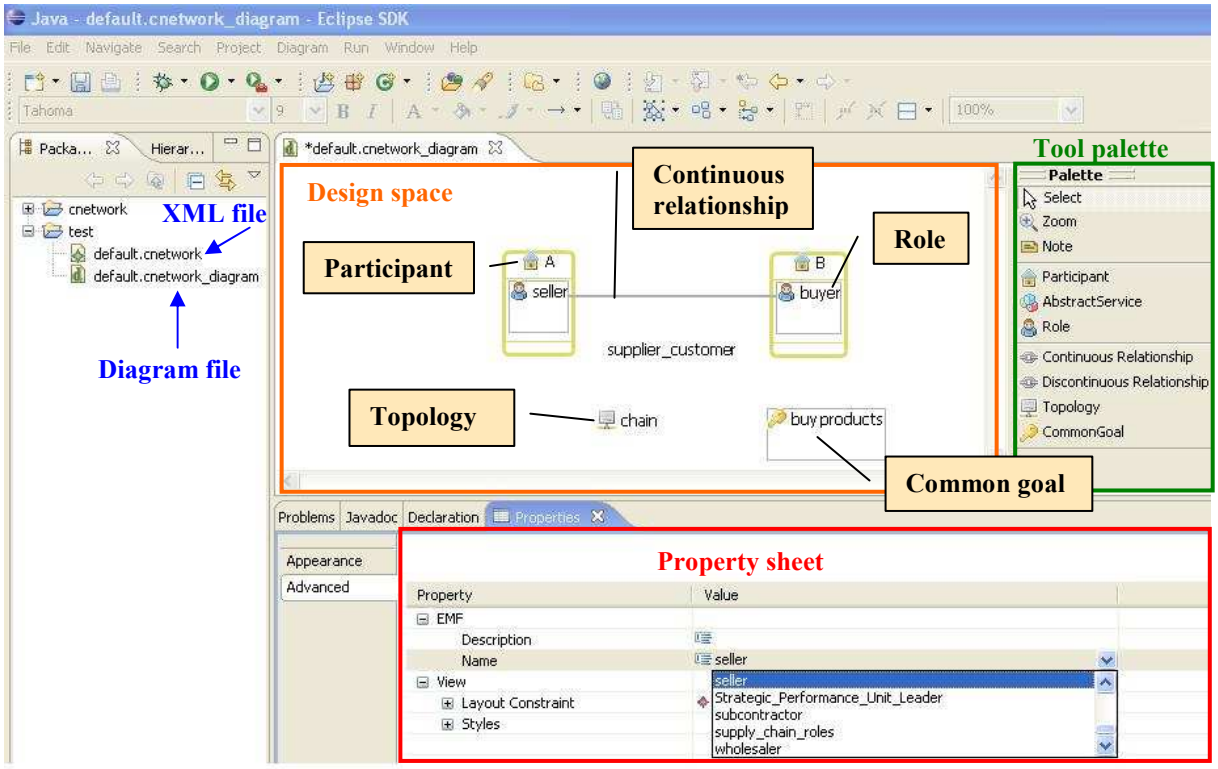


Fig. IV. 10. Runtime diagram of the NE

- **Tool palette:** a set of tools allowing users to create elements on the design space. This tool palette is generated with respect to the tooling definition model defined earlier. The tools that the NE offers are participant, abstract service, role, continuous and

discontinuous relationships, topology, and common goal. These are essential for defining a collaborative network.

- **Design space:** an empty space for drawing a collaborative network diagram by using the tools in the palette. The graphics displayed on this space are defined in the graphical definition model and associated with the tooling definition model. We use this space to graphically characterize the collaborative network through its participants, abstract services or roles of each participant, relationship between participants, topology, and common goals to be achieved.

- **Property sheet:** a set of attributes that we have to define when creating an element. During the development of the domain model, we specified what attributes we need for each element. The attributes are for example name, description, type, etc. The value of each attribute can be specified by selecting from a given list (if it is enumeration) or filling in from scratch. According to Fig.IV.10, the NE shows the *name* and *description* attributes of a role element. For the name attribute, we have to select a role from the given list, while for the description attribute we have to add it ourselves.

The creation of a new collaborative network provides not only a diagram file (*.cnetwork_diagram) representing the network graphically, but also its associated XML file (*.cnetwork) which conforms to the NE's meta-model. The XML file obtained will be used for manipulation afterwards (for transforming and importing into the Knowledge Base as a new instance).

2.1.2. Knowledge Base

2.1.2.1. Goals and scope

The main reason for developing the Knowledge Base (KB) is to support the collaboration pattern deduction functionality for executing deduction rules stored in it. Another interesting point is to visualize the ontology, and store instances concerning collaboration and services. The following figure shows the use of KB in the prototype:

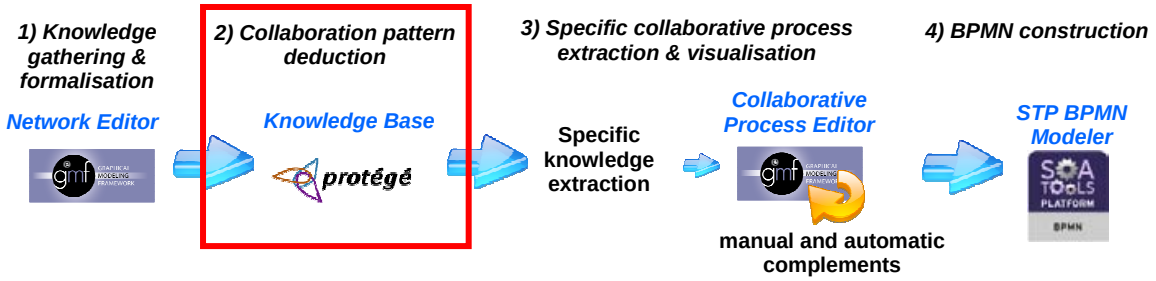


Fig. IV. 11 Use of Knowledge Base in the prototype

The knowledge base can be categorized into machine-readable and human-readable knowledge bases. As the purpose of constituting the KB is to automate deductive reasoning, our KB should be machine-readable. It should be kept up-to-date, and carefully designed in

content and structure. An ontology may be used to specify the structure (concepts and properties) of KB. This means that an ontology, together with a set of instances (individuals) and concepts (classes) constitute the KB.

Our KB requires a collaborative network model defined by the NE as input. After having done the deduction, new knowledge will be asserted in the KB automatically.

2.1.2.2. Selection of technology

According to the above discussion, to construct a knowledge base, we need an ontology to specify its structure, and some instances to be stored in it. The ontology used for constituting our KB is the CNO conceptually presented in the previous chapter.

In Chapter 3, we already selected the OWL DL and SWRL as languages for representing the ontologies (CO and CPO) and deduction rules respectively. Thus, here we have to select a technology that can edit OWL DL and SWRL.

There are a number of technologies available in the market for editing OWL, for instance KAON[16], Protégé [Noy et al., 2001], topBraid[17]. These three tools are integrated tool suites which have an extensible architecture, and whose knowledge model is usually independent of an ontology language [Gomez-Perez, 2004]. These tools provide a core set of ontology related services and are easily extended with other modules to provide more functions. The complete analysis of these three tools is described in Annex G.

We found that Protégé is widely used and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development. [Bouslimi et al., 2008] used Protégé to edit OWL ontology and SWRL rules in order to share and reason about information in a semantic web context. Protégé supports several inference engines. According to Fig. IV. 11, we require a tool that can interoperate with the other tools in our prototype, especially the GMF technology that we use to develop the NE. Thus, the import and export capabilities are also important criteria for us. Protégé seems to be a better choice than the others because it can deal with XML and XMI formats which are the main formats we use in interoperating between the different tools in the prototype. Furthermore, Protégé provides SWRL Tab (Editor) as a plug-in for editing and executing SWRL rules.

2.1.2.3. Development concept

To constitute the KB, we use the CNO defined in Chapter 3. The CNO is an OWL-based ontology. An OWL ontology has similar components to Protégé. Here the terminologies used to describe these components are Classes (concepts), Properties, Individuals (instances), and Rules.

[16] http://sourceforge.net/projects/kaon
[17] http://www.topquadrant.com/topbraid/composer/index.html

a. Individuals

One of the most important aspects of a knowledge base is the quality of information it contains. The information we will store in our KB is called individual. Individuals represent objects in the domain that we are interested in. So individuals are also known as instances that are referred to as being instances of classes. In OWL, it must be explicitly stated that individuals are the same as each other, or different to each other. Otherwise they might be the same as each other, or they might be different from each other.

In our case, the individuals we originally store in our KB come from the dataset [PH-OWL] which is an OWLized version of the MIT Process Handbook because the CNO that constitutes the KB is mostly based on the Process Handbook ontology. This dataset provides approximately 5000 instances of processes, goals and resources including roles. These instances are generic and can be used to define many kinds of processes. These individuals are stored in their corresponding classes and properties. Fig. IV. 12 illustrates an example of instances stored in the class *Abstract service*:

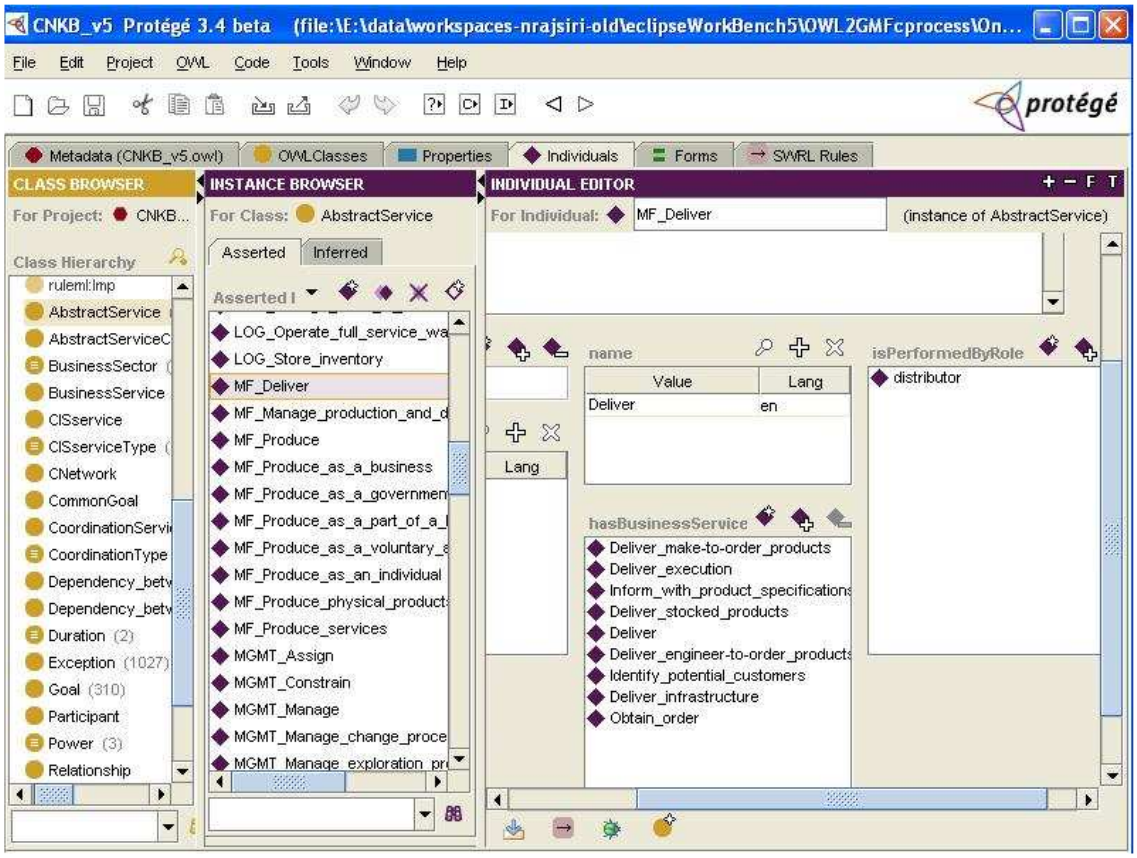


Fig. IV. 12. Instances of the class Abstract service

b. Properties

Properties in the Protégé can be defined as binary relations, and as data type (e.g. character string, numeric value, boolean, or enumeration). They are known as associations (binary relations) and attributes (data type) in UML.

Properties as relations link two individuals together. Properties may have a domain and a range specified. They link individuals of the domain to individuals of the range. For example, the property *playRole* might link the individual *Enterprise AB* to the individual *Seller*, or the property *provideBusinessService* might link the individual *Enterprise AB* to the individual *Obtain order*. Properties can also have inverses. They can be limited to having a single value (to being functional). For example, the property *hasPower* of the *Topology* class, a topology can have only one decision making power type. They can also be either transitive or symmetric.

The Fig. IV. 13 shows that the property *hasCommonGoal* would probably link individuals belonging to the class *CNetwork* to individuals belonging to the class *CommonGoal*, and the inverse of *hasCommonGoal* is *isAchievedByNetwork*:

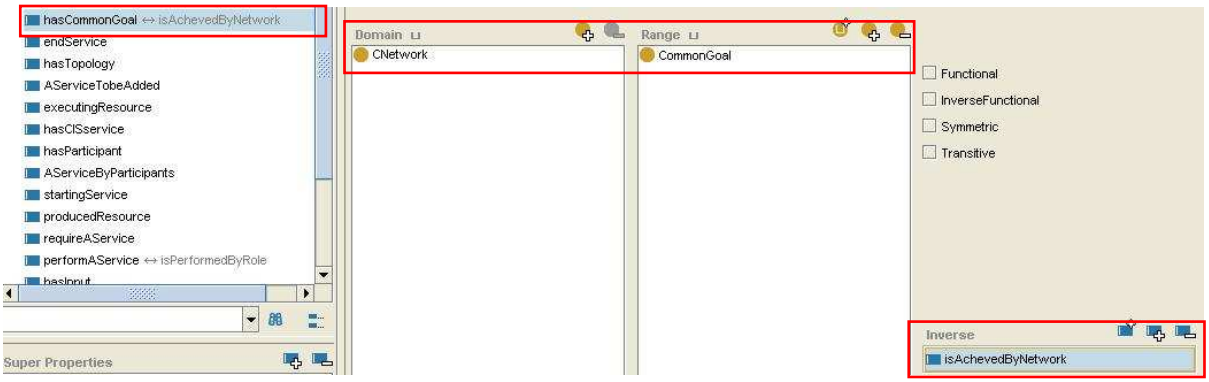


Fig. IV. 13. *hasCommonGoal* property defining the relation between *Collaborative Network* and *Common Goal* classes

Properties like data types are for defining attributes and their format (e.g. string, float, int, boolean, etc.) in individuals. Fig. IV. 14 shows an example of data type *name* whose domain refers to the classes of *Participant, Resource, Role*, etc. and whose range refers to the *string* format:

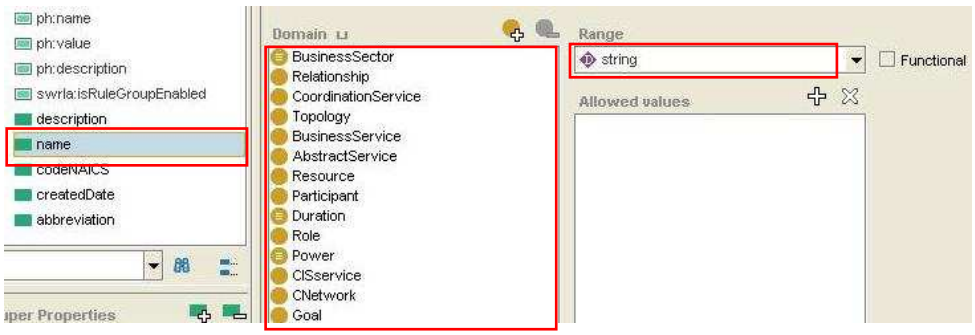


Fig. IV. 14 *name* property defining as string

c. Classes

OWL classes are interpreted as sets that contain individuals. They are described using formal descriptions that state precisely the requirements for membership of the class. For example, the class *Participant* would contain all the individuals that are actors of collaborative networks.

Classes may be organized into a superclass-subclass hierarchy, which is also known as taxonomy. Subclasses specialize (are subsumed by) their superclasses. For example, considering the classes *Topology* and *Chain*, *Chain* might be a subclass of *Topology* (so *Topology* is the superclass of *Chain*). This can be interpreted as follows: all chains are topologies, all members of the class *Chain* are members of the class *Topology*, being a *Chain* implies that it is a *Topology*, and *Chain* is subsumed by *Topology*. One of the key features of OWL DL is that these superclass-subclass relationships can be computed automatically by a reasoner. Fig. IV. 15 shows the class Topology and its subclasses (Chain, Star, and P2P):

Fig. IV. 15. Topology class and its subclasses

In OWL, properties can be used to create restrictions to constrain individuals. Restrictions in OWL fall into three main categories: Quantifier Restrictions (‘∀’ only, ‘∃’ some), Cardinality Restrictions (‘=’ cardinality, ‘≤’ maxCardinality, ‘≥’ minCardinality), and has Value Restrictions (‘ɜ’ has). From Fig. IV. 15, the ‘∀’ symbol is translated into *only*. This symbol constrains the relationships along a given property to individuals that are members of a specific class. For example, *hasPower only Power* describes the individuals all of whose *hasPower* relationships are with members of the class *Power*.

To write restrictions, [Horridge et al., 2004] discussed the difference between *Necessary* and *Necessary & Sufficient* conditions. If class A is described using necessary conditions, then we can say that if an individual is a member of class A it must satisfy the conditions. We cannot say that any individual that satisfies these conditions must be a member of class A. However, if class A is now defined using necessary and sufficient conditions, we can say that if an individual is a member of the class A it must satisfy the conditions and we can now say that if any individual satisfies these conditions then it must be a member of class A. The conditions are not only necessary for membership of A but also sufficient to determine that something satisfying these conditions is a member of A.

The restrictions for the class *Chain* are displayed as shown in Fig. IV. 15. According to the definitions above, the restrictions are defined as necessary and sufficient conditions. Any

individual having *continuous* as duration and *hierarchic* as power is a member of this class *Chain*.

d. Rules

Besides individuals, properties, and classes, we also need to formalize the deduction rules with SWRL Editor. As such rules ensure the interactions between classes of the CNO, they should be created as a part of ontology in the same way as individuals, properties, or classes [O'Connor et al., 2005]. The deduction rules are really important in our knowledge-based system because it ensures the morphing of the collaboration knowledge (in CO) into the collaborative process knowledge (in CPO). Five groups of rules have been defined and discussed in Section 3.2.2 of Chapter 3. The SWRL Editor has been used to edit these rules. The following figure shows the interactive interface of SWRL Editor and an example of SWRL rule (GR.1 Rule 2 shown in Annex A):

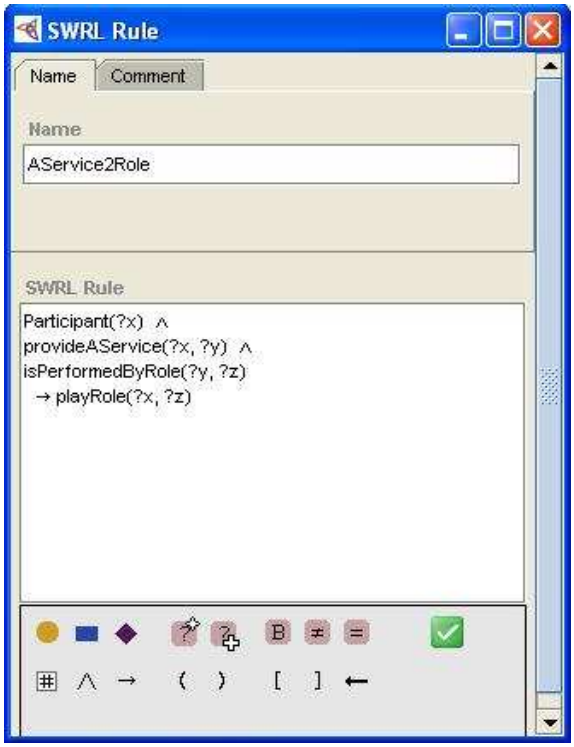


Fig. IV. 16. Editing a rule with SWRL Editor

The execution of SWRL rules is described in Annex F.

2.1.3. Collaborative Process Editor

2.1.3.1. Goals and scope

The main reason for developing the Collaborative Process Editor (CPE) is to support the collaborative process visualization of the third functionality. The main idea of developing this CPE is to provide a computer-aided design tool in the collaborative process domain in the same way as the NE. The use of CPE in the prototype is shown as follows:

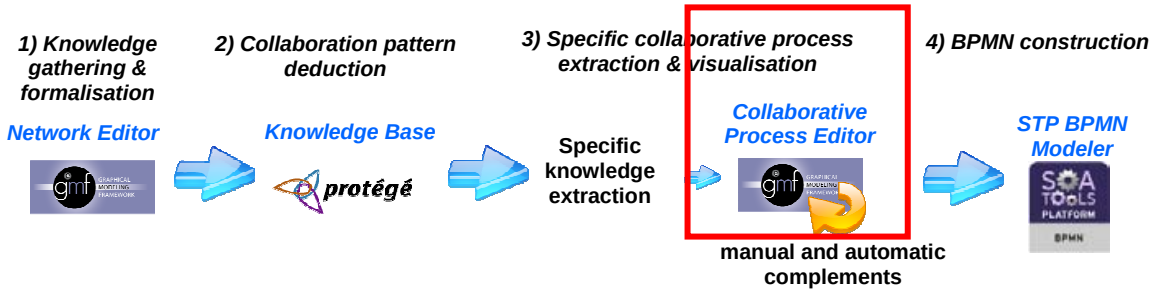


Fig. IV. 17 Use of Collaborative Process Editor in the prototype

The extraction capability of the third functionality is supported by SPARQL query language that we will talk about in Section 2.2.1.

The CPE is similar to the NE, only the tool palette is different. The tool palette of the CPE provides not only the same creation tools as the NE, but it also provides other additional tools allowing the user to create the elements of the collaborative process. For example, tools for creating services, flows, gateways, etc. The CPE offers users the possibility of visualising the knowledge about the collaborative process extracted by SPARQL queries. Any modifications are allowed but have to be done with the agreement and satisfaction of the partners of the studied network. Therefore this editor requires efficient and effective communication between the users and the business partners of the network.

After SPARQL queries have extracted only the knowledge concerning a studied network, the CPE will organize that knowledge and represent it in the form of a collaborative process model. The output of this tool is a collaborative process model validated by all involved partners and ready to be transformed into the BPMN model.

The essential requirement of the CPE is to ensure the succession of the BPMN models from the business level to the logic level (Fig.I.3). The collaborative process model obtained from the CPE totally conforms to the meta-model of a collaborative process (Fig.II.6) defined by Touzi [Touzi, 2007]. This guarantees that we will have a good BPMN process model at the end of the BPMN construction functionality.

2.1.3.2. Development concept

According to the above goals, we realize that the CPE is also related to the DSM as same as the NE. However the CPE is a GMF-based development.

a. Meta-modelling

The CPE is designed to be an aided design tool which can visualize and edit collaborative processes. Its domain model should describe the elements of the collaborative process domain (defined in the right side of Fig.II.10). In addition, a collaborative process visualized with this tool comes from the knowledge extracted from the KB. Thus, the domain model is strongly related to the CNO that constitutes the KB. It adopts the meta-model of NE and adds the

concepts related to the domain of the collaborative process, such as resource, business service, dependency, MIS service, gateway, and event.

As several concepts of the meta-model of collaborative process are already taken into account when defining the CNO, we are pretty sure that the collaborative process models obtained from the CPE conform more or less to the model required for generating the logic architecture of the MIS (Section 2.2.1 of Chapter 1). However, the CNO does not have every concept defined in the meta-model of collaborative process. It lacks the concepts of gateway and event which are both important elements of the BPMN process. So, we also now add these two concepts in the domain model of the CPE in order to make the generated collaborative processes more oriented to BPMN process. Fig. IV. 18 presents the domain model of the CPE:

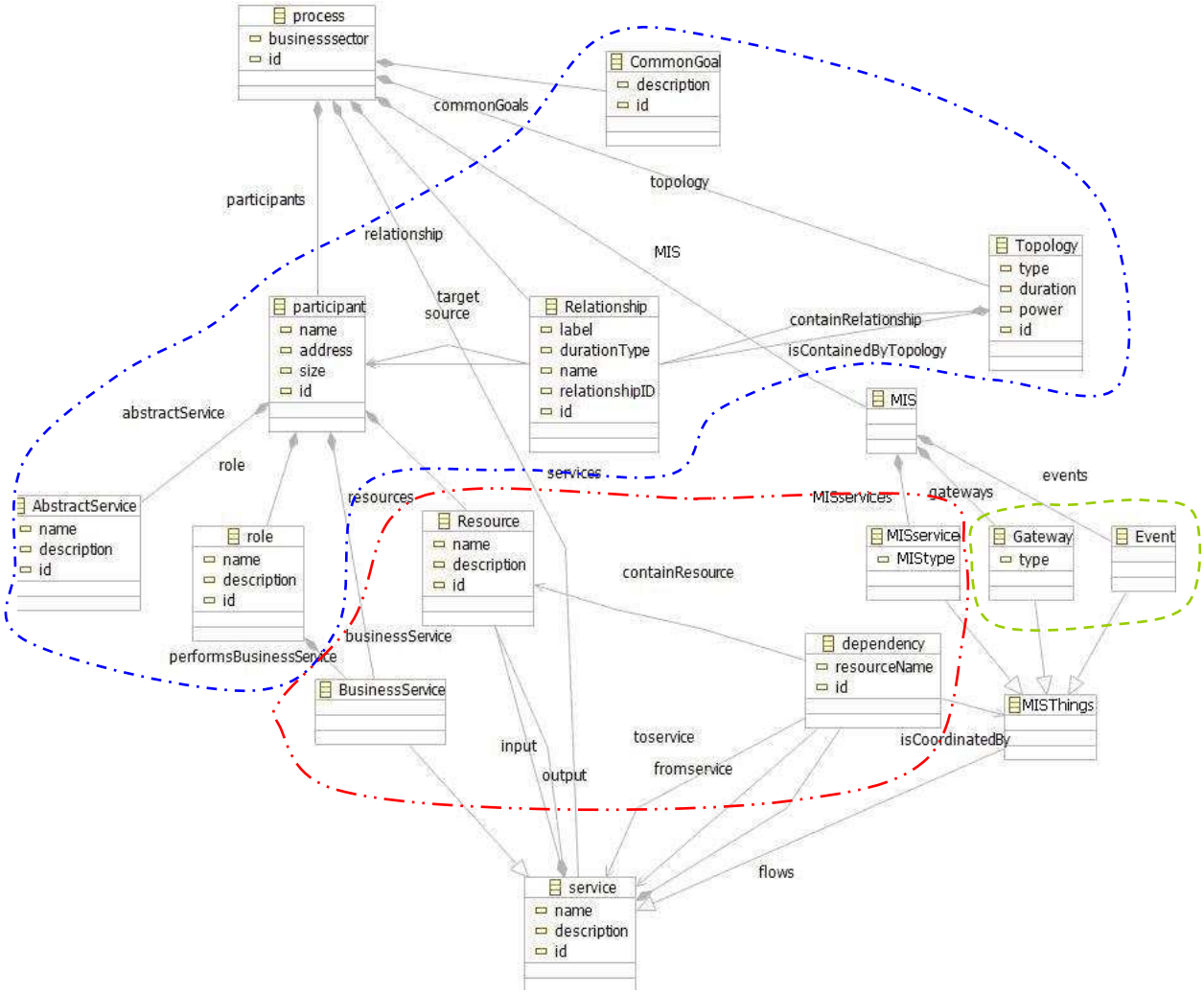


Fig. IV. 18. Domain model of CPE (diagram view)

From the above figure, we can find the concepts already defined in the meta-model of NE (dash-dot line). These concepts principally concern the characteristics of the collaborative network, including participants. The others are the new concepts about service related to the collaborative process. We mainly extract these new concepts from the CPO (dash-dot-dot

line). We also add some additional concepts from the meta-model of collaborative process (dashed line) which are not yet included in the CPO in order to be more BPMN-oriented. We can describe such new concepts as follows: a dependency links together two services which can be business services or MIS things (MIS service, gateway, and event). Every dependency contains resources and is coordinated by a MIS service. A process can have only one MIS which itself is composed of MIS elements.

b. Graphical and tooling definition model

A graphical definition model is used to define the figures, nodes, diagrams, connections, etc. that will be displayed on the diagram. As the domain model of CPE integrates the one of NE, its graphical definition model is roughly the same as the NE's. From the domain model discussed in the previous section, the concepts are defined as shown in Fig. IV.19:

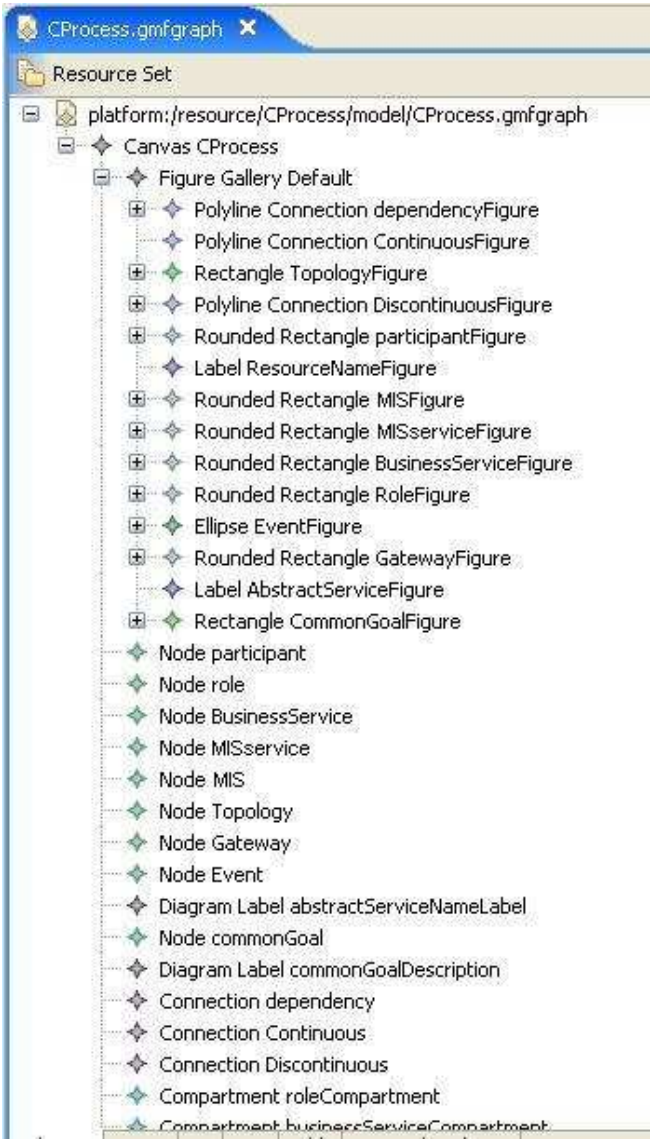


Fig. IV.19. Graphical definition model of CPE

- participant, role, MIS, MIS service, and business service as node in rounded rectangle figure
- topology, and common goal as node in rectangle figure
- abstract service, and resource as diagram text label
- discontinuous /continuous relationship, and dependency as connections in full line figure
- event as node in ellipse figure
- gateway as node in rounded rectangle integrated polygon figure

The illustration of these elements is shown in Fig. IV. 21.

To be able to graphically create the above concepts, we need a tooling definition model. We classify the tools under three groups: participant group (for creating participants' details), MIS group (for creating MIS's objects), and networking group (for creating the characteristics of collaborative network). Fig. IV. 20 shows the tooling definition model of the CPE:

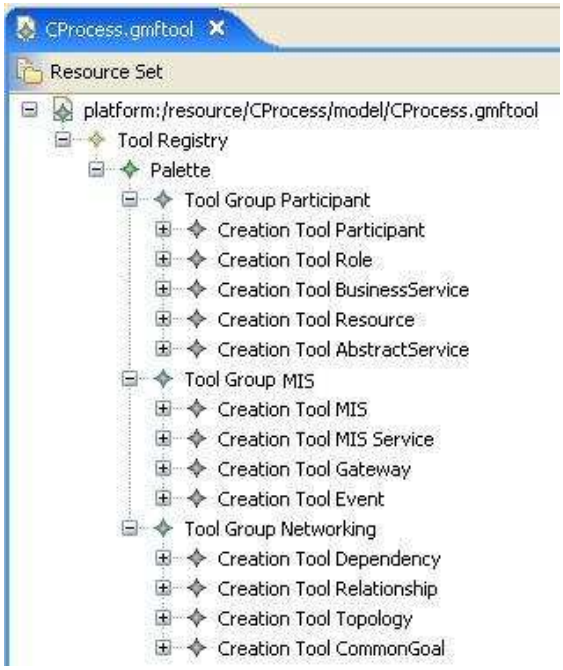


Fig. IV. 20 Tooling definition model of CPE

In comparison to the tooling definition model of NE, we add the tools for creating business service and resource in the first group, and dependency in the third group. The tools in the second group are all new.

The graphical and tooling definition models will be used to generate the runtime diagram of CPE. We show the runtime diagram in the following section.

2.1.3.3. Functionalities

The aim of the CPE is to be an aided design and visualization tool for the collaborative process. Fig. IV. 21 shows the runtime diagram of the CPE. Compared to the NE, the CPE provides the same design space and property sheet, but the tool palette is different.

The tool palette of the CPE contains more tools than the NE's because the domain model of CPE includes the one of NE and has other additional concepts. Thus, as discussed above, the CPE's tool palette contains the same tools as the NE's, and also the tools for creating business service, resource, MIS, MIS service, gateway, event, and dependency. These additional tools are shown in the dashed line rectangles on the right hand side of Fig. IV. 21:

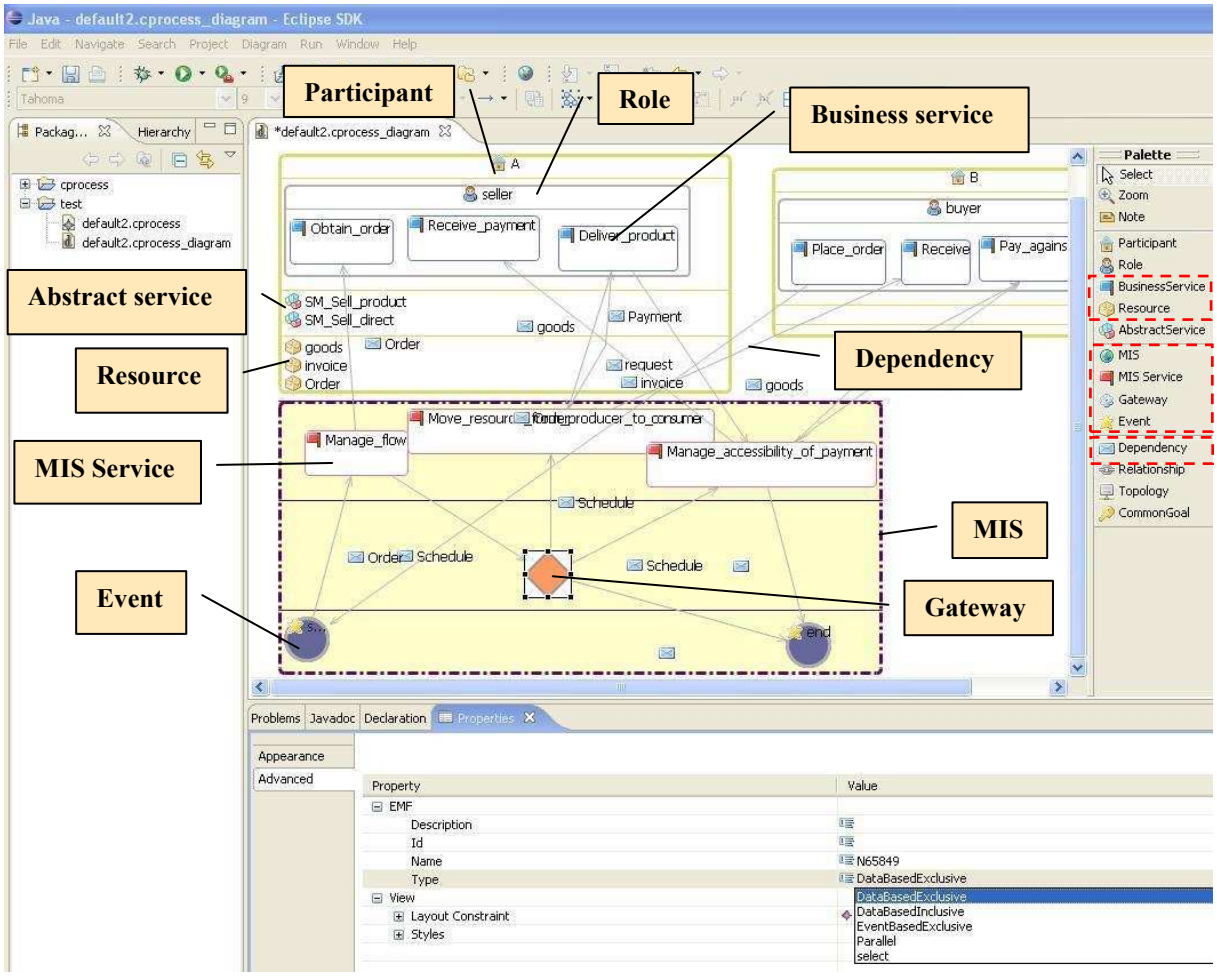


Fig. IV. 21. Runtime diagram of the CPE

In the same way as the NE, the creation of a new collaborative process provides a diagram file (*.cprocess_diagram) representing the process graphically, and its associated XML file (*.cprocess). Once all the partners involved in a studied network have agreed on this process, the XML file will be used to transform it into a BPMN relevant model.

2.1.4. STP BPMN Modeller

We focus on finding a technology for supporting the BPMN construction and visualization functionality. The following figure shows the use of this technology in the prototype:

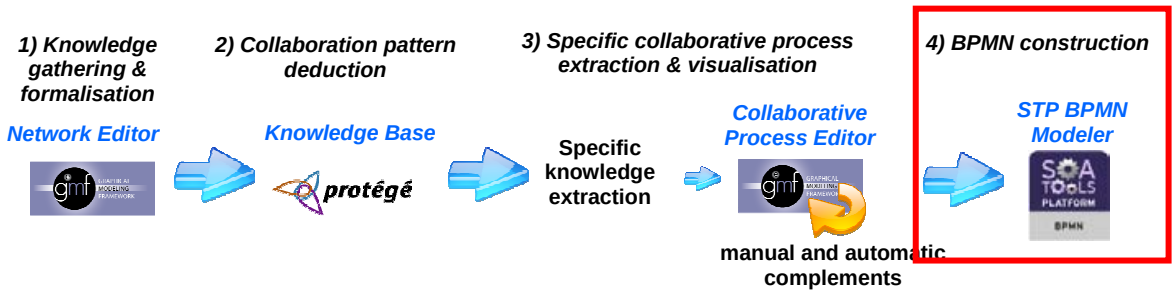


Fig. IV. 22 Use of STP BPMN Modeller in the prototype

This technology is applied at the last step of the prototype. The main intention is to obtain collaborative process models written in BPMN syntax. These models will be used to define a SOA-based MIS model at the logic level (Fig.I.3). Thus, the conformity of BPMN models is required. However, it is not necessary to verify or ensure this issue at this stage as it is taken into account in the previous stage (by the CPE).

2.1.4.1. Selection of technology

The official website of BPMN language [BPMN] provides a list of tools supporting modelling with this language. We are particularly interested in Intalio BPMS Designer, and STP BPMN Modeller. We considered several criteria regarding the selection of our tool:

- supporting the BPMN specification approved by the OMG
- generating an exploitable file in XML preference
- editing, and visualising graphically business processes written in BPMN

Both tools can answer these criteria, and are fully open source but are under different licences. The BPMS Designer is distributed under the Mozilla Public License (MPL) by Intalio. The STP BPMN Modeller is also donated by Intalio as a part of the SOA Tools Platform (STP) project under the Eclipse Public License (EPL). The STP BPMN Modeller complements the BPEL Engine donated to the Apache Software Foundation and the Tempo BPEL4People workflow framework hosted on Intalio.org. All three components form the foundation for Intalio BPMS Designer, the first BPM solution to support a Zero-Code development model [White et al., 2007].

The STP BPMN Modeller seems easier to implement and use than the BPMS Designer. The implementation of this modeller is similar to the NE and CPE since they are all based on the same GMF technology. The STP BPMN Modeller is a graphical editor for creating BPMN diagrams. The possible usage and extensions of this modeller are for example:

- Creating BPMN diagrams to document process orchestration or workflows.

- Generating org.eclipse.stp.bpmn EMF objects. Traverse, annotate, transform to generate BPEL, or other object models.
- Extending the editor to support drag and drop, as well as other application specific usage.
- Implementing a particular version of the BPMN specification: add the properties, validation services, and generation algorithms.

As the BPMN Modeller is based on GMF, its meta-model, and graphical definition, tooling definition, mapping, generator models, as well as runtime diagram plug-in are available and also provided in the package. This is another advantage of using the STP BPMN Modeller rather than the Intalio BPMS Designer since we need the BPMN meta-model in Ecore to perform the model transformation afterwards. This point will be presented in the ATL Transformation (Section 2.2.2.3).

2.1.4.2. Functionalities

The STP BPMN Modeller is based on GMF like the NE and CPE. The main components and models for GMF-based development are provided in the STP BPMN package. Although the STP BPMN Modeller provides the same design space and property sheet as both the NE and CPE, the creation tools contained in the tool palette are dedicated to the design of BPMN processes.

The tool palette of the STP BPMN Modeller contains the tools to allow the creation of the standard elements of BPMN as specified in the OMG specification for example pool, lane, task, message, and sequence flows, as well as several types of gateway, and event. Fig. IV. 23 shows the runtime diagram of the STP BPMN Modeller:

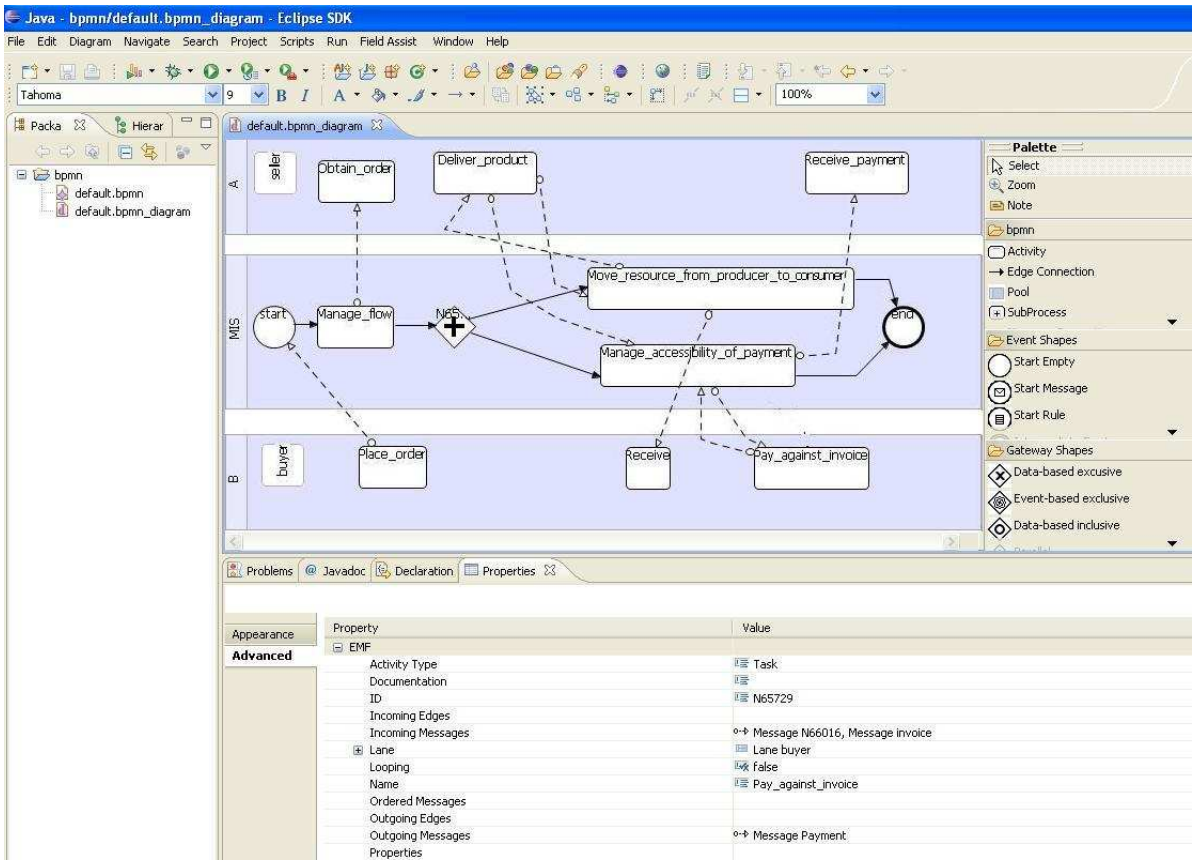


Fig. IV. 23. Runtime diagram of the BPMN Modeller

In the same way as NE and CPE, the creation of a new BPMN model provides a diagram file (*.bpmn_diagram) representing graphically the collaborative process model in BPMN and its associated XML file (*.bpmn).

2.2. Concept and technologies for fulfilling the development of the principal components

In this section, we will discuss the concept and technologies that we use for completing the construction of the whole prototype. Fig. IV. 24 shows the complementary concept and technologies:

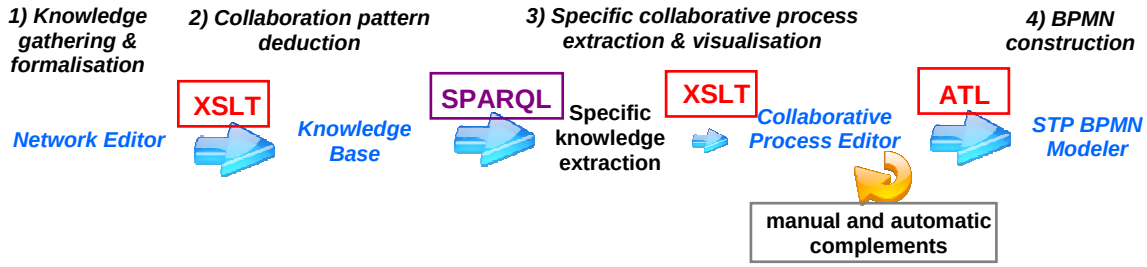


Fig. IV. 24. Other concept and technologies for fulfilling the prototype constitution

The complementary concepts and technologies are classified under three categories:

- Query language (SPARQL) for extracting specific knowledge from the KB. The query helps to extract only the knowledge that we are interested in.

- Transformation languages (XSLT, and ATL) for making the different technologies interoperable. These transformations concern the three movements of knowledge in the same context (Fig. IV. 1): collaboration to collaboration (NE to KB), and process to process (specific knowledge extraction from KB to CPE, and CPE to STP BPMN Modeler).

- Automatic and manual complements at the third functionality: generating gateways and events.

We implement these technologies as well as the NE, CPE, and STP BPMN Modeller on the Eclipse platform. The Eclipse is written primarily in Java to provide software developers and administrators an integrated development environment (IDE). We use the Java APIs to reference all of the programming interfaces and execute the deduction rules in the KB.

2.2.1. SPARQL Query language

The main purpose of using query language is to support the specific knowledge extraction of the third functionality. The query can help in selecting only the interesting knowledge from the KB. Once the extraction of the knowledge has been done, the CPE will manipulate this knowledge and visualize it in the form of a collaborative process.

We start this section by talking about the different technologies for querying ontology and the reasons why we chose SPARQL. Then we will give a brief overview about SPARQL in order to understand how it works. At the end we will present an example of a result obtained after applied SPARQL to our work.

2.2.1.1. Selection of technology

There are a number of query languages for RDF[18] and OWL ontologies, such as RDQL, SeRQL, SPARQL, etc. These query languages were recently conducted by the W3C, and are all based on matching triples with RDF graphs. According to [Haase et al., 2004] and [Hutt, 2005], they compared these three languages by basing the comparison on the crucial properties of query language: value comparison, data type support, expressiveness, semantic support, aggregation functions (min, max, count), and advanced operation support (union, intersection). They found that SeRQL and SPARQL provide nearly the same properties. Moreover, both SeRQL and SPARQL provide SQL-like syntax which is easy to read. SeRQL has strong support from the open source community, while SPARQL has strong support from the W3C. In fact, we finally decided to use SPARQL because it is well documented and standardized. In addition, it provides query results in XML, and it is possible to implement in the Eclipse.

[18] http://www.w3.org/2001/11/13-RDF-Query-Rules/

However, to reason SPARQL queries, we need an engine. Numerous query engines support SPARQL [SPARQL], and many are available as open source software for example, Jena, SPARQL Engine, Pellet, etc. The most frequently used engines are Jena and Pellet. Jena is a Java toolkit for manipulating RDF models which has been developed by Hewlett-Packard Labs. Pellet is an open source Java based OWL-DL reasoner, commercially supported by Clark & Parsia LLC.

Pellet includes an optimized query engine capable of answering ABox queries. It is also possible to use the Jena's query objects with the Pellet's query engine. Note that if the query to be answered is not an ABox query, the Pellet query engine is inapplicable and the bindings will fall-back on using the default Jena query engine [Pellet-Faq].

Hence, we use SPARQL for querying the KB and the queries will be answered with Jena and Pellet engines.

2.2.1.2. Overview of SPARQL

SPARQL stands for Simple Protocol And RDF Query Language. SPARQL[19] is defined in terms of the W3C's RDF data model and will work for any data source that can be mapped into RDF. The specification is under development by the RDF Data Access Working Group (DAWG).

SPARQL is a query language and data access protocol for the Semantic Web:

- The SPARQL query language is a syntactically-SQL-like language for querying RDF graphs via pattern matching [SPARQL-FAQ]. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL queries can return results in XML format.

- The SPARQL protocol uses WSDL 2.0 to define simple HTTP and SOAP protocols for remotely querying RDF databases. It can be used for querying any data repository mapped to the RDF model. The XML results format is used to generate responses from services that implement this protocol.

SPARQL can provide a common query mechanism for all Web 2.0 applications and can be an option for publishing open data on the Web.

2.2.1.3. Query with SPARQL

The following figure shows the application of SPARQL in the prototype:

[19] http://www.w3.org/TR/rdf-sparql-query/

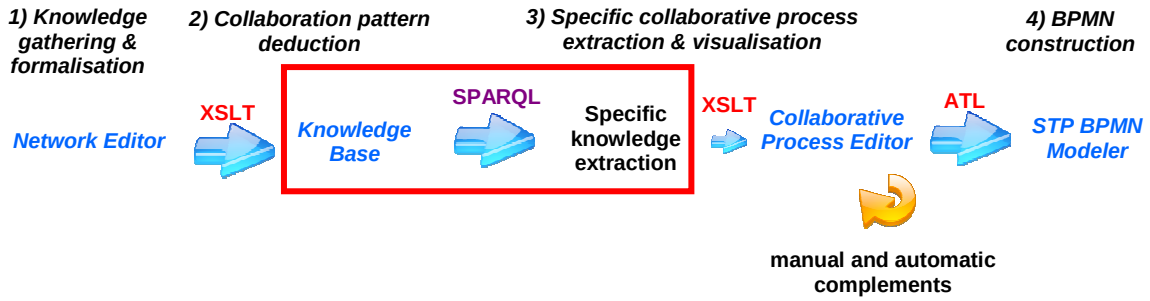


Fig. IV. 25 Use of SPARQL in the prototype

The use of SPARQL is focused on querying the knowledge corresponding to the collaborative network that we are studying. This knowledge is extracted from the KB and will be used later by the CPE. Such knowledge concerns:

- common goals
- relationships and topologies with their type
- participants, their roles, and provided abstract and business services with associated input and output resources
- dependencies and their manipulated resources
- MIS services that coordinate those dependencies

The full implementation of SPARQL is explained in Annex B. Here, we give a short example of a SPARQL query for extracting the name and roles of the participants in a network. The SPARQL works with the KB (owl-based). The query is written as follows:

```
//create an empty model
OntoModel model = ModelFactory.createOntologyModel(PelletReasonerFactory.THE_SPEC);

//read model
model.read("CNO.owl");

String queryBegin =    " PREFIX rdf:  http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                     + " PREFIX   : http://nettyrajsiri.googlepages.com/CNKB_v5.owl#"
String queryEnd = "}";

//query
String queryStr = queryBegin
          + "SELECT  ?name ?role "
          + "WHERE { "
          + "?N rdf:type :CNetwork."
          + "?N :hasParticipant ?P."
          + "?P :name ?name."
          + "?P :playRole ?role."
          + "}"
+ queryEnd;

//create the query
Query query = QueryFactory.create(queryString);
```

Result variables

Specify where to find name and role of participant

Fig. IV. 26 SPARQL query to extract name and role of the participants in a network

The above SPARQL query returns the results *name* and *role* according to the variables indicated in the query. The query result is represented in XML format as follows:

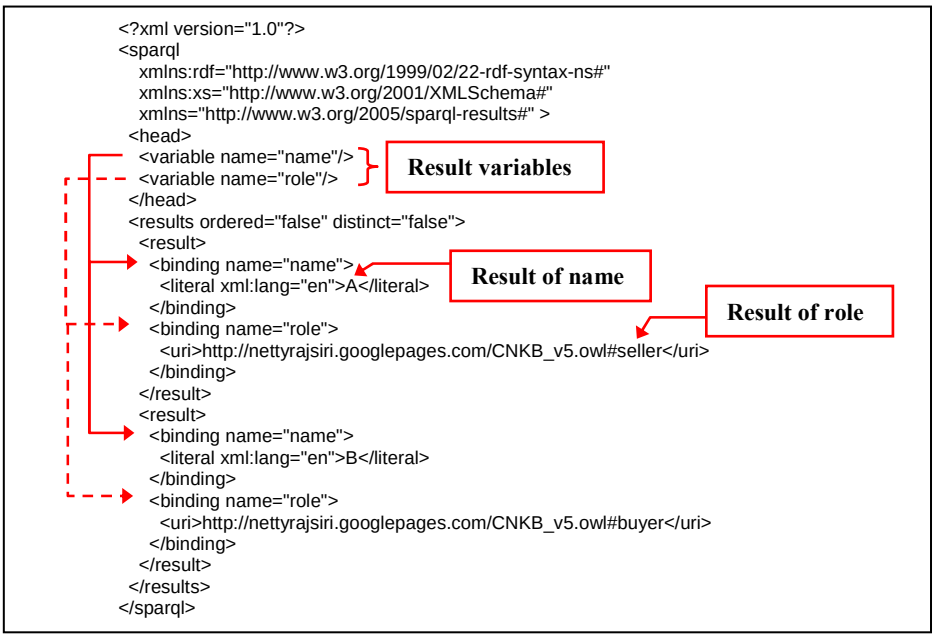


Fig. IV. 27 SPARQL query result in XML

In this example, the query returns two results of *name* and *role*. This means that there are two participants in this studied network. The first participant has the *name A* and the *role seller*. Another participant has the *name B* and the *role buyer*.

2.2.2. Transformation languages

The main purpose of using transformation languages is to convert a model into another which both conform to the same or different meta-models. In our work, we have four components developed basically on different technologies and concepts. So the models of each component are apparently different.

To attain the full implementation of the prototype, we have to link the components together and in the right order. Consequently, transformations of the output model of a component into the input model conforming to its following one are needed. Three transformations are involved in our development:

- Output of NE (XML) into input of KB (OWL)
- Output of SPARQL query (XML) into input of CPE (XML)
- Output of CPE (XML) into input of STP BPMN Modeler (BPMN)

The first two transformations deal with XML-to-XML transformation since OWL is also based on XML [RDF-OWL], while the last one concerns XML-to-BPMN transformation.

Moreover, these three transformations concern the movements of knowledge (Fig. IV. 1): collaboration to collaboration (for the first one), and process to process (for the two last ones).

2.2.2.1. Selection of technologies

Transformation approaches can be distinguished between model-to-code, and model-to-model transformations. All of these approaches support syntactic typing of variables, and patterns. Based on our development, we focus on the XML-to-XML and XML-to-BPMN transformations that concern only the model-to-model transformation approaches. Such transformation translates between source and target models which can be instances of the same or different meta-models.

A large number of model transformation approaches like relational, graph-transformation based, structure-driven, hybrid, and other approaches have been developed since the OMG issued a Request for Proposal on QVT (Query/Views/Transformations) in 2002. Various model transformation languages like BOTL, MOLA, GreAT, AndroMDA, F-Logic, UMLX, ATL or XSLT have been proposed in order to respond to a specific context [Czarnecki et al., 2003]. Thus, it is not easy to converge on a single language.

To deal with an XML-to-XML transformation, XSLT appears as the most outstanding XML model transformation language [Czarnecki et al., 2003]. Since models can be serialized as XML using the XMI (XML Metadata Interchange) implementing model transformation, using XSLT seems very attractive. XSLT is a computer language designed specifically to transform an input XML document into an output XML document which satisfies some specific goal.

However, using XSLT for transforming an XML into a BPMN seems to be more complicated than the XML transformation, and requires much effort due to the complexity of BPMN meta-model. Transformation by using meta-model concepts on a very high level appears to be more realistic. Such transformation requires a mapping definition between elements of meta-models. ATL is pretty good at meta-model-based transformation. Moreover, the works of Touzi use ATL to transform BPMN into SOA-based UML which the BPMN model is written in Ecore. ATL seems therefore to be our best choice for the compatibility with the work of Touzi and also the possibility of generating an Ecore model as a result.

2.2.2.2. Transformations with XSLT in the prototype

The following figure shows that we apply the transformations with XSLT twice in the prototype:

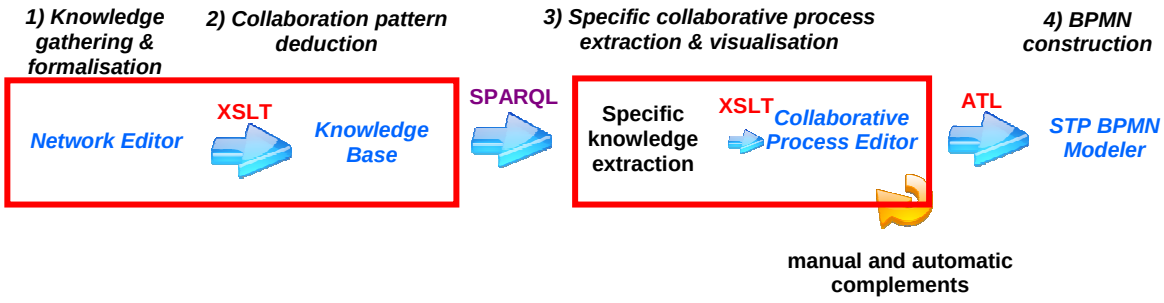


Fig. IV. 28 Uses of XSLT in the prototype

In Annex C, we provide a complete example of transformation with XSLT.

a. From Network Editor to Knowledge Base

The first transformation with XSLT concerns the transformation of an XML-based collaborative network model of the NE into an OWL-based model of the KB. Here, we intend to obtain the result model conforming to the OWL-based CO (upper part of Fig.III.8).

Thus, the source model conforms to the meta-model of NE (Fig. IV. 6), while the result model is to be imported into the KB. The mapping between the source and result elements is defined as follows:

N°	XML source elements (NE-based) ⟶	Result elements (OWL-based KB)
1	CNetwork:network	CNetwork
2	topology	Topology
3	commonGoals	CommonGoal
4	participants	Participant
5	role	playRole
6	relationship	Relationship

Table IV. 1 XML source elements and their corresponding result elements

The above mapping shows that there are six rules that we have to define in the XSLT stylesheet. The following figure illustrates this transformation including the source and result models as well as the associated number of transformation:

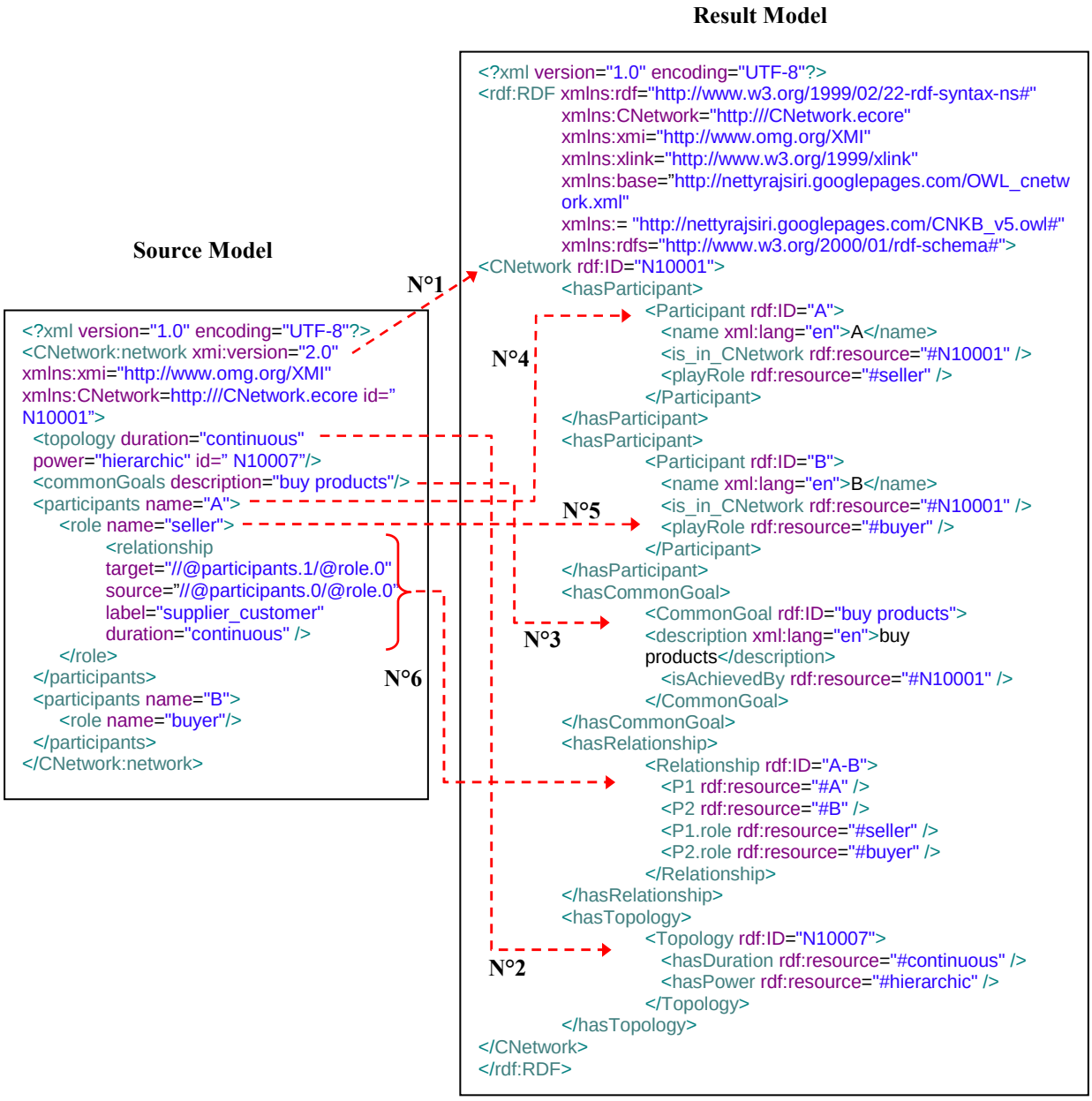


Fig. IV. 29 Transformation with XSLT of a source model (NE-based) into a target model (OWL-based)

It seems that this transformation is quite simple and direct. The names of the source elements are mostly the same as the ones of the result elements. The transformation N°6 (Relationship) is more complicated than the others because we have to deal with the positions defined in the values of the *target* and *source* attributes. The rule has to find the corresponding positions of participant and role on the XML schema. Besides, there are some additional elements in the result model which have to be engraved in the XSLT stylesheet, for example, *hasParticipant*, *hasCommonGoal*, *hasRelationship*, *hasTopology*, etc. These elements are the properties defined in the CNO ontology.

b. From Specific knowledge extraction to Collaborative Process Editor

The second transformation with XSLT concerns the transformation in the third functionality. This transformation is from an XML-based model of a specific knowledge extraction by SPARQL into an XML-based model of CPE. Here, we intend to obtain a collaborative process model conforming to the meta-model of CPE. This means the CPE can visualize the model obtained from this transformation without any problem.

The SPARQL query results are defined in the result elements which contain one binding for each variable. The meta-model of CPE is shown in Fig. IV. 18. The mapping between source and result elements is as follows:

N°	XML source variables (SPARQL-based) ⟶	Result elements (CPE-based)
1	binding *topology*	topology
2	binding *commonGoal*	commonGoal
3	binding *participant*	Participants
4	binding *role*	role
5	binding *relationship*	relationship
6	binding *abstractService*	providesAbstractService
7	binding *businessService*	performsBusinessService
8	binding *input*	input
9	binding *output*	output
10	binding *dependency*	flows
11	binding *MISservice*	MIS service

Table IV. 2 XML source SPARQL variables and their corresponding result elements

The above table shows that the transformation is direct. The complexity and difficulty of this transformation depend particularly on the numbers and names of variables used in the SPARQL queries.

The following figure illustrates the XSLT transformation (N°3 and 4) of a SPARQL query result into a collaborative process model conforming to the CPE:

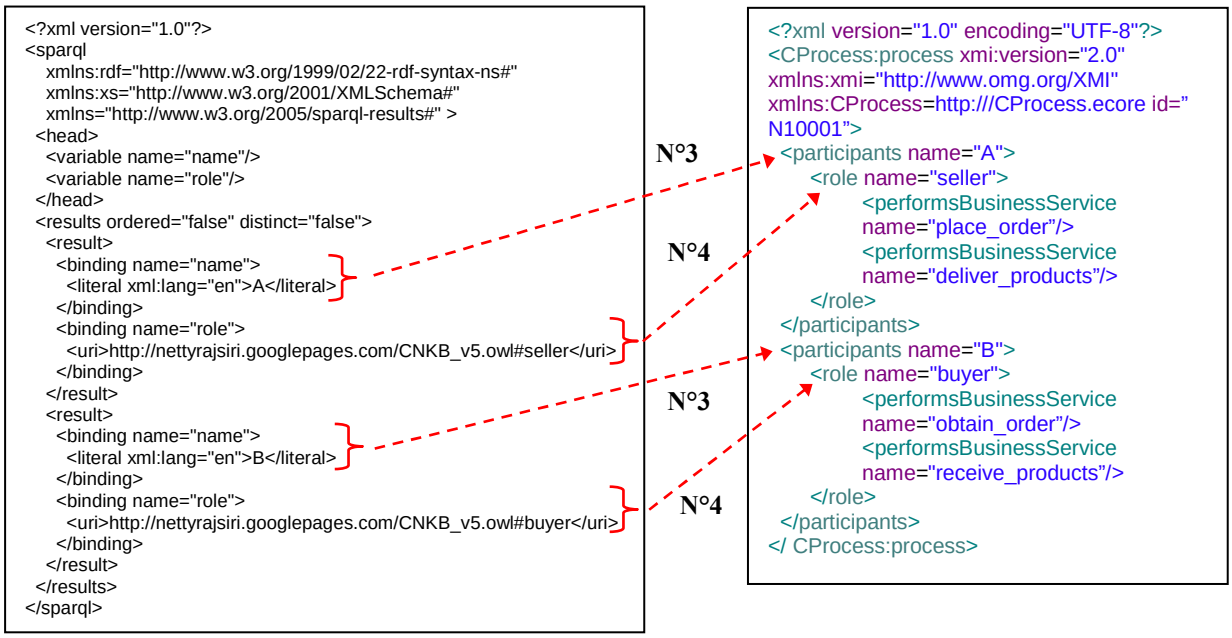


Fig. IV. 30 Example of a transformation of a SPARQL query result into a CPE-based model

2.2.2.3. Transformation with ATL in the prototype

The transformation with ATL takes place at the last step of the prototype in order to connect the CPE and the STP BPMN Modeller together. The following figure shows the use of ATL in the prototype:

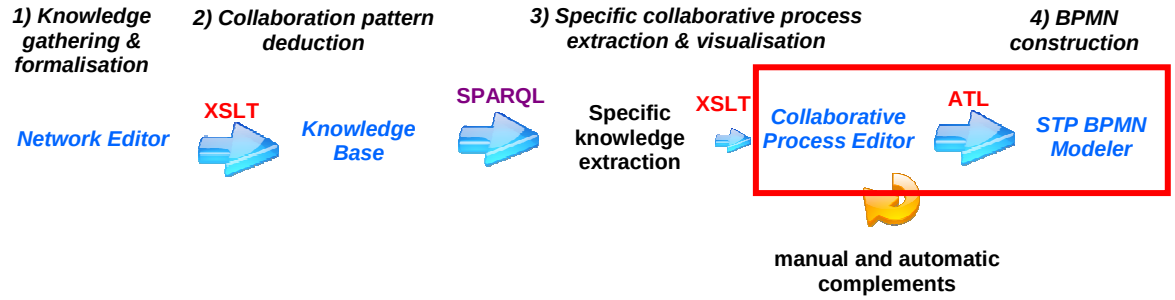


Fig. IV. 31 Use of ATL in the prototype

According to [ATL manual, 2006], transformation with ATL requires a source meta-model, a source model, a target meta-model, and an ATL file. It is also mandatory that these two meta-models and the source model should conform to Ecore.

The full transformation with ATL is explained in Annex D. Here below, we summarize the important points for performing this transformation:

- The source model is generated by the third functionality (the CPE). However, the model obtained from the CPE is described in XML, not Ecore as required. Thus, we need to inject the CPE-based source model into an XML model programmatically with a Java application. This injected XML model is Ecore-based.

- The source meta-model is not the meta-model of CPE, but XML.
- The target meta-model is, of course, the BPMN meta-model.
- The mapping between the elements of these two meta-models is defined as follows:

N°	XML meta-model (Source) →	BPMN meta-model (Target)
1	Root	BpmnDiagram
2	Element with the name's value *participants* or *CIS*	Pool
3	Element with the name's value *role*	Lane
4	Element with the name's value *performsBusinessService*, *CISservices*, *gateways* or *events*	Activity
5	Element with the name's value *flows* and type *seqFlow*	Edge
6	Element with the name's value *flows* and type *msgFlow*	Message

Table IV. 3 Mapping between the elements of XML and BPMN meta-models

The above mapping shows that there are six transformation rules that we have to define in the ATL file. All these six rules have to be applied in order to complete the transformation of the collaborative process obtained from the CPE into the BPMN model.

Fig. IV. 33 shows an example of a two-participant collaborative process using the CPE. After applying the ATL transformation to this example, we obtain the target model as shown in Fig. IV. 33. We also indicate the number of transformations on the figure.

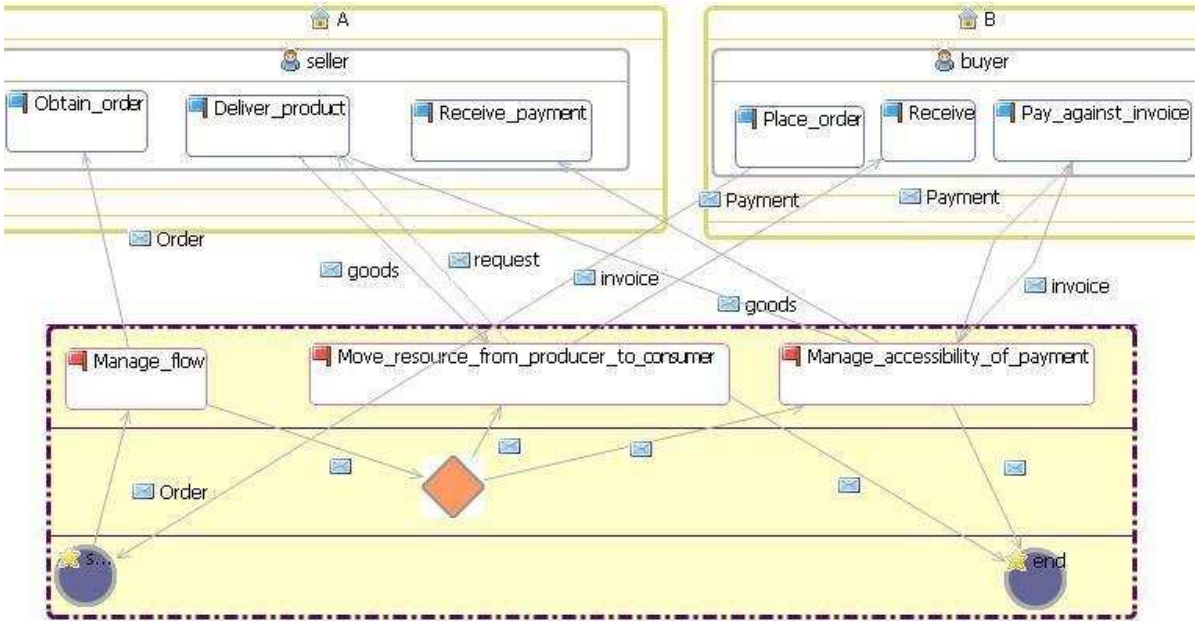


Fig. IV. 32 Collaborative process between two participants

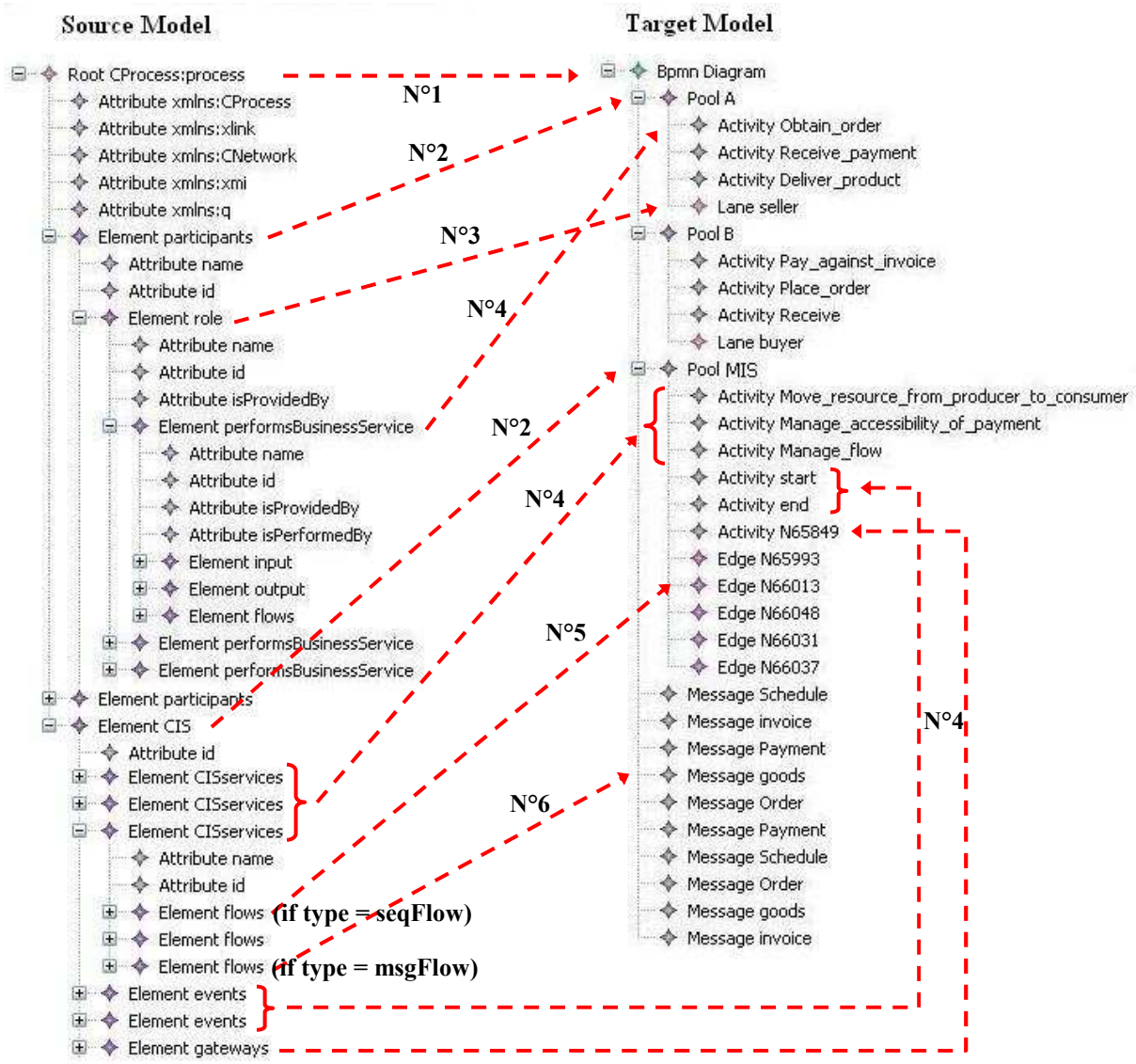


Fig. IV. 33 Six transformations of the source model and the target model (Ecore view)

The BPMN diagram corresponding to the above target model can be visualised with the STP BPMN Modeller. The following figure shows this BPMN diagram:

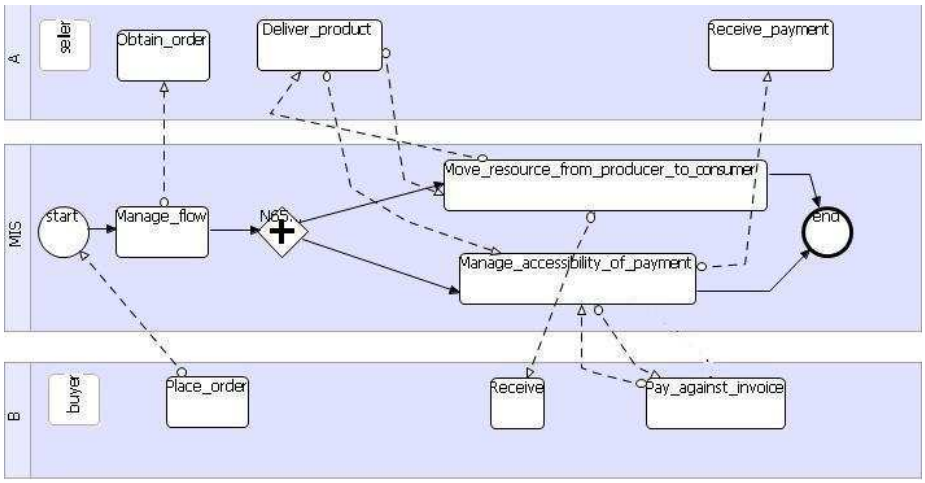


Fig. IV. 34 BPMN diagram visualised with STP BPMN Modeler

We can see that on this BPMN diagram, the A, B and MIS pools provide several services. Only the MIS pool has additional events, gateway, and sequence flows (full lines). The message flows (dashed lines) communicate between services across pools. The way the elements are arranged on this BPMN diagram corresponds to the restrictions specified in the meta-model of collaborative process (Fig.II.6). We have mentioned so far that the BPMN diagrams obtained at the last step of the prototype conform to the meta-model of collaborative process as we take into account this meta-model in the CPE (Third functionality).

2.2.3. Complementary concepts at the third functionality

The third functionality concerns the specific collaborative knowledge extraction and visualization functionality. We remind ourselves again here of the four actions in this functionality:

1) Extraction of knowledge by using SPARQL queries already discussed in Section 2.2.1.
2) Representation of the queried knowledge in form of collaborative process with the CPE tool, already presented in Section 2.2.2.2/b.
3) Verification of the collaborative process generated from the previous action with the involved partners.
4) Generation of a new complete collaborative process composed only of relevant objects, events, and gateways.

The two last actions have not been presented yet. Thus, the following sections are dedicated to these two last actions. We start with the third action then the generation of gateways and events of the fourth action will be presented separately.

2.2.3.1. Verification of collaborative process

The verification is to be applied to the very first collaborative process generated and visualized with the CPE (result of the second action). We defined two levels of verification:

- The first level concerns the deletion of services including business services, abstract services, and MIS services. This deletion causes the automatic removal of associated flows and resources. Up to now, this level has been done manually. The CPE's users have to work with the network participants in order to delete the relevant elements and keep only the required ones in the process. In the future, we intend to develop a graphical user interface, like a checkbox, that will allow users to make multiple selections from a list of services.

- The second level concerns the sequencing of services. Here we have to take into account the meaning of flows between services in the collaborative process. We defined an algorithm to verify and bind the resources transferred between services with chain. The result of this algorithm is a suggestion of service sequencing. The CPE's users can use this suggestion to organize services. At the current stage, this level has not been fully implemented.

2.2.3.2. Generation of gateways

The generation of gateways should be done in the last action. Why is this generation needed?

The collaborative process models we visualize with the CPE come from the knowledge extracted from the KB by SPARQL query. Because we do not take into account the concept of gateway when defining the CNO of the KB, the extracted knowledge does not concern this concept.

The concept we use for generating gateway is based on dependency. Dependencies are referred to the flows of resources that are transferred between services of the enterprises. Gateway is an important modelling element of BPMN. It is used to control how flows interact as they converge and diverge within a Process [BPMN, 2004]. The extraction from the KB by SPARQL query offers the knowledge about the dependencies. Fig. IV. 35 illustrates the correspondence between dependencies and gateways:

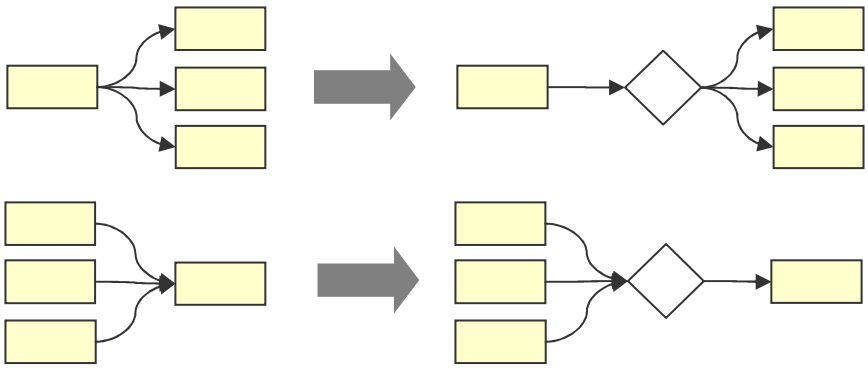


Fig. IV. 35 Relations between dependencies and gateways

Note that the rectangle figure can represent a service, or even a gateway. The polygon figure (right side) represents only a gateway that controls multiple flows in or out. Thus, the two

patterns of dependencies (left side) concern dependencies between services, gateways, or combinations of them.

We define some transformation rules with XSLT based on these two patterns. Once the patterns match, the gateways are automatically generated. However, we can only generate the figure of gateway, not its type. We are interested in these four types of gateway:

- Parallel gateway: every incoming flow is active when this gateway is used as a separator. In the case of merging, this gateway provides a token to each outgoing flow.

- Data-based exclusive gateway: the outgoing flows from this gateway are alternative and conditioned. The condition test is done in order (top down) and once one of them is valid, the token is passed on. In the case of merging, this gateway lets every token pass successively.

- Event-based exclusive gateway: the principle functionality of this gateway is mainly the same as the previous one, but the condition is based on an event attached to the gateway.

- Data-based inclusive gateway: the outgoing flows from this gateway are conditioned. The difference from the data-based exclusive gateway is that it is possible to have more than one valid outgoing flow from the inclusive gateway, which is not the case with the exclusive gateway. In the case of merging, this type of gateway synchronizes the incoming flows and fuses the tokens simultaneously.

The type and conditions of the gateway will be manually specified by the CPE's user because it needs to take into account the meanings of the resources containing in each flow.

2.2.3.3. Generation of events

The generation of events should be done in the last action the same as with gateways. For the same reason as gateways, the generation of events is needed.

Up to now we have only taken into account the generation of start and end events. Such generation is based on the dependency concept. We list the rules applied to this generation as follows:

- If MIS service has no outgoing sequence flow (flow in the MIS), then generate a new sequence flow out from this MIS service to the end event. The figure below illustrates this rule:

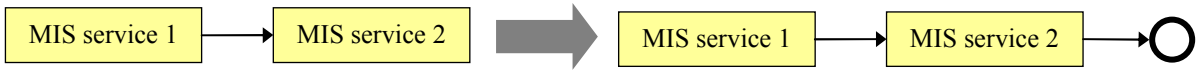


Fig. IV. 36 Generation of end event

- If business service of partners has no incoming message flow (flow between MIS and partners), then generate a new message flow out from this business service to the start

event and generate a new sequence flow out from the start event to the initial MIS service. The figure below illustrates this rule:

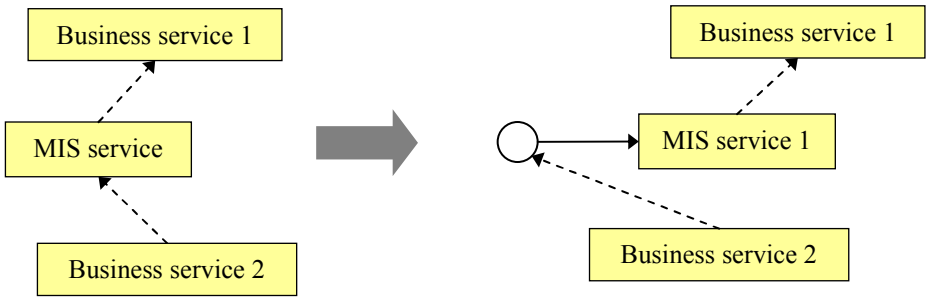


Fig. IV. 37 Generation of start event

These transformations are automatically done with XSLT. The start and end events can be empty, message, time, etc. Their types will be specified manually by the CPE's users. However, it is possible to have intermediate event in the BPMN process. At the current stage, intermediate events needed in the process are also added manually.

3. Conclusion of the chapter

In this chapter, we presented the prototype of our knowledge-based system to automate the specification of collaborative processes. According to Fig.IV.1, the system is composed of three main parts: knowledge gathering (left), knowledge representation and reasoning (middle), and collaborative process modelling (right).

Knowledge gathering concerns the acquisition of implicit knowledge expressed by business partners on the basis of their experiences and perspectives. We developed the Network Editor (NE) to facilitate the acquisition of knowledge and formalize this knowledge.

Knowledge representation and reasoning is the most important part of the system because it deals with the morphing of knowledge on collaboration into a collaborative process. We use the ontology-based approach (presented in theory in Chapter 3) as a means of accomplishing this second part. The Knowledge Base (KB) has been developed on the basis of such an approach and allows the deduction of collaboration patterns.

Collaborative process modelling concerns the extraction of collaborative process knowledge, and the construction of BPMN collaborative process model from the extracted knowledge. To accomplish this part, we implemented several technologies and tools. We extract the knowledge on collaborative process from the KB by SPARQL queries. This extracted knowledge is organized in the form of a collaborative process model and visualized with the Collaborative Process Editor (CPE). The complementary concepts (verification and generations of gateways and events) are also needed in this functionality. Even though the gateways and events are automatically generated, we need to re-verify the result of the generations in order to ensure that the collaborative process model is correct. This collaborative process model is BPMN-oriented, but does not yet conform to the BPMN specification. The ATL transformation is introduced to deal with this issue. The real BPMN

process is visualized with the STP BPMN Modeller. This tool is provided under the Eclipse Public License (EPL).

Many technologies were integrated to build up the prototype for example, GMF, Protégé, SWRL Editor, Jess, SPARQL, Jena, Pellet, XSLT, and ATL. They were all implemented on the Eclipse platform based on Java. Java APIs have been applied to reference the programming interfaces.

The prototype was developed under the hypotheses discussed in Chapter 1. Our prototype provides the necessary knowledge (collaborative process model) to be used as the input for the works of Touzi (MISE project) and for the runtime collaborative platform of EBM WebSourcing. The main constraints for our work originated from the works of Touzi which requires the collaborative process model to be written in BPMN and to conform to the meta-model of the collaborative process (Fig.II.6). We ensure the conformity of the collaborative process model at the CPE by taking into account this meta-model when specifying the CPE's domain model.

The prototype will be tested and validated in the next chapter with a collaborative process example.

Chapter 5. Experimentation

In the previous chapter, we presented the prototype of our knowledge-based system dedicated to facilitating the collaborative process specification. The prototype consists of four main functionalities. We developed four tools to support the specific requirements of these four functionalities. Some tools can be executed automatically, but some require user efforts to accomplish.

The objective of this chapter is to demonstrate how the prototype works by experimenting with it on collaboration cases. This chapter starts by defining the scope of the experimentation. The experimentation will be explained step by step in order to show the mechanisms of the prototype.

1. Scope of the experimentation

The prototype was tested with several collaborative situation case studies. The objective of these experiments was to test the prototype that we have developed, as well as to validate its concepts and functionalities in order to improve it later. Moreover, the experiments may allow us to realize if the prototype is able or not to specify appropriate collaborative processes from any collaborative situations.

The experimentation follows the four steps of the approach, as shown in Table V. 1 and Fig.V. 1. Each step has its own tool(s) some of which may require human actions.

Steps	Details	Tools	Remarks
Step 1: Knowledge gathering and formalization	- collect knowledge from involved partners - use knowledge to model a relevant collaborative network	- Network Editor (NE)	manual
Step 2: Collaboration pattern deduction	- transform and import the collaborative network model into the KB - execute the deduction rules	- Knowledge Base (KB)	automatic
Step 3: Specific collaborative process extraction & visualisation	- query the knowledge specific to the input network - organize that knowledge in form of collaborative process - visualise the collaborative process with CPE	- SPARQL - Collaborative Process Editor (CPE)	semi-automatic
Step 4: BPMN construction & visualization	- transform the obtained collaborative process into the BPMN process - visualise the BPMN process with STP BPMN Modeller	- ATL - STP BPMN Modeller	automatic

Table V. 1 Four steps of experimentation

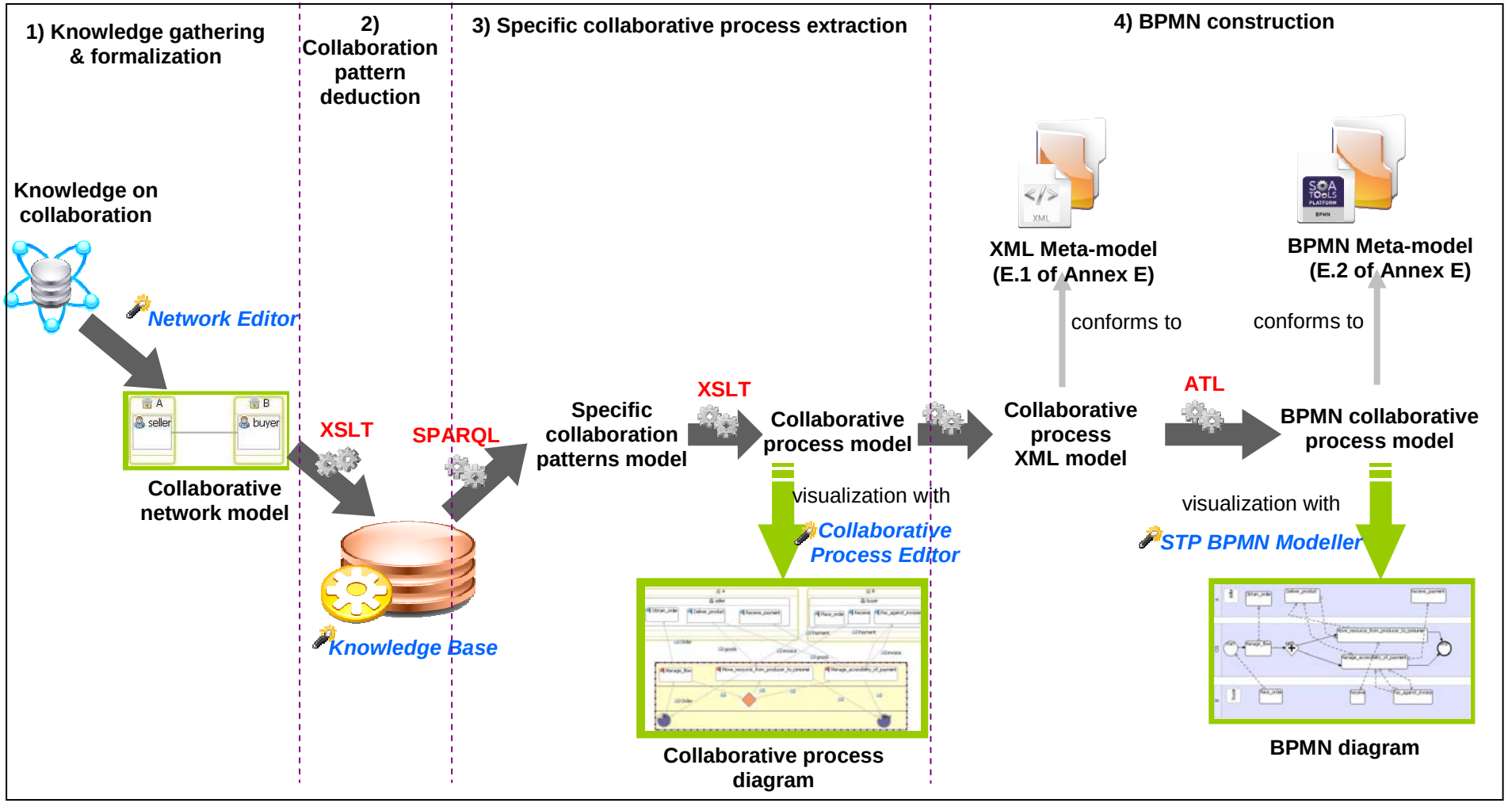


Fig.V. 1 Four steps of experimentation with tools

We demonstrate here two collaboration cases: a simple case and a complex one. The simple collaboration has only two partners, while the complex one has seven partners. Both cases are in a supplier-customer context. In Section 2, the experimentation of a simple collaboration case will be explained in detail in order to show the mechanisms and the use of the prototype. The tools will be applied to perform each step of the approach (from the knowledge gathering to the BPMN construction). In Section 3, the other experimentation with a complex collaboration will not detail the mechanisms of each step, but the results obtained at the end of each step.

2. Supplier-Customer Experimentation

In this section, we conduct a very simple collaborative situation between a supplier and a customer. This situation is really general, making it relatively simple to understand how our modelling approach works.

The network has two partners (*A* and *B*) who collaborate in order to buy 100 parts of bolts. Partner *A* has bolts that partner *B* desires. The collaboration starts when *B* sends a purchase order to *A*. They establish a strong relationship and communicate with each other frequently.

This simple case allows us to conceive its collaborative process, which is harder to do in the complex case. The collaborative process we imagined before starting the experimentation compounds mainly the sell and buy services. The process will start by the customer sending an order to the supplier. Then, the supplier prepares and delivers the ordered goods to the customer. Finally, the customer receives these goods and pays. This is what we expect to obtain at the end of the experimentation.

2.1. Step 1: Knowledge gathering and formalization

a. Objective

This step focuses on capturing all necessary knowledge concerning the collaboration that will take place. We classified the knowledge to be collected under two groups:

- collaboration characteristics: relationships between each pair of participants, and common goals
- participants' details: general information of participant, roles, and abstract services

b. Application

The Network Editor (NE) is the tool we used in order to facilitate the knowledge gathering and formalization. The interoperability providers (e.g. consultants of EBM WebSourcing) are the main users of this tool. The partners involved in the studied network were brought together and interviewed by the consultants. They were asked about:

- Common goal(s) that the partners define together
  - What are the goals of this collaboration?

  - What do the partners intend to achieve?
  - What is the network established for?
- Intensity or frequency of communication
  - How often do the partners communicate to each other during the collaboration?
  - Is the collaboration occasional?
- Roles and responsibilities in this collaboration
  - Who does what?
  - What does each partner provide to the others?
- Dispersion of decision-making power in this collaboration
  - Who holds the power?
  - Who makes decisions?

Note that if a studied network has more than two partners, we have to determine whether a relationship exists or not between each pair of partners. This definition allows us to know who communicates with whom, and the type of relationship they have. But this was not applied in our case since our studied network has only two partners. It is also possible for each partner to play more than one role.

c. Result obtained

The interoperability provider interpreted the knowledge obtained from the interview, as shown in the table below:

Participants	*A*	*B*
Roles	*Seller*	*Buyer*
Relationship	*Supplier-customer*	
Decision making power	*Hierarchic*	
Duration type	*Continuous*	
Common goal	*Buy 100 parts of bolts*	

Table V. 2 Interpretation of the supplier-customer collaboration

Fig.V. 2 shows the relevant collaborative network diagram drawn with the NE tool:

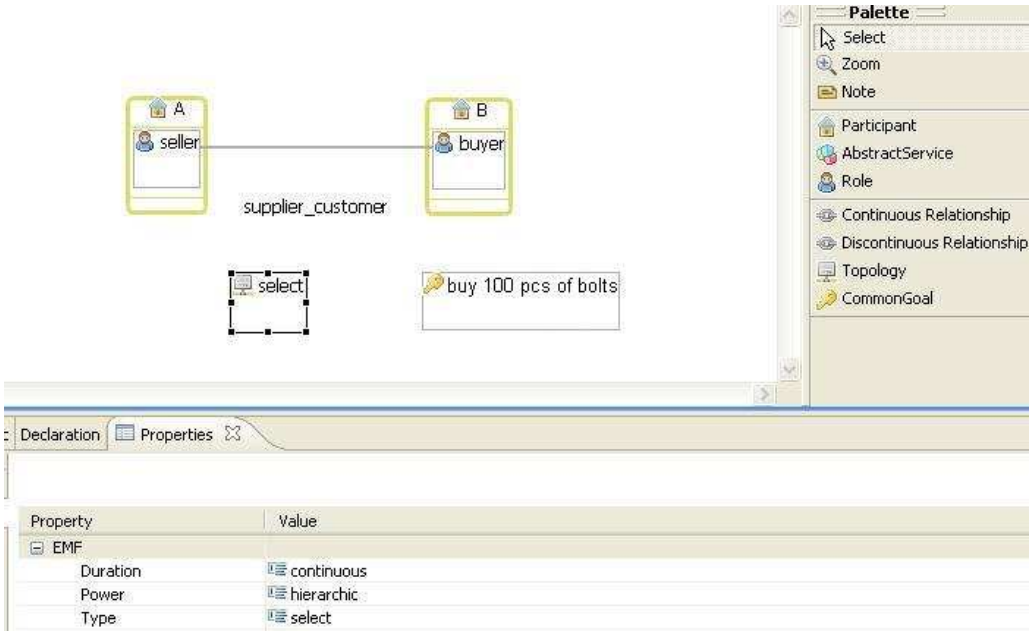


Fig.V. 2 Collaborative network diagram of the supplier-customer collaboration

This diagram represents, through the NE, a network model in XML schema. The associated XML network model is transformed and imported into the KB afterwards.

The definition of the collaborative network depends mostly on the interpretation and experience of the NE's users. For the same knowledge (captured by interview), users may interpret it differently. Thus, it is possible to define more than one potential collaborative network in order to obtain an appropriate collaborative process that satisfies the clients as much as possible.

2.2. Step 2: Collaboration pattern deduction

a. Objective

This step concerns the deductive reasoning of collaboration patterns based on the collaborative network model obtained from the first step. The deduction is performed over the Knowledge Base (KB).

b. Application

The deduction can be executed once the NE's collaborative network model has been characterized and imported into the KB as a new individual of the *CNetwork* class. To be able to import, we have to transform the NE's network model into an OWL model conforming to the KB. This transformation is an XML-to-XML transformation which is done by applying an XSLT stylesheet as discussed in Section 2.2.2.2/b (Implementation of XSLT) of Chapter 4. After the transformation, we obtained an OWL model of our collaborative network ready to be imported into the KB.

The SWRL deduction rules and a number of individuals have been defined and stored in the KB. These individuals come from the dataset (an OWLized version of the MIT Process Handbook) as discussed in Section 2.1.2.3/a (Individuals) of Chapter 4. The rules were specified and it was shown how the deductions could be made in Section 3.2.2 of Chapter 3.

The SWRL deduction rules were performed programmatically over the Jess engine in the KB via the *SWRLRuleEngineBridge* API Java. We have talked about processing the SWRL rules with the Jess engine in Annex F. Once the deduction was made, new OWL concepts were created and inserted into the KB automatically.

c. Result obtained

Fig.V. 3 shows that the individual named *N10001* of the *CNetwork* class was created. This individual refers to the collaborative network model defined in the previous step. Once the deduction rules were executed, a number of concepts were automatically added into the corresponding properties of this *N10001* individual. For example:

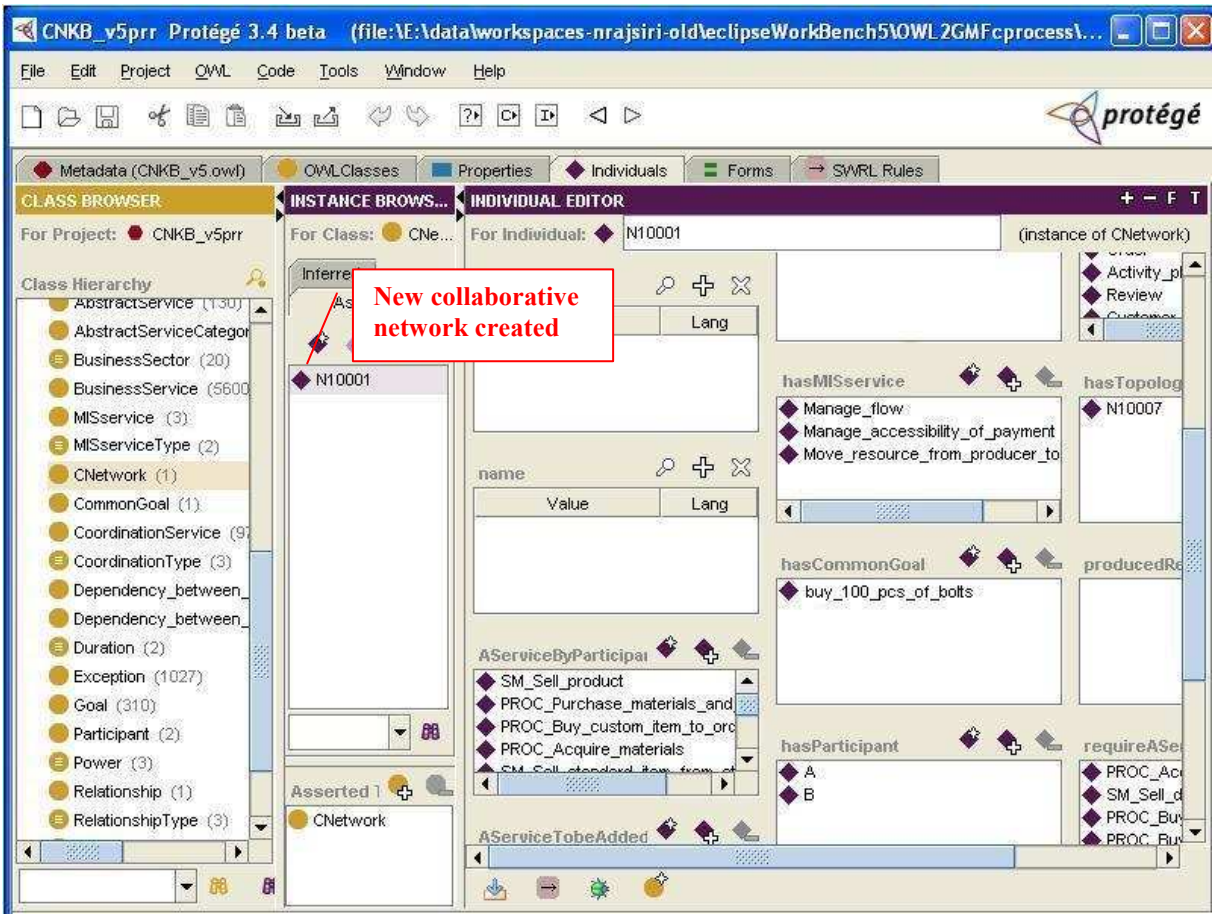


Fig.V. 3 Knowledge Base showing the new individual of the CNetwork class and its deduction results

- Participants *A* and *B* were created in the *Participant* class and inserted into the *hasParticipant* property.
- Common goal *buy 100 parts of bolts* was created in the *CommonGoal* class and inserted into the *hasCommonGoal* property.

- *Manage flow*, *Manage accessibility of payment*, and *Move resource from producer to consumer* were deduced into the *hasMISservice* property.
- Topology *N10007* was created in the *Topology* class and inserted into the *hasTopology* property.

2.3. Step 3: Specific collaborative process extraction and visualization

a. Objective

This step aims to propose a collaborative process model that corresponds as much as possible to the collaborative situation defined in the first step. Moreover, all involved partners of the studied network should agree on using this process model for handling their collaboration.

This step is focused mainly on querying the knowledge specific to the collaborative situation from the KB. The knowledge obtained will be organized in form of a collaborative process that can be visualised by the CPE.

b. Application and results obtained

The application of this step is the most complicated in comparison to the other steps. It is composed of four actions some of which require users' effort to accomplish. Only for the first action we use SPARQL query language, while for the others we use the CPE tool.

Action 1: Extraction of knowledge concerns inquiring to the KB about the knowledge related to the studied network through SPARQL queries. As discussed in Section 2.2.1.3 of Chapter 4, we wrote the queries in order to obtain this knowledge:

- common goals
- relationships, and topologies with their type
- participants, and their roles, as well as the abstract services they provide
- business services, and their associated input, and output resources
- dependencies, and their manipulated resources
- MIS services that coordinate those dependencies

The queries are generic enough and applicable to any collaboration cases.

Result obtained: The result of the SPARQL queries was written in the XML schema. As this schema is really huge, we do not provide it in an XML form, but in a table-based form. Table V. 3 summarizes the variables used in the SPARQL queries and their returned results:

Variables	Results
?commonGoal	buy 100 pcs of bolts
?relationship	A-B
?topology	chain typed topology
?participant	A, B
?role	A: seller, B: buyer
?abstractService	A: sell, sell product, sell via distributor, sell direct, market and sell...

	B: buy, buy resources, buy in a store, source, acquire materials, purchase capital goods...
?businessService	A: obtain order, deliver, receive payment, market product to relevant customer, manage customer relationship…. B: identify needs, place order, manage suppliers, receive products, store items, pay...
?input	Order, authorization, goods, money, bill…
?output	Order, goods, money, bill, schedule…
?MISservice	Manage flow (order, schedule…) Manage accessibility of payment (bill, pay…) Move resource from producer to consumer (goods…)

Table V. 3 Variables and results obtained from the SPARQL queries

An example of XML result of a SPARQL query is shown in Fig IV. 28.

Action 2: Representation of the queried knowledge in form of collaborative process visualizable with the CPE tool. This action concerns the transformation of the XML schema of the SPARQL results into an XML model that conforms to the CPE tool. This transformation was done in the same way in which the NE's network model was transformed into an OWL model in Step 2. An XSLT stylesheet has been created in order to handle this transformation, and can be applicable to any collaborative situations.

Result obtained: Since we tried to extract as much knowledge as possible regarding the given collaborative situation, the obtained collaborative process is probably fussy as shown below:

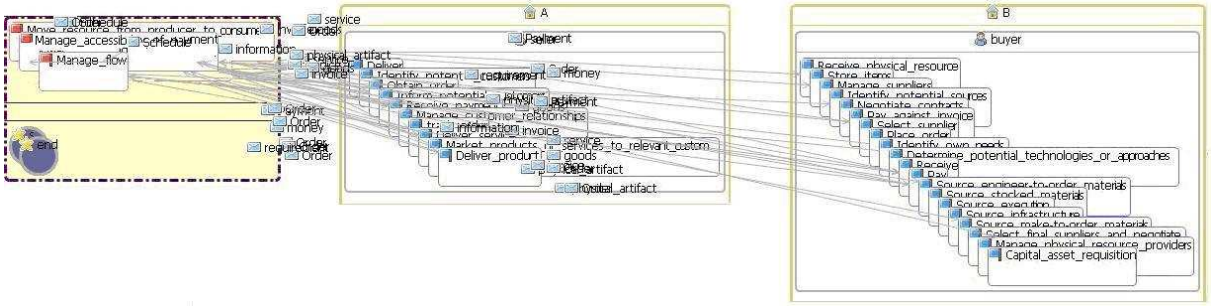


Fig.V. 4 Fussy collaborative process generated

Action 3: Verification of the collaborative process generated from the previous action with all involved partners (A and B). The main objective of this action is to manually delete the undesired objects (e.g. business services, dependencies, collaborative services, etc.) from the process. In Section 2.2.3.1 of Chapter 4, we discussed how we need to develop a user interface to facilitate this action and why this action needs to be carried out collaboratively with the partners. In this case, we deleted for example:

- Business services of A: manage customer relationship, deliver services, track order, identify potential customers, etc.
- Business services of B: store items, receive physical resource, manage supplier, identify potential sources, source infrastructures, etc.

Result obtained: The collaborative process after deleting the undesired objects is shown as follows. This process was the first validation that all involved partners made.

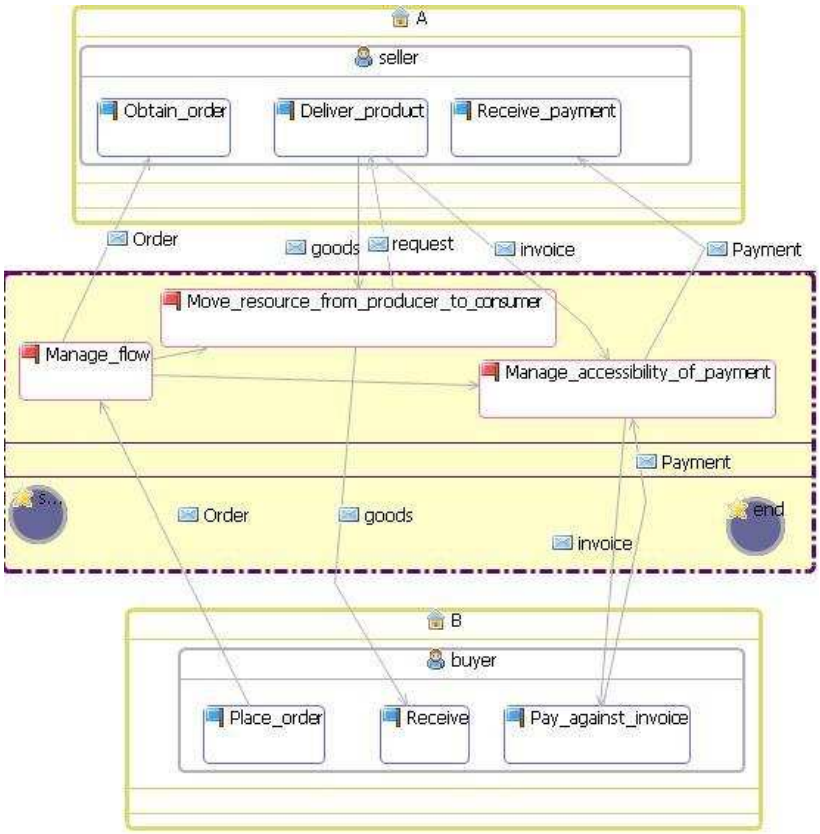


Fig.V. 5 Collaborative process after deleting unnecessary objects

The collaborative process model obtained here is not complete yet. This model is created from the knowledge extracted from the KB by SPARQL queries (action 1). The KB is originally built upon the CNO which does not take into account the concepts of gateway and event which are the important modelling elements of BPMN. It is used to control how flow interacts as they converge and diverge within a process.

Action 4: Generation of a new collaborative process that consists only of relevant objects as well as gateways and events. The concept for generating gateways and events has been presented in Section 2.2.3.2 and 2.2.3.3 of Chapter 4. A type of gateway was selected by considering the direction of resource flows passing around the collaborative process. All partners made the final validation of the collaborative process.

Result obtained: From the result of the third action (Fig.V. 5), only one gateway was generated in order to deal with the parallel outgoing flows from *Manage flow* to *Move resource from producer to consumer,* and to *Manage accessibility of payment* services. So, the gateway is typed *parallel*. Besides, the *place order* service of *B* is linked with the *start event*. Also, the *Move resource from producer to consumer* and *Manage accessibility of payment* services are linked with the *end event*.

We assumed that the partners agreed on using this collaborative process (Fig.V. 6) to manage their collaborations. Thanks to the CPE tool, we also obtained an associated collaborative process model in XML to be used in Step 4.

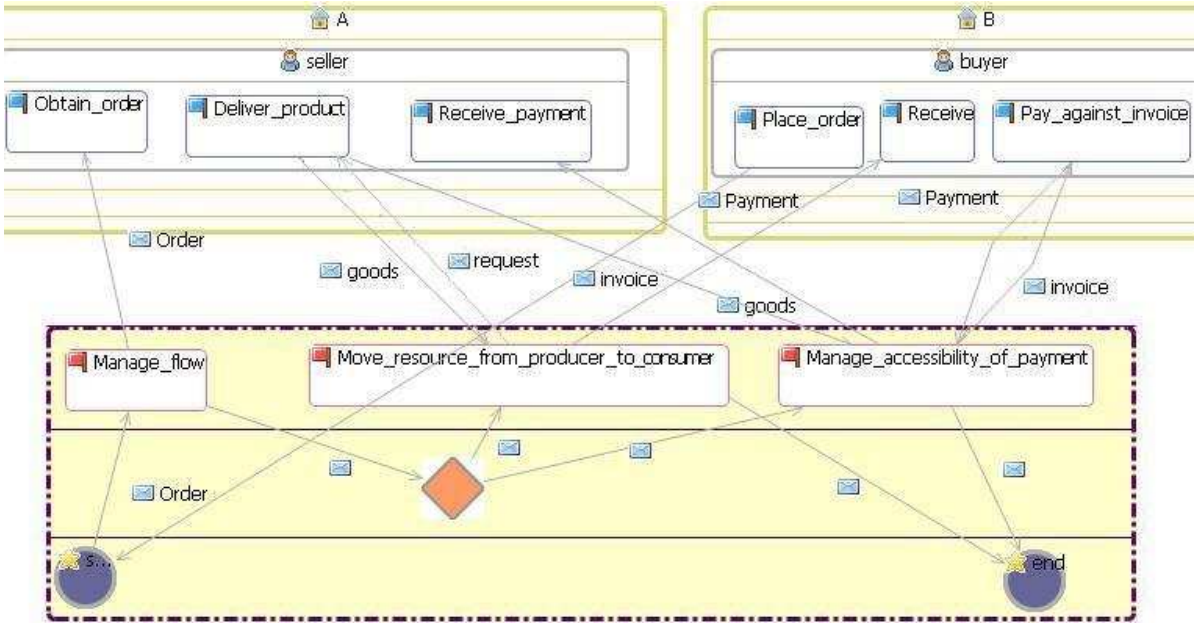


Fig.V. 6 Final collaborative process validated

In this collaborative process, A, B and MIS provide several services. Only the MIS has events, gateway, and flows (sequence flows) in addition, but there are none in A and B. There are flows (message flows) that communicate between MIS services and business services. We can see that this collaborative process corresponds to the meta-model of collaborative process (Fig.II.6).

If there are any disagreements between the partners on the collaborative process generated, it is possible to modify it at two levels, depending on how many modifications would be required:

- Superficial modification, if only a few changes are required (e.g. create new services, flows, events, gateways, etc.). These kinds of modification can be done with the CPE.

- Profound modification in the case of a wrongly generated process. This can be caused by the quality of knowledge captured from the business partners or the interpretation of this knowledge. Such modification requires a return to the first step (knowledge gathering and formalization) in order to interpret the knowledge captured by interview in a different way. This modification causes the regeneration of a new collaborative process model.

2.4. Step 4: BPMN construction and visualization

a. Objective

This step deals with the transformation of the collaborative process model validated by the partners into a BPMN relevant process.

b. Application

The ATL is the tool used to perform the transformation of the CPE's collaborative process model into the BPMN model. In Section 2.2.2.3 of Chapter 4 we discussed the transformation concept. We can summarize the requirements for performing such a transformation as follows:

- a source meta-model: XML meta-model
- a target meta-model: BPMN meta-model
- an initial model: the CPE's collaborative process model. Thanks to the CPE tool, the validated collaborative process diagram (Fig.V. 6) obtained from the fourth action of the previous step also provided its associated XML schema. This XML schema was used as the initial model of the ATL, and was to be injected into an XML source model.
- an ATL file, and its associated ASM file. The ATL file containing the morphing rules has been written in order to perform this transformation.

See the full implementation of ATL in Annex D. The meta-models are provided in Annexe E. The transformation was done programmatically via API Java using the *AtlLauncher* class. This class allows us to:

- inject the CPE's collaborative process model into an XML source model
- run the ATL transformation via the ASM associated file

c. Result obtained

Once the transformation has been done, we obtained a collaborative process model conforming to the BPMN meta-model. The resulting model contains:

- three pools for each of A, B, and MIS
- a number of activities, and lanes inside the A and B pools for representing business services and role respectively
- a number of activities inside the MIS pool for representing coordination services, gateways, and events
- a number of edges inside the MIS pool for representing sequence flows
- several messages for representing message flows transmitting inside the network

The following figures present the BPMN collaborative process model and its associated diagram visualised with the STP BPMN Modeller:

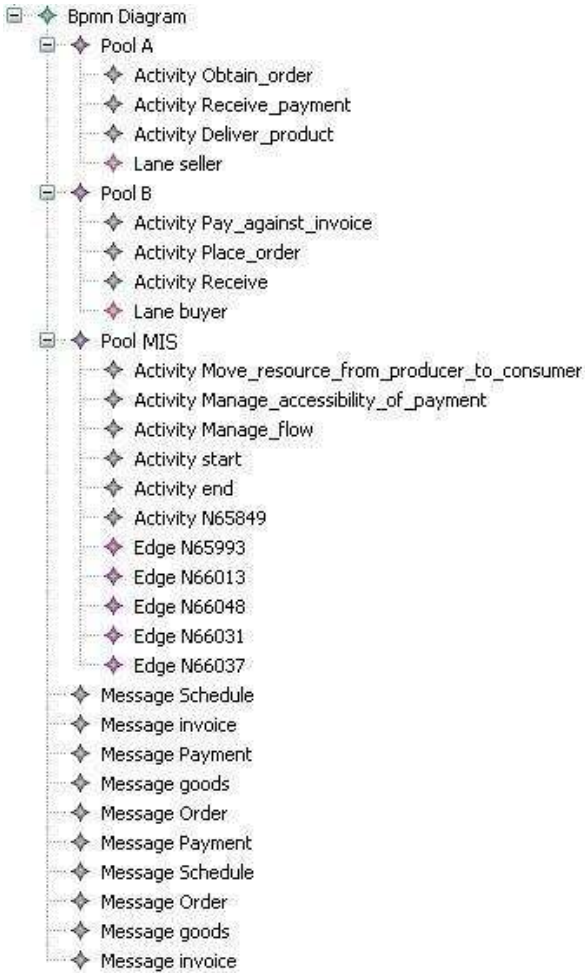


Fig.V. 7 Generated BPMN model (Ecore diagram)

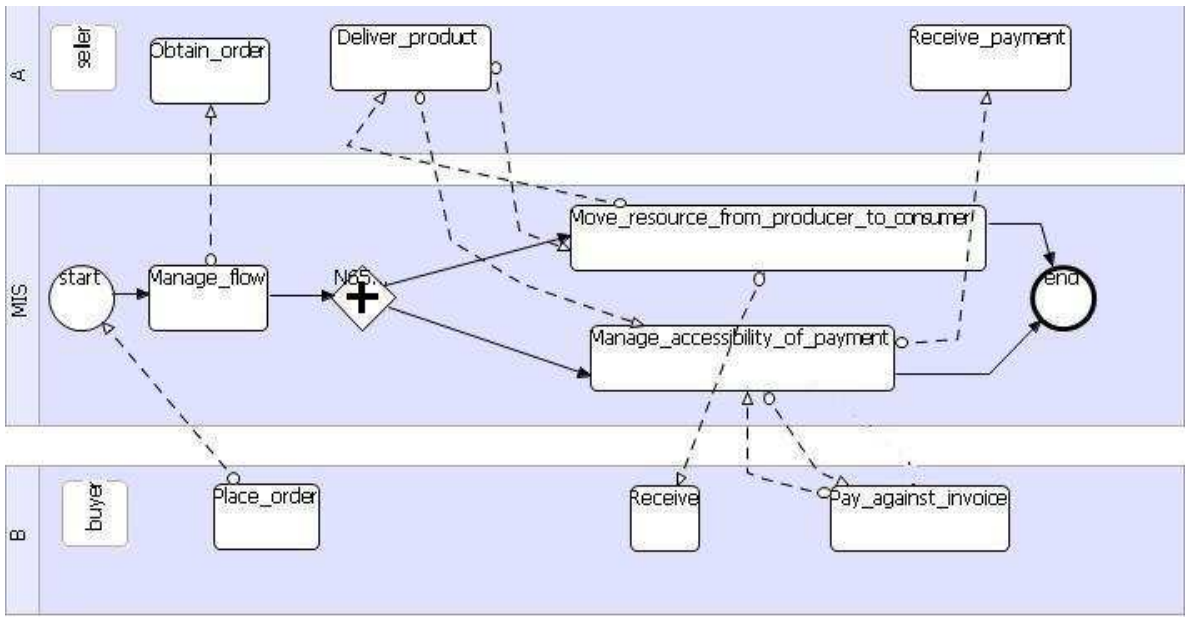


Fig.V. 8 BPMN model visualization with the STP BPMN Modeller

The BPMN process is similar to what we imagined at the beginning of this experimentation. The process starts at receiving an *order* sent from the *place order* service of *B*. This *order* is entered to the *manage flow* service for transferring to the *obtain order* service of *A*. Then, the *manage flow* service informs the other two services of the *MIS* in order to approve the *deliver product* service and prepare the delivery. The *deliver product* service of *A* ships the *goods* to the *receive* service of *B* via the *move resource from producer to consumer* service of *MIS*. The *deliver produce* service of A also sends the *invoice* to the *manage accessibility of payment* of MIS in order to forward to the *pay against invoice* of B. The *pay* service of *B* sends the *payment* to *A* via the *manage accessibility of payment* service of *MIS*.

3. Experimentation with a complex collaboration

We have previously presented the experimentation of a simple collaboration in order to show the mechanisms of our modelling approach. In this section, we introduce another experimentation which is more complex. We will present the result obtained at each step of the prototype.

3.1. Description of collaboration

The network has seven partners. Partner A buys materials for building planes on the order of the customers (C1 and C2). Partner A has three suppliers (F1, F2, and F3). Partners F1 and F2 supply the same materials. They negotiate between themselves about the quantities of the materials that each of them has to provide to A. Partner F3 is an intermediate supplier who buys materials from another supplier, F4, and sells them to A.

It is hard to imagine a corresponding collaborative process in this case because there are many interactions established between partners. It is, however, possible to achieve by the same method we used in the simple collaboration case shown previously.

3.2. Result obtained of Step 1: Collaborative network model

From the above description, we interpreted and represented it in the form of a collaborative network with the Network Editor as shown below:

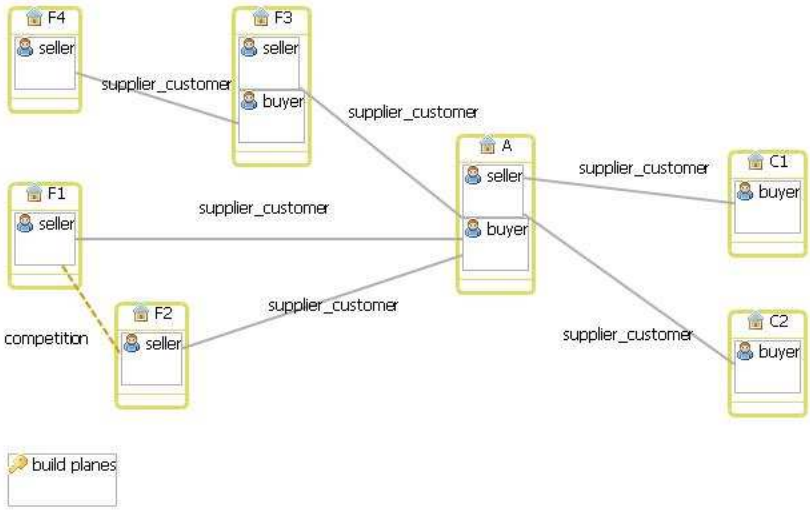


Fig.V. 9 Collaborative network model of a seven-partner network

Every partner establishes supplier-customer relationships between themselves, except the one between F1 and F2. As F1 and F2 sell the same products, they establish the competition relationship between them.

3.3. Result obtained of Step 2: Knowledge deduction

The collaborative network model shown previously was transformed and imported into the Knowledge Base. The SWRL rules were executed and new deduced knowledge was added automatically into the KB. The following figure illustrates the KB after executed the rules:

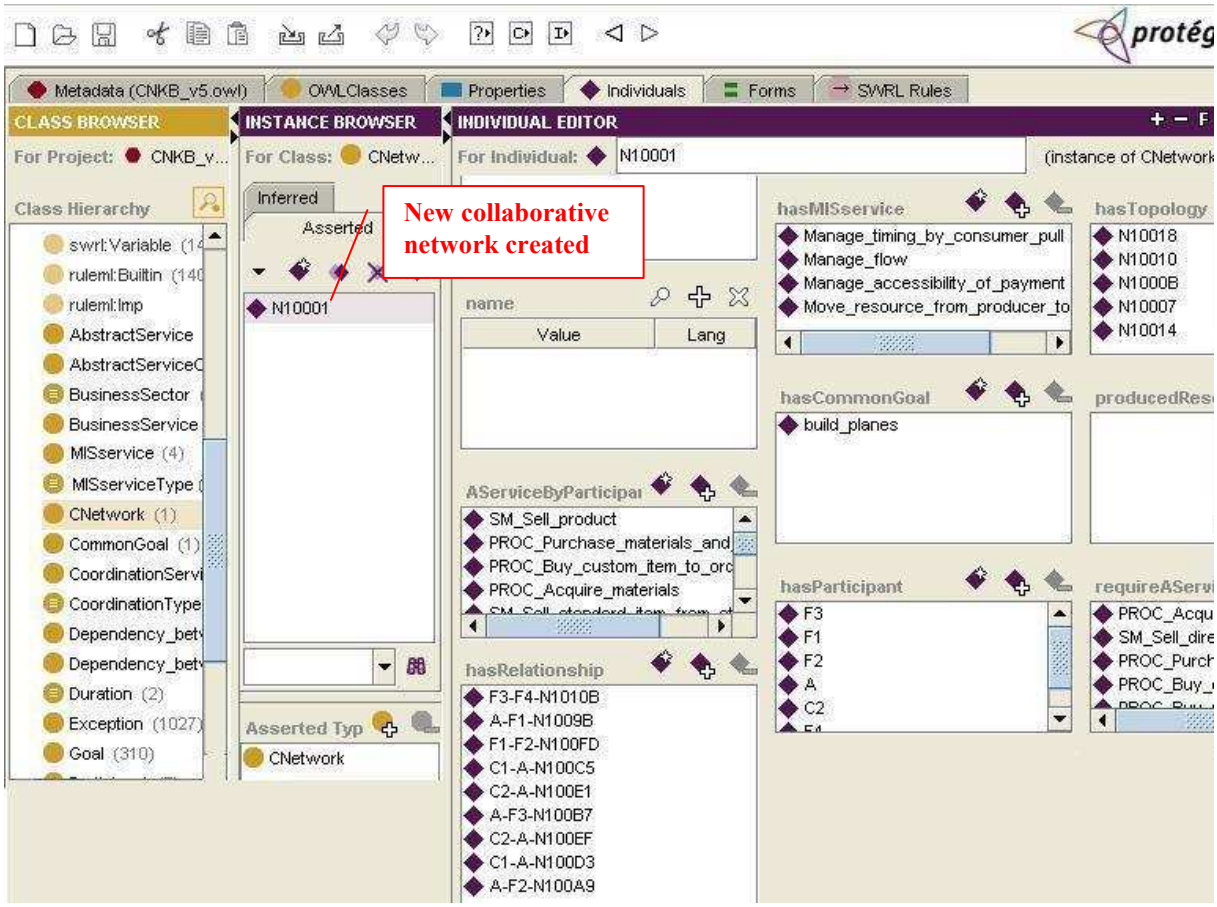


Fig.V. 10 Knowledge Base after executing the SWRL rules

3.4. Results obtained of Step 3: Collaborative process model

We followed the four actions as done in the first example. But, we show here only the results obtained from actions 2 and 4 which concern the first generated and the validated collaborative process models.

We started by extracting knowledge related to the network we are studying. As the network has seven participants, the services and the flows of data in this case are numerous. Thus, the SPARQL queries returned the large quantity of knowledge.

Then, the CPE manipulated and arranged the query results in the form of a collaborative process. Because of the high volume of query results, the processing time for this second action took a few hours. The very first obtained collaborative process visualized with the CPE is shown in Fig.V. 11. We have simplified the display of the above collaborative process using the CPE. The CPE offers the functionality that allows users to close their undesired elements, as shown below:

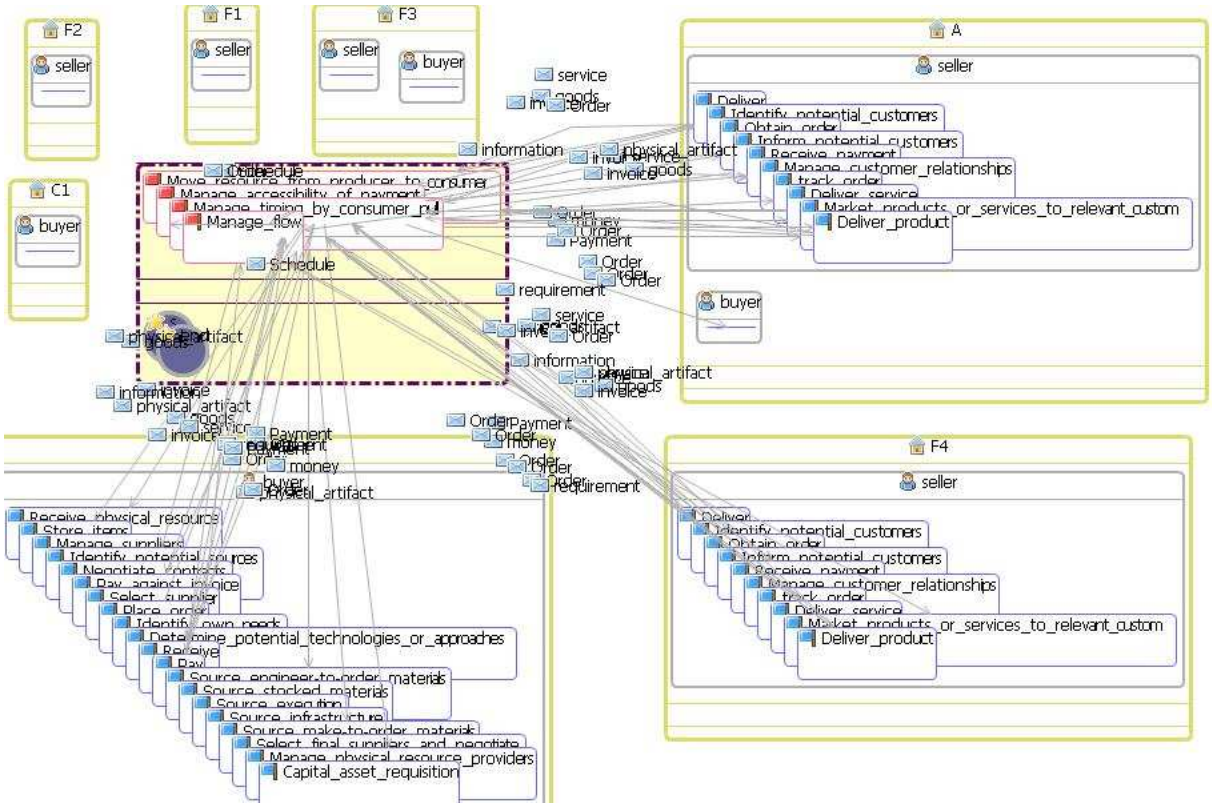


Fig.V. 11 Fussy collaborative process model of the 7-participant collaboration

The CPE's users worked with these seven participants in order to delete the worthless elements from the obtained collaborative process and also to organize the services. As we have mentioned in Section 2.2.3.1 of Chapter 4, up to now the users have to do such things manually. We realized that it is hard to deal with and requires much user effort because of the large number of flows and services in the process. For this reason we need to develop a checkbox of services to facilitate this action.

Once the undesired elements have been deleted from the collaborative process, the gateways and events were generated. We verified once again the correctness of these generations with respect to the meaning of flows and services. The following figure shows the final validated collaborative process model:

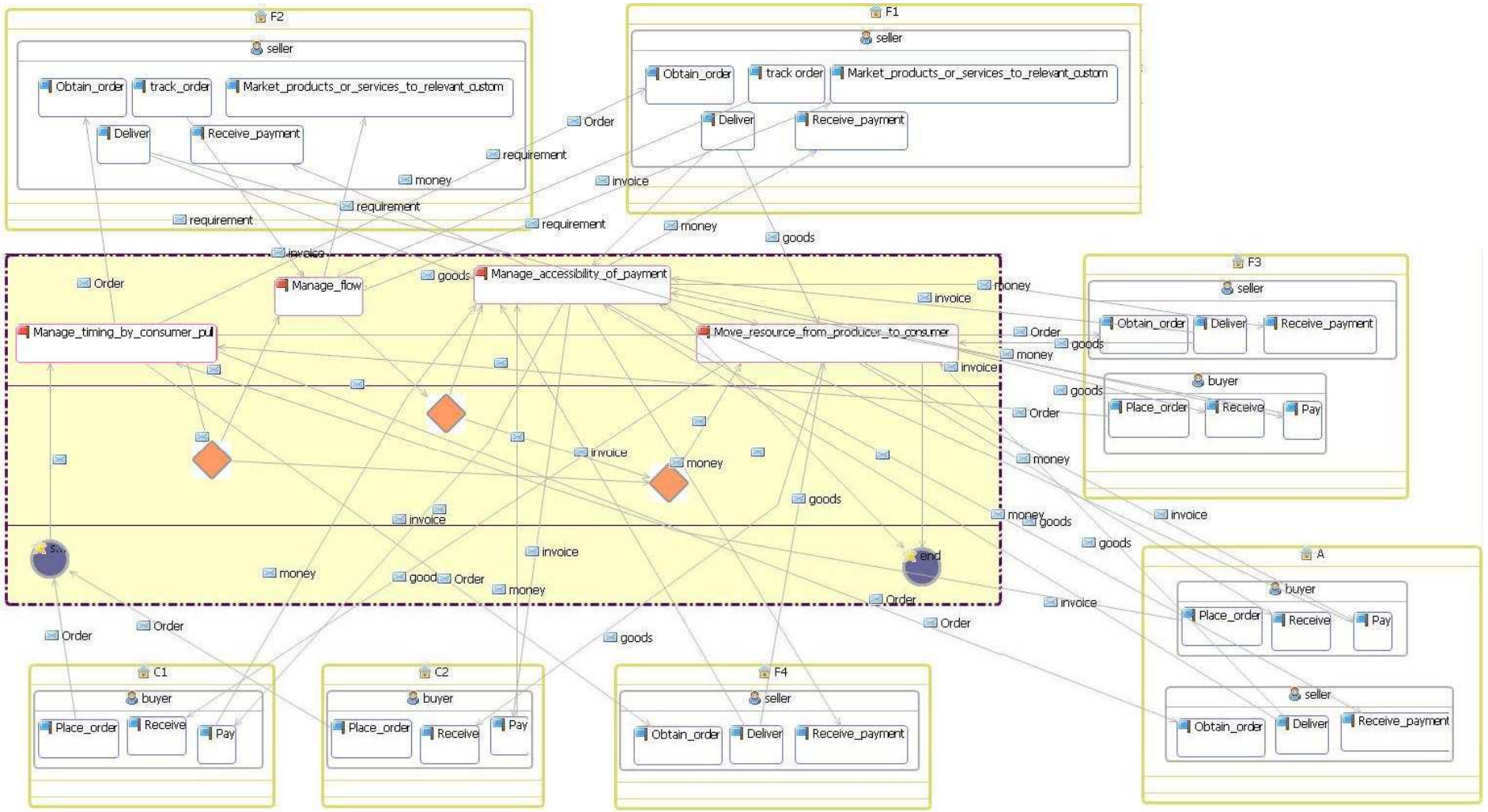


Fig.V. 12 Final validated collaborative process model of the 7-participant network

This collaborative process conforms to the requirements of the meta-model of collaborative process (Fig.II.6). There are only business services and roles in the partners' frames, and no flows inside. In the MIS frame, there are MIS items (MIS services, events, and gateways), and flows between them. The business services connect to the MIS items by the flows.

3.5. Result obtained of Step 4: BPMN collaborative process

From the above final validated collaborative process, we performed an ATL transformation and finally obtained the BPMN collaborative process as shown in Fig.V. 13:

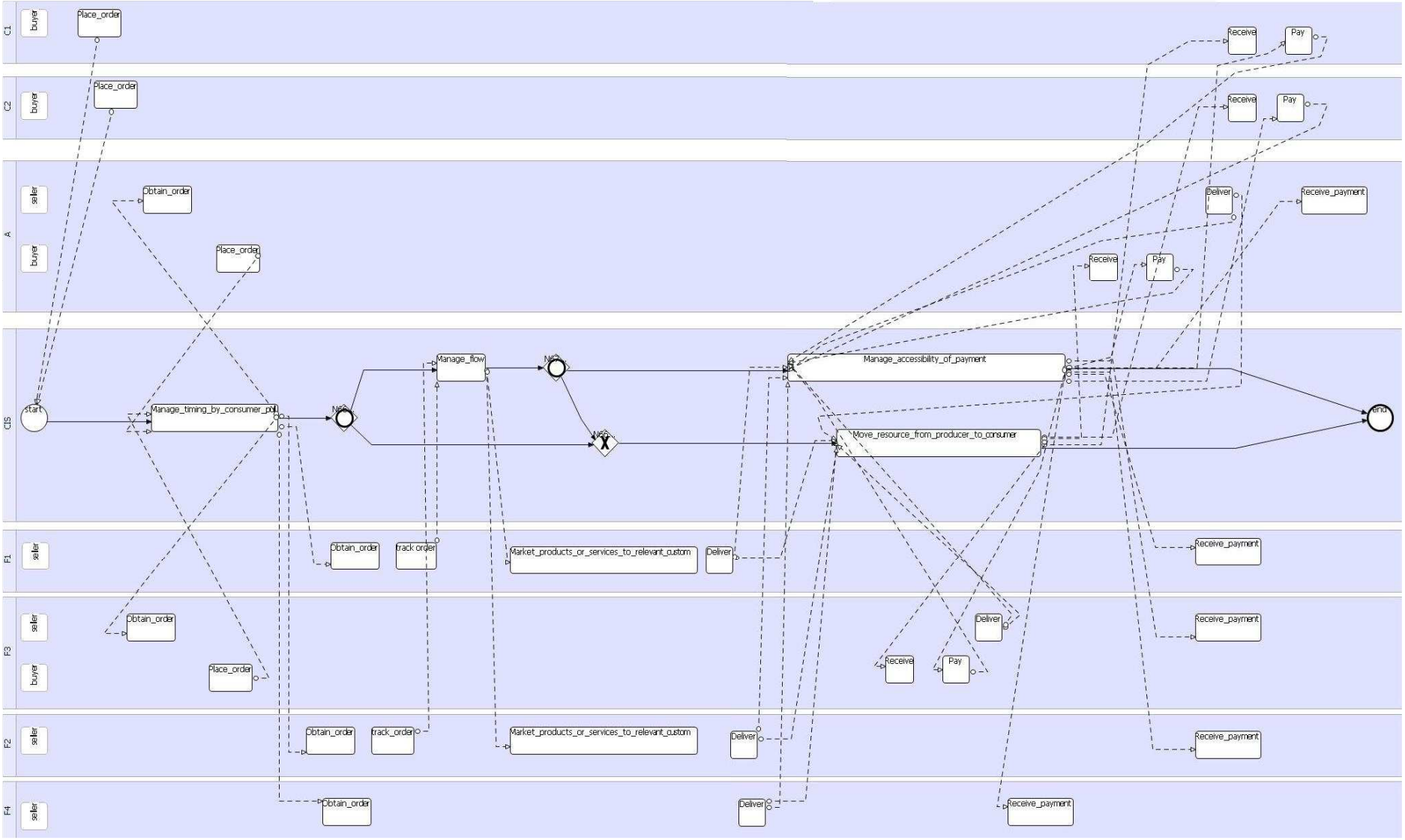


Fig.V. 13 BPMN collaborative process corresponding to Fig.V. 12

The process starts when one of the customers (*C1* and *C2*) places their *orders* to A via the *manage timing service* for buying the materials required for building planes. The *place order* service of *A* sends the *orders* of materials upon the requirements of *C1* and *C2* to the different suppliers (*F1*, *F2*, and *F3*).

As *F1* and *F2* provide the same materials, they need to organize between themselves in order to deliver the right quantities of materials to *A*. The *track order* and *market products to relevant customer* services allow *F1* and *F2* to negotiate, follow up, and deliver their *materials* in the right quantities to the right customers through the *manage flow* service of *MIS*. The *materials* are shipped from the *deliver* service of *F1* and *F2* to the *receive* service of *A* via the *move resource from producer to customer* service of *MIS*. The *deliver* service also sends the *bill* to the *pay* service of *A* via the *manage accessibility of payment* service of *MIS*.

Partner *F3* is an intermediate supplier who buys the materials ordered by *A* from its supplier, *F4*. So, *F4* delivers respectively the *materials* and *bill* to the *receive* and *pay* services of *F3* using the same MIS services as *F1* and *F2*.

Once *A* has obtained all *materials* required by *C1* and *C2*, *A* assembles the materials and builds the final products through its own private services. Then, *A* ships these *end products* to *C1* and *C2* according to the orders. Finally, *CI* and *C2 pay* the *bill* to the *receive payment* service of *A* via the *manage accessibility of payment* service of *MIS*.

4. Conclusion of the chapter

The prototype discussed in Chapter 4 was experimented with several collaborative situation case studies. The objective of these experiments is to test the prototype and validate our ontology-based approach.

In this chapter, we presented the experimentation with two collaborations: a simple collaboration of two partners, and a complex collaboration involving seven partners. Both collaborations are in a supplier-customer context. The experimentation followed these four steps: knowledge gathering and formalization, collaboration pattern deduction, specific collaborative process extraction, and BPMN construction. Each step was presented in this chapter providing objective, application, and result obtained at the end. The latter one is the most complicated because it is composed of four actions and required much effort from the user to attain the goal.

The experimentations allow us to realize that it is possible to define a corresponding collaborative process from the description of a simple collaboration, but it is not easy to do so in the case of complex collaboration. The prototype offers the tools that help users in specifying the collaborative process. But, several points still need to be taken into account in order to make it more convivial such as:

- When defining a collaborative network, users have to specify (at least) one role for each partner. The definition of roles has not been provided yet. So it is not easy to select a suitable one.

- The first generated collaborative process (Fig.V. 4 and Fig.V. 11) is extremely fussy. It is not understandable and hard to deal with. The enrichment of the CNO may add constraints to the deduction and make the results of deduction more precise and exact. Furthermore, we may need to develop an interface in order to facilitate the deletion of undesired services from the collaborative process.

- The processing time of the whole modelling process (Fig.V. 1) depends on the complexity of the collaboration case, particularly the transformation inside the third functionality (from the SPARQL results to the CPE).

- The Knowledge Base should store instances that cover a large area of business sectors. At present, the instances originally stored in the knowledge base come only from the MIT Process Handbook. This may limit our knowledge-based system and the collaborative processes generated from it. Thus, it is necessary to enlarge our actual Knowledge Base.

Chapter 6. Conclusions and Outlook

1. A Reminder of the framework

The scope of our work relies on the enterprise interoperability of information systems. To reach the ideal interoperability, we have to overcome three barriers: organizational, logic, and technological. The MDE provides a three-level approach (CIM, PIM, and PSM) for dealing with these three barriers. The previous works carried out in the MISE project address the logic and technological barriers, or PIM and PSM levels of the MDE.

This dissertation addresses particularly the implementation of collaborative processes in an inter-enterprise network context, which concerns the organizational barrier and CIM level. The work developed in this dissertation can link the works of EBM WebSourcing and the MISE project. Based on the perspectives of EBM WebSourcing, our work provides the design time editors (Network Editor and Collaborative Process Editor) which generate the collaborative process model to be used in the runtime platform. For the MISE project, the collaborative process model we generate can be the input for the development of the MIS.

The main objective of our work is to define a collaborative process that corresponds to the characteristics of collaboration together with the requirements of the involved partners of that collaboration. We decided to tackle this objective with several questions: “How to characterize collaboration?” and “What conceptual solution to use for modelling a collaborative process?”

The first question focuses on defining the collaboration space in which we are interested. We answer this question by studying the notion of collaboration, including its definitions and classifications. In addition, the knowledge retrieved from experience and the past collaborations of business partners allow characterising collaboration according to their perspectives. This knowledge is implicit. We developed an editor, called Network Editor, to facilitate the acquisition and formalization of this knowledge. Such knowledge is crucial since it influences the characterization. More quantity and better quality of knowledge captured leads to a more accurate characterization of the collaboration.

The second question concerns the removal of the organizational barriers. This is the main part of our work. We proposed to develop a knowledge-based system to define the collaborative. The framework of this system is enclosed by the previous works of the MISE project. The main constraints are: the uses of BPMN as the collaborative process modelling language and the meta-model of collaborative process to guarantee the conformity between CIM and PIM levels. Our knowledge-based system concerns the knowledge morphing from collaboration into collaborative process by using ontologies and deduction rules. To carry out the development of such a system, we are interested in studying the modelling approaches, concepts, and technologies for dealing with knowledge representation and reasoning.

2. Conclusions of the dissertation

The objective of this dissertation is to develop a knowledge-based system dedicated to specifying a collaborative process model specific to a given collaboration case. For this purpose, we propose to work at a higher abstraction level than the CIM to capture knowledge which allows us to characterize collaboration by basing it on the perspectives and experiences of business partners. We try to reuse existing knowledge about business processes from the MIT Process Handbook before moving down to the CIM level.

Our work concerns the characterization of collaboration, the solution for specifying collaborative processes, and the solution development. These points are summarized as follows:

2.1. Characterization of collaboration

So far we have discussed the approaches for inter-enterprise collaboration including the definitions, classification, and characteristics. Establishing collaboration means setting up a collaborative network of multi-enterprises.

A network consists of four principle components: node, link, relation, and flow [Poulin et al., 1994]. Nodes (e.g. individual, enterprise, group of enterprises, etc.) and links between nodes define the frame in which relations interact by defining the common objectives, the general type of partnership, and the functioning rule. Finally, flow is defined as the movement of resources (e.g. materials, information, etc.) between nodes. The main criteria for characterising and configuring a collaborative network are partners (or nodes), roles (e.g. seller, buyer, provider, maker, etc.), goals to be achieved, relationships (e.g. supplier-customer, competition, and group of interest) [Fombrun et al., 1982], topology of network (e.g. star, chain, and peer-to-peer) [Katzy et al., 2000], and dependency (e.g. flow, fit, and share) of resources between services [Crowston, 2003].

The precision of collaboration characterization strongly depends on the quantity and quality of the knowledge retrieved from business partners and leads us to obtain an appropriate collaborative process from our knowledge-based system.

2.2. Solution for specifying collaborative processes: Knowledge-based system

In this dissertation, a collaborative process is a process shared between independent parties in order to achieve common objectives.

Our main purpose is to define a collaborative process that corresponds to the characteristics of collaboration. The knowledge-based system has been developed in order to accomplish this task at the CIM level. The concepts and technologies for developing such a system are enclosed and constrained by the PhD of Touzi and the requirements of EBM WebSourcing. The BPMN has been chosen to be the collaborative process modelling formalism because it covers both organizational and partial information views of a process, as well as directly addressing the business process modelling. The meta-model of a collaborative process

(Fig.II.6) has been taken into account while developing the system because it guarantees the connectivity of our work and the PhD of Touzi.

In the system, there are two kinds of knowledge: static (domain knowledge) and dynamic (inference and task knowledge dealing with the reasoning process). Static knowledge can be represented by ontologies, while the dynamic one is represented by rules. Since our knowledge-based system concerns the knowledge morphing from collaboration into a collaborative process, it has been developed on the basis of OWL ontology and some SWRL rules. We created the Collaborative Network Ontology (CNO) which itself consists of a Collaboration ontology (CO) for collaboration features and a Collaborative process ontology (CPO) for collaborative process features, and rules. The CPO is mostly derived from the MIT Process Handbook [Malone et al., 2003]. It also integrates some important concepts of the collaborative process meta-model [Touzi, 2007], such as MIS services. These two ontologies are connected to each other by the semantic and structural links (rules) allowing the deductive reasoning. This deduction makes it possible to morph the knowledge. Such rules are executed over the Jess engine by manipulating instances stored in the Knowledge Base (KB). The instances concern the elements (e.g. business processes, resources, etc.) constituting collaborative processes, which mostly come from the MIT Process Handbook.

The knowledge-based system starts by receiving the knowledge on collaboration captured from business partners. This knowledge is formalized in the form of a collaborative network model and imported into the KB as a new instance. Then the rules are executed in order to deduce new knowledge. The deductions are performed on the basis of the imported knowledge about collaboration (corresponding to the CO) and the instances originally stored in the KB. Once the execution has been done, the new deduced knowledge is inserted automatically in the KB. This new knowledge is about the collaborative process (corresponding to the CPO) specific to the input collaborative network model. Finally, we extract this new knowledge from the KB and transform it into an appropriate collaborative process model conforming to the BPMN specification.

However, the solution we proposed is not the only one solution for coping with the design of collaborative process model and the interoperability of information systems. We tried to emphasize the use of ontologies and deduction rules in order to propose a well-defined solution based on knowledge engineering.

2.3. Open source prototype for implementing the solution

We developed a prototype dedicated to support the above solution on a knowledge-based system in order to facilitate the collaboration between business partners. Many open source technologies have been used to fulfil the development. We distinguish four principle functionalities each of which has specific requirements:

- Knowledge gathering functionality concerns the acquisition and formalization of knowledge expressed by business partners. We developed the Network Editor (NE) to facilitate this functionality.

- Collaboration pattern deduction functionality is the most important part of the system. We developed the ontology-based approach (presented in the Chapter 3) as a means of accomplishing this functionality. The Knowledge Base (KB) has been developed on the basis of such an approach for storing instances and performing deductions.

- Specific collaborative process extraction and visualisation functionality is the most complicated functionality and requires much more user effort than the others. We extract the knowledge on collaborative process from the KB by SPARQL queries. This extracted knowledge is organized in the form of a collaborative process model and visualized with the Collaborative Process Editor (CPE).

- BPMN construction functionality focuses on representing the collaborative process model obtained from the previous functionality in the form of a BPMN model. The ATL transformation is introduced for dealing with this issue. The real BPMN process is visualized with the STP BPMN Modeller. This tool is provided under the Eclipse Public License (EPL).

The prototype works in a semi-automatic way. Some of these functionalities require user efforts to accomplish, especially the first and third ones. The third functionality is the most complicated one because we implemented several technologies to accomplish its objectives. At the current stage, the prototype works as we expect, but it still has several inconvenient points that should be improved. These are discussed in the next section.

3. Perspectives

3.1. Perspectives on the current solution and the prototype

In Chapters 3 and 4 we have discussed the concepts, goals and use of technologies for constituting the knowledge-based system to support the design of collaborative processes. The prototype of this system which has been developed consists of four tools. The principal functionalities provided by each tool have also been presented. However, the prototype is not yet complete due to the complexity of the approach, development concepts as well as technologies. Some limitations and perspective works are highlighted as follows:

- The concepts, relations, and restrictions of the actual CNO as well as the deduction rules can be enriched. At the current stage, the deduction and extraction of knowledge from the KB (the second and third functionalities) bring out a quantity of knowledge that is related to the input collaborative situation. The first collaborative process model obtained from this generation is however really complex and hard to deal with (Fig.V.4). The improvement of the CNO and the rules may put more constraints on the deduction and make the deduction results more precise and exact. The rules can be improved at the concept levels of common goal, topology, and continuous and discontinuous relationships, which we have not significantly taken into account at this stage.

- The generation of gateways and events defined in the third functionality is not yet complete. Up to now, this functionality can generate only the start and end events and four types of gateway (parallel, event-based exclusive, data-based exclusive and data-based inclusive gateways) which are frequently used in BPMN processes. We have not yet included for example, message event, time event, complex gateway, etc. Thus, we may need to enrich this generation concept by taking into account these missing elements.

- The instances originally stored in the KB come only from the MIT Process Handbook. This makes our knowledge-based system and the collaborative process models generated from it limited. Thus, we need to enlarge the KB by storing more instances coming from other sources, such as collaboration use cases, creating new business services by partners from scratch, etc.

- The prototype itself can be improved to be more user-friendly in terms of user interfaces, design concept, and functionalities. For example, at the current stage, the definition of roles is not yet provided which makes it hard for users to select a suitable one. The prototype should provide more human aids, such as a tool manual, ability to search or browse by keywords, tags, etc.

- Even if the current prototype is made to be connectable to the Translator V1.0, these two tools have been implemented separately. The future work (starting PhD) will be to create a multi-view integrated framework in order to provide a loose approach (to avoid the tight aspect of the CIM-PIM-PSM drive).

These limitations are not easy to deal with and overcome rapidly. Some of them need to be investigated in detail. This investigation will lead to develop and open-up new visions for future work.

3.2. Event-based approach for improving the current solution of collaborative process definition

An event-based approach can be another way to improve our current solution. We can enhance our modelling mechanism by integrating this event-based approach to better define a collaborative process.

A collaborative process includes a set of services executed by several partners that take over specific roles. Our current solution deals with the service sequencing through the concept of dependency of resources between services. When we find two services where the service input of one is the same as the service output of the other, we link these two services together. Once we have obtained the dependencies we generate events. Although this solution takes into account the event aspect, this is not enough, as an event is an important aspect for making the process dynamic and flexible.

The fundamental concept in the event-based approach is, of course, an event. An event is a notable thing that happens inside or outside the enterprise [Michelson, 2006]. It can be defined as a significant change in the state of a system or an environment [Mani Chandy,

2006]. Coming back to our case, the event generation concept that we now use is based on the flow but not on the states of resources, conditions or trigger events which are the crucial patterns of event. We can adopt the event-based approach to improve our current event generation and service sequencing. The event patterns are described as event-condition-action (ECA) rules. For example:

- Event: request by buyer to send materials
- Condition: new purchase order is received and not yet processed (purchase order is in the received or non-processed state)
- Action: send purchase order to the delivery service (purchase order is in the processed state)

The ECA rule can be expressed as follows: when *event* is produced, if *condition* is satisfied, then *action* will be performed [Bouslimi et al., 2008]. The implementation of such an event-based approach may impact on the work of this dissertation. The generated collaborative process models will be more dynamic, complete and concrete because we will take into account the event patterns during the design of the collaborative processes. When new events happen or change the collaborative process definition changes accordingly. Thus, this approach may offer flexibility to our current solution.

3.3. Mapping identified services on the obtained BPMN collaborative process and real services of business partners

Another fact we have found during the implementation of our current knowledge-based solution concerns the BPMN collaborative process generated from the system. As our KB is populated only by the business processes of the MIT Process Handbook, the BPMN collaborative process model we obtain at the end of our system is a generic process, not the real process. This means all the services appearing in the generated processes are the Process Handbook's services, not the real partners' services. Mapping of the generated services with the real ones is needed. Such mapping relies on both syntactic and semantics. If we can achieve this mapping, the conceptual (semantic) barrier of the EIF framework will be removed (Fig.I.7).

The syntactic and semantic mapping of services may be done once we have obtained the SOA-based logic model of the MIS at the PIM level and before transforming this logic model into the technological model at the PSM level (Fig.VI.1). We intend to do this mapping as late as possible because we want to keep the generic collaborative process (BPMN) as long as possible in order to facilitate any modifications and agility of the whole system (including the three levels of the MDE). To accomplish this mapping, we may require a service registry.

A service registry is a crucial element in a SOA to simplify and automate searching for an appropriate service [Durvasula et al., 2006b]. Each service provider (business partner) must define which services to expose and its semantic description and publishes them to the service registry. The services appearing on the obtained BPMN process will be mapped to the real services available in the registry. In the early 2009, a master thesis will be in charge of this subject.

3.4. On-going work on the agility of the MIS

As collaboration always develops throughout its life cycle, the MIS should be reactive and flexible in order to better respond to any changes. The agility aspect has been proposed in the context of the ISyCri project (ANR/CSOSG2006) to deal with the interoperability of systems in a crisis situation. This project was launched in May, 2007.

According to [Truptil, 2008], in a crisis situation (natural disaster, industrial accident, etc.), several partners have to act simultaneously to solve the emergency situation. Their coordination in such a context is crucial. This project proposes to tackle this point by addressing the agility issue: responsiveness of the network (ability to act rapidly and efficiently), and flexibility of the obtained system of systems (ability to evolve and follow the changing situation). A PhD dissertation was created in late 2007 by S. Truptil in order to work on this issue. As the ISyCri project also addresses the interoperability of information systems, this new dissertation adopts the works (the knowledge-based system and the MIS concept) of the MISE project and adds the agility aspect into the MIS. This dissertation is crucial and can not only be applied to the crisis situation in particular but also to the generic context of the MIS. If we position this new dissertation on the MDE approach it concerns every level (CIM, PIM, and PSM) of the MDE. This work uses an iterative approach in the design process with precise looping criteria.

3.5. Positioning all related works and perspectives based on the MISE context

Since the beginning of this dissertation, we have positioned the previous works of the MISE project on the MDE approach (Fig.I.3). Here at the end, we re-position all the related works and also the perspectives described in this last section, but this time on the "double Y" developed by [Touzi, 2007]. This double Y is transformed from the classical Y of the model-driven approach by adding a new logic branch. This branch deals with the logic meta-modelling induced by the model transformation step at the PIM level.

Fig.VI.1 illustrates the positioning of the previous works of the MISE project, the agility of MIS carried out by Truptil, the mapping between generated services and real services, as well as the event-based approach. The previous works of the MISE project are represented with the full line, while the last three points are represented with the dashed lines:

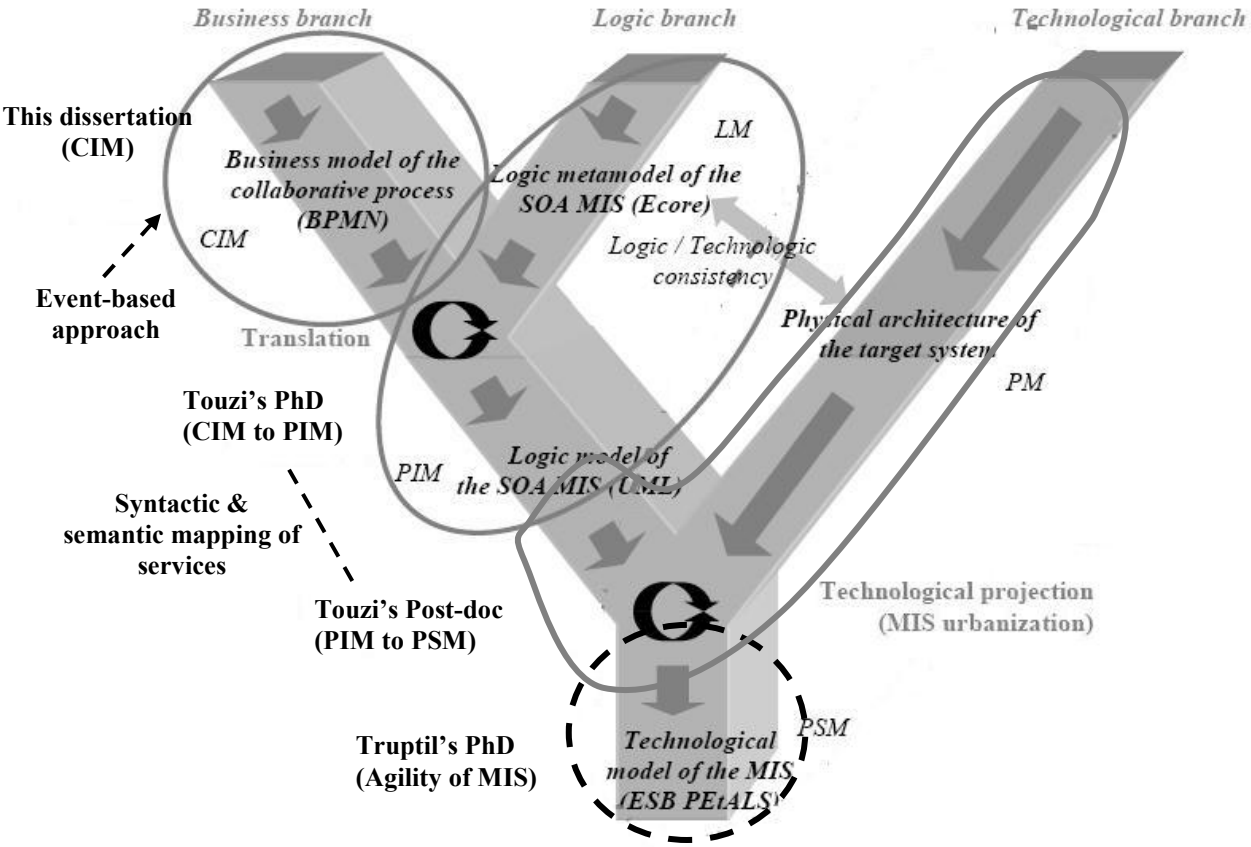


Fig.VI. 1 Global overview of the MISE project and perspectives [Touzi, 2007]

The integration of these works will constitute a consolidation of the MDE approach to fully support inter-enterprise collaboration. Ultimately this integration will enable us remove the three barriers (organizational, technological, and conceptual) to enterprise interoperability and to reach the maximum collaboration capacity where the interoperable enterprises are seen as a single entity.

3.6. Positioning the works developed in this dissertation based on the perspectives of EBM WebSourcing

Here we position the tools developed in this dissertation on the perspectives of EBM WebSourcing. We use the same perspective schema as shown in the general introduction. The on-going work of the company will be also introduced and positioned on this schema. The figure below illustrates the tools and on-going work:

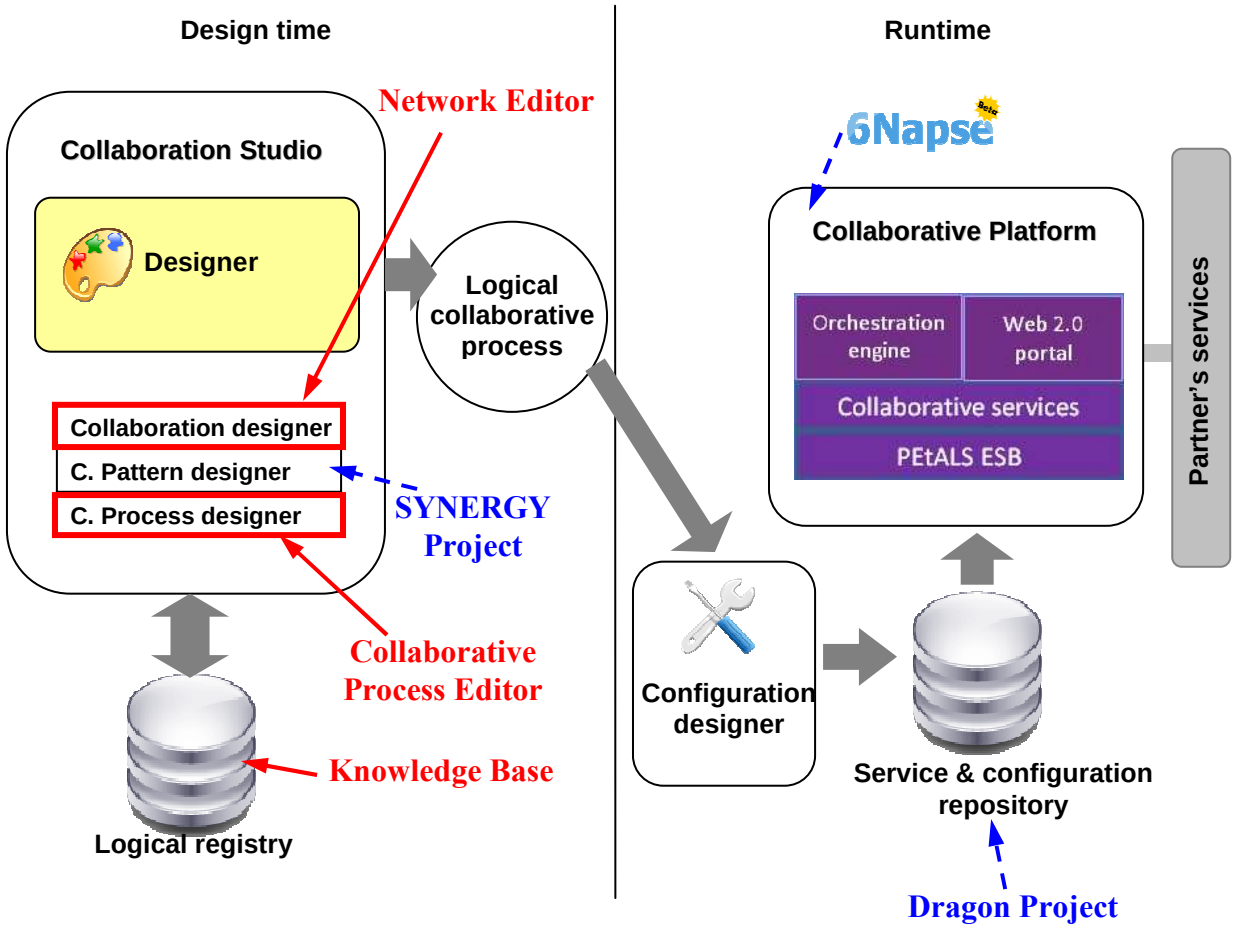


Fig.VI. 2 Perspectives of EBM WebSourcing and on-going work

6Napse - an enterprise collaborative platform

Since 2007, 6Napse has been developed at EBM WebSourcing. Its goal is to provide an intuitive, social and collaborative platform to companies. Whereas most of the current available platforms are focused on data or users, 6Napse is collaboration-centric. Collaboration in 6Napse involves at least two people from two different companies working together for a specific purpose: file exchange, services usage, commercial procedure, etc.

6Napse, [Rajsiri et al., 2008], follows a few basic concepts in order to provide all required functionalities to companies. These functions are described below:

- Social network: There are two fundamental aspects in a social network: user identity and the social glue. In 6Napse, users are registered as employees, and belong to companies. Users may only share information about their business profile, but all extra or public information will be skipped. When interactions (chat, requests, actions, etc.) take place between users, partnerships will be established between companies. The social aspect of the platform mainly resides in collaborations: companies subscribe in order to find partners and to be found by exposing their available services. The network grows up with collaborations and registered businesses.

- Collaborative platform: The whole platform is collaboration-centric which means all activities, directly or indirectly, deal around collaboration. 6Napse proposes two ways to collaborate with another company: 1) by using platform native services (e.g. mail, global news messages, advertisement or shared space), or 2) thanks to the services provided by companies (e.g. a company specialising in language translation may register on the platform and offer a service that translate any invoice from English/French to French/English). The second kind of services allows other registered companies to use the exposed services and collaborate with the company that exposes those services.

The 6Napse platform is connected to the main repository that acts like any other server. The main repository is in charge of synchronization between all secondary servers hosted by registered companies. Each server will have a PEtALS ESB node embedded in order to provide an efficient tool for companies to expose their services.

Up to now, 6Napse is still an on-going project. The beta version will be released this year.

SYNERGY project

SYNERGY is a research project funded by the European Commission with the aim of developing dynamic and adaptive knowledge management systems and services to enable virtual organizations (VOs) [Popplewell et al., 2008]. This project has been launched since February, 2008. The SYNERGY consortium has eight participants who are: scientific and technological expertise, have experience in practical requirements and application domain for collaboration, and have capacities for exploitation scientifically and commercially.

SYNERGY stands for Supporting highlY-adaptive Network Enterprise collaboration thRouGh semanticallY-enabled knowledge service. The SYNERGY project aims to enhance effective knowledge sharing between organizations and to stimulate collaboration by developing a highly intelligent technological system based on collaboration patterns and knowledge flows. The goal is to enhance support of the networked enterprise in the successful and timely creation of, and participation in, collaborative VOs by providing an infrastructure and services to discover, capture, deliver and apply knowledge relevant to collaboration creation and operation.

EBM WebSourcing contributes the integration and implementation capability to the SYNERGY project. We are mainly concerned with the prototype development, open source exploitation, and collaboration pattern editor.

In the SYNERGY context, we are in charge of developing a collaboration pattern editor based on an event-based approach. This editor helps VO members to define and simulate collaboration patterns that they are going to use during the life cycle of their VO. The editor will be used during the design time. The output of the pattern editor will be executed and monitored by the collaboration pattern assistant (CPA) during the runtime. The CPA will be developed by ICCS (Institute of Communication and Computer System).

Based on the perspectives of EBM WebSourcing (Fig.VI.2), the collaboration pattern editor will be used in design time together with the Network Editor and Collaborative Process Editor which have been developed in this dissertation.

Dragon project

The Dragon SOA governance project will provide a full set of software tools for SOA governance[20]. These tools will allow us to describe syntax and semantic of services and manage the whole life cycle of services from the design phase to the execution phase. The services will be deployed in PEtALS and we can observe what is going on along its life cycle. The objectives are to provide:

- A governance registry/repository in which service developers can discover, deploy, document and reuse services in collaboration environment. Validation, de-duplication, versioning and enforcement take place in this context.

- A service platform interface (JBP interface with PEtALS) which focuses on controlling deployment through approval processes and applying runtime access-control policies to services.

Version 1.0 will be released by the end of 2008 and will include: PEtALS service registry/repository core and GUI with services meta-data and artefacts management such as WSDL, XSD, policy specification, BPEL process specifications, UDDI integration, and advanced integration within PEtALS service platform.

[20] SOA governance means “the ability to organize, enforce and re-configure service interactions in an SOA”.

Acronyms

AIAI	Artificial Intelligence Applications Institute
AIF	ATHENA Interoperability Framework
API	Application Programming Interface
ATL	Atlas Transformation Language
BO	Business Opportunity
BPEL	Business Process Execution Language
BPM	Business Process Management
BPMI	Business Process Management Initiative
BPMN	Business Process Modelling Notation
BPMO	Business Process Management Ontology
BPMS	Business Process Management System
CIM	Computer Independent Platform
CO	Collaboration Ontology
CPO	Collaborative Process Ontology
CNO	Collaborative Network Ontology
CNO (ECOLEAD)	Collaborative Networked Organization
CPE	Collaborative Process Editor
DL	Description Logic
DSM	Domain Specific Modelling
DSL	Domain Specific Language
EAI	Enterprise Application Integration
EDI	Electronic Data Integration

EIF	Enterprise Interoperability Framework
EMF	Eclipse Modelling Framework
ESB	Enterprise Service Bus
GEF	Graphical Editor Framework
GMF	Graphical Modelling Framework
HTML	Hypertext Markup Language
IDEAS	Interoperability Development for Enterprise Application and Software
ISU	Interoperable Service Utility
KAON	KArlsruhe Ontology
KB	Knowledge Base
MDA	Model Driven Approach
MDE	Model Driven Engineering
MIS	Mediation Information System
MISE	Mediation Information System Engineering
NE	Network Editor
OKBC	Open Knowledge Base Connectivity
OMG	Object Management Group
OWL	Web Ontology Language
PIM	Platform Independent Model
PSL	Process Specification Language
PSM	Platform Specific Model
RDF	Resource Description Framework
RDFS	RDF Schema

RuleML	Rule Markup Language
SOA	Service Oriented Architecture
SWRL	Semantic Web Rule Language
TOVE	TOronto Virtual Enterprise
UML	Unified Modelling Language
UMM	UN/CEFACT Modelling Methodology
URI	Uniform Resource Identifier
VBE	Virtual Breeding Environment
VE	Virtual Enterprise
VO	Virtual Organization
W3C	World Wide Web Consortium
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XSD	XML Schema
XSLT	XML Transformation

References

[Aamodt et al., 1995] Aamodt A., Nygard M. Different roles and mutual dependencies of data, information, and knowledge - an AI perspective on their integration, Data and Knowledge Engineering, vol. 16, p. 191-222, 1995.

[Alessandro et al., 2007] Alessandro M., Klein M., Gianluca E. Metrics-Based Process Redesign with the MIT Process Handbook, Knowledge and Process Management, Volume 14 N.1, pp 46–57, Published online in Wiley InterScience, (www.interscience.wiley.com) DOI: 10.1002/kpm.269, 2007.

[APFA, 2004] Association pour Promouvoir le Français des Affaires, www.presse-francophone.org, 2004.

[ATHENA, 2004] ATHENA Integrated Project, « Requirement for interoperability framework, product-based and process-based interoperability infrastructures, interoperability life-cycle services », ATHENA deliverable A4.1, 2004.

[ATL-HowTo] http://wiki.eclipse.org/ATL_Howtos#How_do_I_launch_transformations_programmatically.3F

[ATL manual, 2006] ATLAS Group. ATL User Manual Version 0.7, LINA & INRIA, February, 2006.

[Bénaben et al., 2008] Bénaben F., Touzi J., Rajsiri V., Truptil S., Lorré JP., Pingaud H. Mediation Information System Design in a Collaborative SOA Context through a MDD Approach, Proceedings of the First International Workshop on Model Driven Interoperability for Sustainable Information Systems (MDISIS'08) held in conjunction with the CAiSE'08 Conference, Montpellier France, 2008.

[Berners-Lee et al., 2001] Berners-Lee T., Hendler J., Lassila O. The Semantic Web. Scientic American, May, 2001.

[Bernstein, 1998] Bernstein, A. The Product Workbench: An Environment for the Mass-Customization of Production-Processes, Workshop on Information Technology and Systems (WITS), Helsinki, Finland, 1998

[Bernstein, 2000] Bernstein, A. How can cooperative work tools support dynamic group processes? Bridging the specificity frontier, Computer Supported Cooperative Work, ACM Press, Philadelphia, PA, 2000

[Berre et al., 2007] Berre AJ., Elvesæter B., Figay N., Guglielmina C., Johnsen SG., Karlsen D., Knothe T., Lippe S. The ATHENA Interoperability Framework, Establishing the Foundation of Collaborative Networks, Springer, I-ESA Conference, 2007.

[Boley et al., 2001] Boley H., Tabet S., Wagner G. Design Rationale of RuleML: A Markup Language for Semantic Web Rules. In: International Semantic Web Working Symposium (SWWS), 2001.

[Boley et al., 2002] Boley H., Grosof B., Sintek M., Tabet S., Wagner G. RuleML Design, Version 0.8, http://www.ruleml.org/indesign.html, 2002.

[Bouslimi et al., 2008] Bouslimi I., Hanachi C., Tout H., Ghédira K. A coordination framework for Cooperative Information Gathering, Int. J. Advanced Intelligence Paradigms, Vol. 1, No. 1, p.60–79, 2008.

[Bouzguenda, 2006] Bouzguenda L. Coordination multi-agents pour le Workflow interorganizationnel lâche, thèse de doctorat, IRIT, 2006.

[BPMI, 2004] BPMI, Business Process Modelling Notation (BPMN), version 1.0, May 3rd 2004

[BPMN] Business Process Modelling Notation (BPMN): www.bpmn.org

[BPMN, 2004] Specification of Business Process Modelling Notation (BPMN), Version 1.0, May 3, 2004.

[BPMO Tutorial] BPMO tutorial defining a private business process in a knowledge base. Available: http://www.bpiresearch.com/BPMO_Tutorial.pdf., 2003

[BPMS Watch] BPMS Watch: www.brsilver.com/wordpress

[Bray et al., 2004] Bray T., Paoli J., Sperberg-McQueen CM., Maler E., Yergeau F., Cowan J. Extensible Markup Language (XML) 1.1 (W3C recommendation). http://www.w3.org/TR/2004/REC-xml11-20040204/. 2004.

[Browne et al., 1999] Browne J., Zhang J. Extended and virtual enterprise-similarities and differences, International journal of Agile Management Systems, vol. 1, p. 30-36, 1999.

[Burn et al., 1999] Burn JM., Marshall P., Wild M. Managing Changes in the Virtual Organization. Proceedings of the Seventh European Conference on Information Systems 40-54, Copenhagen Business School, Copenhagen, 1999.

[Camarinha-Matos et al., 2005] Camarinha-Matos L., Afsarmanesh H. Collaborative networks: A new scientific discipline. Journal of Intelligent Manufacturing, 16(4), 439-452, 2005.

[Camarinha-Matos et al., 2006] Camarinha-Matos L., Afsarmanesh H. COLLABORATIVE NETWORKS - Value creation in a knowledge society, In Proceedings of PROLAMAT'06 (Springer) – Shanghai, China, 14-16 June, 2006.

[Chen et al., 2006] Chen, D., Dassisti, M., Elvaester, B.: Interoperability Knowledge Corpus, Intermediate Report. Deliverable DI.1b, Network of Excellence InterOp, Contract No.IST-508011, 2006.

[Corcho et al., 2002] Corcho O., Fernandez-Lopez M., Gomez-Perez A. Methodologies, tools and languages for building ontologies. Where is their meeting point?, Elsevier, Data & Knowledge Engineering 46, p. 41–64, 2002.

[Crowston, 1994] Crowston, K. A Taxonomy of Organizational Dependencies and Coordination Mechanisms (Working paper No. 3718-94): Massachusetts Institute of Technology, Sloan School of Management, 1994

[Crowston, 2003] Crowston K. A taxonomy of organizational dependencies and coordination mechanisms. In T. W. Malone, K. Crowston & G. Herman (Eds.), The Process Handbook, p. 85–108, Cambridge, MA: MIT Press, 2003.

[Czarnecki et al., 2003] Czarnecki K., Helsen S. Classification of Model Transformation Approaches, Workshop on Generative Techniques in the Context of MDA, OOPSLA, 2003.

[D1.1 Synergy, 2008] Deliverable D1.1 State of the Art and As-Is Analysis, SYNERGY-01-20080730-D1.1-Final, Version 6.0, 2008.

[D8.1 InterOp, 2004] Deliverable D8.1 State of the art and state of the practice including initial possible research orientations, InterOp, version 1.2, 2004.

[D.A.2.1 Athena, 2006] D.A.2.1 Cross-Organizational Business Process requirements and the State of the Art in Research, Technology and Standards, Work package A2.1 Requirements Analysis & State of the Art, Version 2.0, 2006.

[Dellarocas, 1996] Dellarocas, C. A coordination Perspective on Software Architecture: Towards a design Handbook for Integrating Software Components, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1996

[DoD, 2001] Department of Defense. Department of defense Dictionary of Military and Associated Terms, United States of America Department of Defense, 2001.

[Durvasula et al., 2006a] Durvasula S. et al., SOA Practitioners' Guide Part 1: Why Services-Oriented Architecture?, 2006.

[Durvasula et al., 2006b] Durvasula S. et al., SOA Practitioners' Guide Part 2: Reference Architecture, 2006.

[ECOLEAD] ECOLEAD (FP6.IP 506958). D21.3 Establishing VBE common ontologies, 2004-2008

[Etien, 2006] Etien A. Ingénierie de l'alignement : Concepts, Modèles et Processus, La méthode ACEM pour l'alignement d'un système d'information aux processus d'entreprise, Université Paris I – Sorbonne, Mars, 2006.

[Fombrun et al., 1982] Fombrun C.J., Astley W.G. The telecommunication community: an institutional overview, Journal of Communication, 32 (4), 56-68, 1982.

[Fox, 1992] Fox M.S. The TOVE Project: A Common-sense Model of the Enterprise, Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Belli, F. and Radermacher, F.J. (Eds.), Lecture Notes in Artificial Intelligence # 604, Berlin: Springer-Verlag, 25-34, 1992

[Frayret et al., 2003] Frayret JM., D'Amours F., D'Amours S. Collaboration et outils collaboratifs

pour la PME manufacturière, Cefrio, October, 2003.

[Gangemi et al., 2002] Gangemi A., Guarino N., Masolo C., Oltramari A., Schneider L. Sweetening Ontologies with DOLCE. In EKAW-02: Proceedings of the 13th Int. Conference on Knowledge Engineering and Knowledge Management. Ontologies and the SemanticWeb, pages 166–181. Springer, 2002.

[Gmati et al., 2007] Gmati I., Nurcan S. Un cadre de référence pour analyser les exigences d'alignement métier / système d'information, p 1-7, ECI Workshop, Paris, 2007.

[GMF] Eclipse Graphical Modelling Framework: http://www.eclipse.org/gmf/

[Godoy, 2005] Godoy CP. Knowledge-Based Reasoning over the Web, PhD Thesis, Universidad del pais vasco, Novembre, 2005.

[Gomez-Perez et al., 2004] Gomez-Perez A., Fernandez-Lopez M., Corcho O. Ontological Engineering. Berlin, Germany: Springer-Verlag, 2004.

[Grimm et al., 2007] Grimm S., Hitzler P., Abecker A. Knowledge Representation and Ontologies: Logic, Ontologies and Semantic Web Language, Semantic Web Services, Springer, 2007.

[Grandori et al., 1995] Grandori A., Soda G. Interfirms Networking, Antecedents, Mechanisms and Forms. In Organization Studies, Volume 16, p. 16-2, 1995.

[Gruber, 1993] Gruber TR. A translation approach to portable ontologies, Knowledge Acquisition, 5(2), 199-220, 1993

[Gruber, 1995] Gruber TR. Toward principles for the design of ontologies used for knowledge sharing, 1995.

[Guarino et al., 1994] Guarino N., Carrara M., Giaretta P. Formalizing Ontological Commitment. In Proc. of the National Conference on Artificial Intelligence (AAAI-94), Seattle, Morgan Kaufmann, 1994.

[Guarino, 1998] Guarino N. Formal Ontology and Information Systems, In the Proceedings of Formal Ontology in Information Systems, June 1998. Also in Frontiers in Artificial Intelligence and Applications, IOS-Press, Washington, DC, 1998.

[Gueguen et al., 2006] Gueguen G., Pellegrin-Boucher E., Torrès O. Between cooperation and competition: the benefits of collective strategies within business ecosystems. The example of the software industry**,** EIASM, 2nd Workshop on coopetition strategy, Milan, Italy, September 14-15, 2006.

[Haase et al., 2004] Haase P., Broekstra J., Eberhart A., Volz R. A Comparison of RDF Query Languages, Proceedings of the Third International Semantic Web Conference, 2004.

[Henderson et al., 1992] Henderson J., Venkatraman N. Strategic Alignment: A model for organizational transformation through information technology, in T. Kochan & M. Unseem, eds, Transforming Organizations, Oxford University Press,

NY, 1992.

[Horridge et al., 2004] Horridge M., Knublauch H., Rector A., Stevens R., Wroe C. A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0, University of Manchester, August, 2004.

[Horrocks et al., 2003] Horrocks I, Patel-Schneider PF. Reducing OWL Entailment to Description Logic Satisfiability, ISWC 2003 : international semantic web conference N°2, Sanibel Island FL , US, vol. 2870, p. 17-29, ISBN 3-540-20362-1, 2003.

[Hutt, 2005] Hutt K. A Comparison of RDF Query Languages, http://www.rh.edu/~rhb/cs_seminar_2005/SessionE1/hutt.pdf, Proceedings of The 21th Annual Rensselaer at Hartford Computer Science Conference, Connecticut USA, April 16, 2005.

[ICC, 1999] Intelligence Community Collaboration, Baseline Study Report, 1999.

[IDEAS, 2003] IDEAS, A gap Analysis –Required activities in Research, Technology and standardisation to close the RTS Gap- Roadmaps and Recommendations on RTS activites, IDEAS, Deliverables, 2003.

[IEEE, 1990] IEEE: Standard Computer Dictionary - A Compilation of IEEE Standard Computer Glossaries, 1990.

[ISO 16100, 2002] ISO 16100. Industrial automation systems and integration – Manufacturing software capability profiling for interoperability – Part 1: Framework, ISO/TC 184/SC 5/WG 4, 2002.

[ISO 9001, 2000] Norme européenne NF EN ISO 9001 version 2000, Systèmes de management de la qualité – Exigences, AFNOR, 2000.

[Jacobs, 2002] Jacobs J. Gartner's Collaboration Glossary. Gartner Report, 2002.

[Jouault, 2006] Jouault F. Contribution à l'étude des langages de transformation de modèles, Ph.D Thesis, Université de Nantes, 2006.

[Kak et al., 2002] Kak R., Schoonmaker M. RosettaNet E-Business Standards Provide Better Supply Chain Collaboration and Efficiency. ASCET volume a, 2002.

[Kaliban, 2007] Kaliban V. Ontology From AI to IS, C02: Ontology for Interoperability, InterOP, 2007.

[Katzy et al., 2000] Katzy B., Zhang C., Löh H. Reference Models for Virtual Organizations. Working Paper No 2704, Working Paper Series, CeTIM, 2000.

[Keller et al., 2005] Keller U., Feier C. State of the art and requirements on reasoning with semantic web services, RW^2 Project Deliverable, D1.1 v1.1, July, 2005.

[Klein et al., 2003] Klein M., Broekstra D., Fensel F., van Harmelen F., Horrocks I. Ontologies and Schema Languages on the Web. In D. Fensel, J. Hendler, H. Lieberman (eds.), Spinning the Semantic Web, MIT Press, Cambridge, MA, 2003.

[Klyne et al., 2003] Klyne G., Carroll J.J. Resource Description Framework (RDF): Concepts and abstract syntax. W3C Working Draft, available at http://www.w3.org/TR/2003/WD-rdf-concepts-20030123, 2003.

[Knublauch, 2006] Knublauch H. TopBraid Composer and Protege-OWL, http://www.topquadrant.com/topbraid/composer/tbc-protege.html, May, 2006.

[Konstantas et al., 2005] Konstantas D., Bourrieres J.P., Leonard M., Boudjlida N. Interoperability of enterprise software and applications, Springer-Verlag, p. v-vi, 2005

[Kuan, 2004] Kuan M. Using SWRL and OWL DL to Develop an Inference System for Course Scheduling. Masters Thesis, Chung Yuan Christian University, Taiwan, R.O.C., 2004.

[Lambert et al., 1999] Lambert DM., Emmelhainz MA., Gardner JT. Building successful logistics partnerships, Journal of Business Logistics, 20(1), 1999.

[Lassila et al., 1999] Lassila O., Swick R. Resource description framework (RDF) model and syntax specification, W3C Recommendation, http://www.w3.org/TR/REC-rdf-syntax/, 1999.

[Lassila et al., 2001] Lassila O., McGuinness D.The Role of Frame-Based Representation on the Semantic Web. Technical Report KSL-01-02. Knowledge Systems Laboratory. Stanford University. Stanford, California, 2001.

[Laubacher et al., 1997] Laubacher R., Malone T. Flexible Work Arrangements and 21st Century Worker's Guilds. MIT 21st Century Initiative Working Paper, October, 1997.

[Li et al., 2006] Li M., Cabral R., Doumeingts G., Popplewell K. Enterprise Interoperability: A concerted research roadmap for shaping business networking in the knowledge-based economy, Commission for the European Communities, Brussels, 2006.

[Liao, 2003] Liao S. Knowledge management technologies and applications - literature review from 1995 to 2002, Expert Systems with Applications, vol. 25, no. 2, p. 155-164, 2003.

[Luczak et al., 2005] Luczak H., Hauser A. Knowledge management in virtual organizations, Proceedings of ICSSSM'05. 2005 International Conference on Services Systems and Services Management, p. 898, 2005.

[Luke et al., 2000] Luke S., Heflin J. SHOE 1.01. Proposed Specification, SHOE Project technical report, University of Maryland, Available at http://www.cs.umd.edu/projects/plus/SHOE/spec1.01.htm, 2000.

[Maedche et al., 2003] Maedche A., Motik B., Stojanovic L., Studer R., Volz R. Ontologies for Enterprise Knowledge Management. IEEE Intelligent Systems 18(2):26–33, 2003.

[Maier, 1998] Maier MW. Architecting principles for systems-of-systems, Systems Engineering, Volume 1, Issue 4, p. 267-284, 1998.

[Malone et al., 1994] Malone TW., Crowston K. The interdisciplinary study of coordination. Computing Surveys, 26(1), p. 87–119, 1994.

[Malone et al., 1999] Malone, T.W., Crowston K., Lee J., Pentland B. Tools for inventing organizations: Toward a handbook of organizational processes, Management Science, 45(3), 425-443, 1999

[Malone et al., 2003] Malone TW, Crowston K, Herman GA. Organizing business knowledge – The MIT Process Handbook, ISBN 0-262-13429-2, Chapters 1 and 3, 2003.

[Mani Chandy, 2006] Mani Chandy K. Event-Driven Applications: Costs, Benefits and Design Approaches, California Institute of Technology, Gartner Application Integration and Web Services, 2006.

[Matheus, 2004] Matheus C. SWRLp: An XML-Based SWRL Presentation Syntax, In Proceedings of Rules and Rule Markup Languages for the Semantic Web: Third International Workshop, RuleML 2004, Hiroshima, Japan, p. 194-199, November 2004.

[McGuinness et al., 2000] McGuinness D., Fikes R., Rice J,. Wilder S. The Chimaera Ontology Environment. In: Rosenbloom P., Kautz HA., Porter B., Dechter R., Sutton R., Mittal V. (eds) 17th National Conference on Artificial Intelligence (AAAI'00), Austin, Texas, p. 1123–1124, 2000.

[Merilinna, 2005] Merilinna J. A Tool for QualityDriven Architecture Model Transformation, ISBN 951.38.6439.1 (soft back ed.) 951.38.6440.5 (URL:http://www.vtt.fi/inf/pdf/), 2005.

[MetaCase] Domain Specific Modeling with MetaEdit+: http://www.metacase.com

[Michelson, 2006] Michelson BM. Event-Driven Architecture Overview, Patricia Seybold Group, 2006.

[Millet et al., 2003] Millet J., Mukerji J. MDA Guide Version 1.0.1, available on http://www.omg.org, 2003.

[Milton, 2008] Milton N. Knowledge Technologies, POLIMETRICA, Publishing Studies series, Volume 3, 2008.

[Missikoff et al., 2006] Missikoff M., Taglino F. Ontologies for interoperability: a systematic overview. Lecture in ECI Workshop, Paris, 2006

[Morley, 2002] Morley C. La modélisation des processus : typologie et proposition utilisant UML. Processus et Systèmes d'information – Journées ADELI, Paris, France, 2002.

[Morley et al., 2005] Morley C., Hugues J., Leblanc B., Hugues O. Processus métiers et S.I : évaluation, modélisation, mise en oeuvre, éditions DUNOD, ISBN 2100070991, 2005.

[Neches et al., 1991] Neches R., Fikes RE., Finin T., Gruber TR., Senator T., Swartout WR.

Enabling technology for knowledge sharing. AI Magazine 12(3):36–56, 1991.

[Nonaka et al., 1995] Nonaka I., Takeuchi H. The knowledge creating company: how Japanese companies create the dynamics of innovation, Oxford University Press, London, 1995.

[Noy et al., 2001] N. F. Noy and D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology," Stanford Knowl. Sys. Lab., Stanford, CA, Tech. Rep. KSL-01-05, 2001.

[O'Connor et al., 2005] O'Connor M, Knublauch H, Tu S, Grosof B, Dean M, Grosso W, Musen M. Supporting rule system interoperability on the Semantic Web with SWRL, The Semantic Web – ISWC 2005, p. 974-986, 2005.

[Owen et al., 2003] Owen M., Raj J. BPMN and Business Process Management Introduction to the New Business Process Modeling Standard, Popkin Software, 2003.

[Parsia et al., 2005] Parsia B., Sirin E., Cuenca Grau B., Ruckhaus E., Hewlett D. Cautiously approaching SWRL. Preprint submitted to Elsevier Science, 2005.

[Pellet-Faq] http://pellet.owldl.com/faq/answering-queries/

[Petersen, 2005] Petersen SA. The role of enterprise modelling in virtual enterprises, Collaborative Networks and Their Breeding Environments, Springer, 2005

[PH-OWL] OWLized version of the Process Handbook: http://www.ifi.unizh.ch/ddis/ph-owl.html, University of Zurich

[Pingaud, 2003] Pingaud H. Logistiques et technologies de l'information et de la communication : les guides experts /. WEKA. 100 p., ISBN 2-7337-0232-7, 2003.

[Plante, 2006] Plante F. Introduction the GMF Runtime, http://www.eclipse.org/articles/Article-Introducing-GMF/article.html, IBM, 2006.

[Pollard, 2002] Pollard S. Collaboration – The Cure-All in New Economy Competitiveness?, AMR Research Report, 2002.

[Popplewell et al., 2008] Popplewell K., Stijanovic N., Abecker A., Apostolou D., Mentzas G., Harding JA. Supporting Adaptive Enterprise Collaboration through Semantic Knowledge Services, in Enterprise Interoperability III: New Challenges and Industrial Approaches, eds. K. Mertins, R. Ruggaber, K. Popplewell & X. Xu, Springer, Berlin, Germany, p. 381-393, 2008.

[Poulin et al., 1994] Poulin D., Montreuil B., Gauvin S. L'entreprise réseau, Montreal : Public Relais, 1994.

[Putnik et al., 2008] Putnik G.D., Cunha M.M. Encyclopedia of Networked and Virtual Organizations, Information Science Reference, 2008

[Ratchev et al., 2000] Ratchev S., Shiau J., Valtchanov G. Distributed product and facility prototyping in extended manufacturing enterprises, International journal of production reserarch, vol. 38, p. 4495-4506, 2000.

[RDF-OWL] Introduction to OWL: http://www.w3schools.com/RDF/rdf_owl.asp

[RuleML Initiative, 2005] RuleML Initiative. The Rule Markup Initiative Web Site. http://www.ruleml.org, 2005.

[Russell et al., 2003] Russell SJ., Norvig P. Artificial Intelligence: a modern approach. 2nd international edition edn. Prentice Hall Series in Artificial Intelligence, 2003.

[Schlenoff et al., 1999] Schlenoff C., Greninger M., Ciocoiu M., Lee J. The essence of the process specification language. Special Issue on Modelling and Simulation of Manufacturing Systems in the Transactions of the Society for Computer Simulation International, 16 (4), 204-216, 1999

[Schmidt, 2006] Schmidt DC. Model-Driven Engineering, IEEE Computer 39 (2). Retrieved on 2006-05-16, 2006.

[SPARQL-FAQ] http://thefigtrees.net/lee/sw/sparql-faq

[SPARQL] http://esw.w3.org/topic/SparqlImplementations

[STRATEGOR, 1997] STRATEGOR, Politique générale de l'entreprise, stratégie, structure, décision, identité. STRATEGOR est l'équipe des professeurs du département stratégie et politique de l'entreprise du groupe HEC de Jouy-en-Josas, 1997.

[SWRL-FAQ] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLRuleEngineBridgeFAQ

[Tawbi, 2001] Tawbi M. CREWS-L'Ecritoire : Une apporche Guidant l'Ingénierie des Besoins. XIXème Congrès INFORSID, Martigny, Suisse, 24-27 May, 123-141, 2001

[Touzi, 2007] Touzi J. Aide à la conception de Système d'Information Collaboratif support de l'interopérabilité des enterprises, Ph.D Thesis, INPT, 2007.

[Touzi et al., 2008] Touzi J., Bénaben F., Pingaud H., Lorré JP. A Model-Driven approach for Collaborative Service-Oriented Architecture design, Accepted paper to be published in a special issue of the International Journal of Production Economics (IJPE), 2008.

[Truptil, 2008] Truptil S., Bénaben F., Couget P., Lauras M., Chapurlat V., Pingaud H. Interoperability of information systems in crisis management: Crisis modelling and metamodelling, in Proceeding of I-ESA 2008: Enterprise Interoperability III, ISBN 978-1-84800-220-3, p. 583-594, 2008.

[Uschold et al., 1998] Uschold M., King M., Moralee S., Zorgios Y. The Enterprise Ontology, In M. Uschold & A. Tate, (Eds), Knowledge Engineering Review Special Issue on Putting Ontologies to Use, 13(1), 31-89, 1998

[van Heijst et al., 1996] van Heijst G., Schreiber AT., Wielinga BJ. Using explicit ontologies in KBS development," Int. J. Hum. Comput., p. 183–292, 1996.

[Vasko et al., 2006] Vasko M., Dustdar S. A view based analysis of workflow modeling languages, Proceedings of the 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP'06), IEEE, 2006.

[Vernadat, 1999] Vernadat F. Techniques de modélisation en entreprise : Applications aux processus Opérationnels, économica, 1999.

[Vernadat, 2006] Vernadat F.B. Interoperable enterprise systems: architectures and methods, INCOM'06 Conference, St-Etienne, France, 2006

[Wiederhold et al., 1997] Wiederhold G., Genesereth M. The conceptual basis for mediation services, IEEE Expert: Intelligent Systems and Their Applications**,** Volume 12, Issue 5 (September 1997), p. 38 – 47, 1997.

[White et al., 2007] White J., Schmidt D.C., Mulligan S. The Generic Eclipse Modeling System, Eclipse magazine, vol. 6, Jan 2007.

[Wiki Inference engine] http://en.wikipedia.org/wiki/Inference_engine

[Zaidat, 2005] Zaidat A. Spécification d'un cadre d'ingénierie pour les réseaux d'organizations, Ph.D. Thesis, 1-258, 2005.

Annex A: Five groups of deduction rules

In this annex, we present all of the deduction rules we defined in our work. The deduction rules are classified under five groups. They are all written in SWRL. In Section 3.2 of Chapter 3, we presented only one rule for each group with an example of the deduction that can be inferred.

GR.1: Role ← → Abstract service

GR.1 Rule 1: Deduction of abstract service when role is defined

Participant(?x) ∧ playRole(?x, ?y) ∧ performAService(?y, ?z) →
provideAService(?x, ?z)

GR.1 Rule 2: Deduction of role when abstract service is defined

Participant(?x) ∧ provideAService(?x, ?y) ∧ isPerformedByRole(?y, ?z) →
playRole(?x, ?z)

GR.2: Abstract service → Business service

GR.2 Rule 1: Deduction of business service when abstract service is known

Participant(?x) ∧ provideAService(?x, ?y) ∧ hasBusinessService(?y, ?a) →
provideBusinessService(?x, ?a)

GR.3: Resource → Dependency → Coordination service → MIS service

GR.3 Rule 1: Deduction of dependency between business services, deduction of coordination service that can manage the derived dependencies, and creation of MIS service when coordination service is known. The dependency is derived on the direction that we compare an output of P1 with an input of P2.

CNetwork(?a) ∧ hasRelationship(?a, ?z) ∧
P1(?z, ?y) ∧ provideBusinessService(?y, ?c) ∧ hasOutput(?c, ?d) ∧
P2(?z, ?x) ∧ provideBusinessService(?x, ?b) ∧ hasInput(?b, ?d) ∧
CoordinationService(?f) ∧ manipulateResource(?f, ?d) ∧
Dependency_between_BusinessServices(?e) →
fromBusinessService(?e, ?c) ∧ toBusinessService(?e, ?b) ∧ containResource(?e, ?d) ∧
isCoordinatedBy(?e, ?f) ∧ hasMISservice(?a, ?f) ∧ MISservice(?f)

GR.3 Rule 2: Same concept as in GR.3 Rule 1, but the dependency is derived on the direction that we compare an input of P1 with an output of P2.

CNetwork(?a) ∧ hasRelationship(?a, ?z) ∧
P1(?z, ?y) ∧ provideBusinessService(?y, ?c) ∧ hasInput(?c, ?d) ∧
P2(?z, ?x) ∧ provideBusinessService(?x, ?b) ∧ hasOutput(?b, ?d) ∧
CoordinationService(?f) ∧ manipulateResource(?f, ?d) ∧

Dependency_between_BusinessServices(?e) →
toBusinessService(?e, ?c) ∧ fromBusinessService(?e, ?b) ∧ containResource(?e, ?d) ∧ isCoordinatedBy(?e, ?f) ∧ hasMISservice(?a, ?f) ∧ MISservice(?f)

GR.3 Rule 3: Deduction of dependency between MIS services. This rule is based on the same approach as the above rules. We change the concepts of Business service to MIS service. The deduction of such dependency has no direction to be taken into account because we consider every MIS services of the CNetwork at the same time.

CNetwork(?a) ∧ hasMISservice(?a, ?b) ∧ hasOutput(?b, ?d) ∧
hasMISservice(?a, ?c) ∧ hasInput(?c, ?d) ∧
Dependency_between_MISservices(?e) →
fromMISservice(?e, ?b) ∧ toMISservice(?e, ?c)

GR.4: Common goal → Abstract service

GR.4 Rule 1: Deduction of abstract service when a common goal is described.

CommonGoal(?x) ∧ description(?x, ?a) ∧ swrlb:substringBefore(?y, ?a, " ") ∧ AbstractService(?b) ∧ name(?b, ?c) ∧ swrlb:containsIgnoreCase(?c, ?y) → achievesAService(?x, ?b)

GR.5: Power and Duration → Topology

GR.5 Rule 1: Deduction of the star typed topology when the decision-making power is central and the duration is continuous.

Topology(?x) ∧ hasPower(?x, central) ∧ hasDuration(?x, continuous) → hasType(?x, star)

GR.5 Rule 2: Deduction of the P2P typed topology when the decision-making power is equal and the duration is discontinuous.

Topology(?x) ∧ hasPower(?x, equal) ∧ hasDuration(?x, discontinuous) → hasType(?x, P2P)

GR.5 Rule 3: Deduction of the chain typed topology when the decision-making power is hierarchic and the duration is continuous.

Topology(?x) ∧ hasPower(?x, hierarchic) ∧ hasDuration(?x, continuous) → hasType(?x, chain)

Annex B: SPARQL

The following figure shows the use of SPARQL in the prototype:

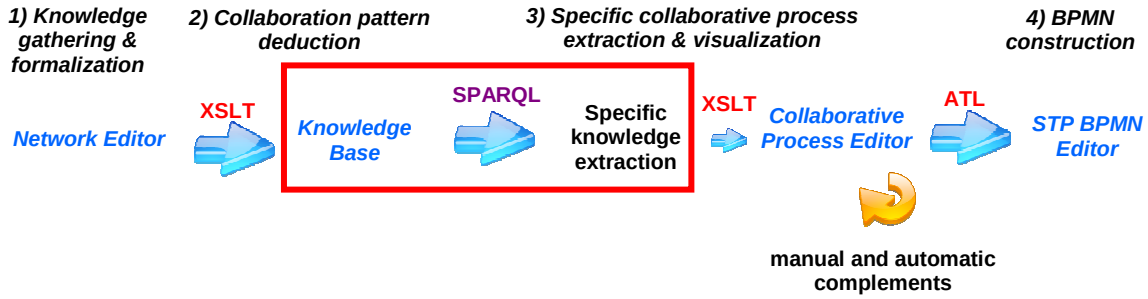


Fig.B. 1 Use of SPARQL in the prorotype

The implementation of a SPARQL query requires Jena and Pellet engines. Jena is in charge of converting the OWL ontology model into the model adapted to the Pellet engine according to the libraries of Jena and Pellet. Then, Pellet captures and processes SPARQL queries regarding to the model. Finally, query results will be returned in XML. Fig.B.2 illustrates the implementation of SPARQL:

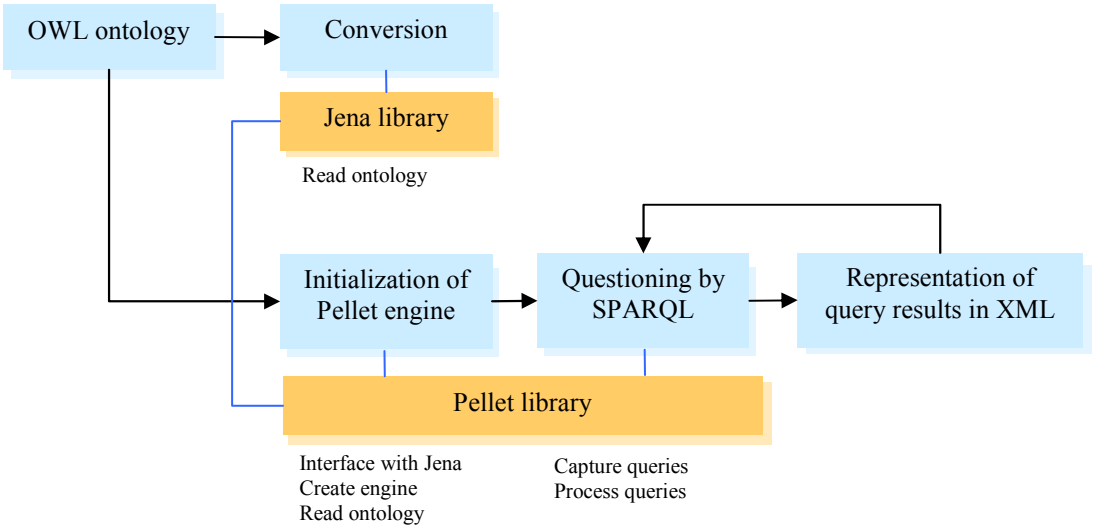


Fig.B. 2 SPARQL query implementation using Jena and Pellet

To demonstrate querying with SPARQL, we first need an ontology to query. Fig.B.3 presents an example of a collaborative network model (OWL-based) stored in the KB as instance. This model describes a simple collaborative network of two partners (*A* and *B*). A relationship between the role *seller* of *A* and *buyer* of *B* is established with continuous communication and hierarchic decision-making power.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:CNetwork="http:///CNetwork.ecore"
        xmlns:xmi="http://www.omg.org/XMI"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:base="http://nettyrajsiri.googlepages.com/OWL_cnetwork.xml"
        xmlns:= "http://nettyrajsiri.googlepages.com/CNKB_v5.owl#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
        <CNetwork rdf:ID="N10001">
                <hasParticipant>
                        <Participant rdf:ID="A">
                                <name xml:lang="en">A</name>
                                <is_in_CNetwork rdf:resource="#N10001" />
                                <playRole rdf:resource="#seller" />
                        </Participant>
                </hasParticipant>
                <hasParticipant>
                        <Participant rdf:ID="B">
                                <name xml:lang="en">B</name>
                                <is_in_CNetwork rdf:resource="#N10001" />
                                <playRole rdf:resource="#buyer" />
                        </Participant>
                </hasParticipant>
                <hasCommonGoal>
                        <CommonGoal rdf:ID="buy products">
                                <description xml:lang="en">buy products</description>
                                <isAchievedBy rdf:resource="#N10001" />
                        </CommonGoal>
                </hasCommonGoal>
                <hasRelationship>
                        <Relationship rdf:ID="A-B">
                                <P1 rdf:resource="#A" />
                                <P2 rdf:resource="#B" />
                                <P1.role rdf:resource="#seller" />
                                <P2.role rdf:resource="#buyer" />
                        </Relationship>
                </hasRelationship>
                <hasTopology>
                        <Topology rdf:ID="N10007">
                                <hasDuration rdf:resource="#continuous" />
                                <hasPower rdf:resource="#hierarchic" />
                        </Topology>
                </hasTopology>
        </CNetwork>
</rdf:RDF>
```

Fig.B. 3 Collaborative network model (OWL-based)

We start by reading the Jena ontology model (*com.hp.hpl.jena.ontology.OntModel*). Jena is in charge of converting the above ontology model into the model adapted to the Pellet engine (*org.mindswap.pellet.jena.PelletReasonerFactory*). To make queries concise, we need to define prefixes and base URIs. This demonstration is focused only on querying name, and role of participants. The query is thus written as follows:

```
//create an empty model
OntoModel model = ModelFactory.createOntologyModel(PelletReasonerFactory.THE_SPEC);

//read model
model.read("CNO.owl");

String queryBegin =    " PREFIX rdf:  http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                       + " PREFIX   : http://nettyrajsiri.googlepages.com/CNKB_v5.owl#"
String queryEnd = "}";

//query
String queryStr = queryBegin
            + "SELECT  ?name ?role "
            + "WHERE { "
            + "?N rdf:type :CNetwork."
            + "?N :hasParticipant ?P."
            + "?P :name ?name."
            + "?P :playRole ?role."
            + "}"
+ queryEnd;

//create the query
Query query = QueryFactory.create(queryString);
```

Fig.B. 4 SPARQL query

Variables are indicated by a *?*. The variables *?N, ?P, ?name, ?role* refer respectively to the individuals of the *CNetwork* class, and *hasParticipant*, *name*, and *role* properties. The *hasParticipant* is the property linking the individuals of the *Cnetwork* class to the individuals of the *Participant* class. The *name* and *playRole* are the properties related to the variable *?P* and refer to the name and role of the individuals of the *Participant* class. The query above will return *?name*, and *?role* variables.

Processing a SPARQL query can be done with *PelletQueryExecution* class. This class allows us to use the Jena's query objects (created with *QueryFactory*) with the Pellet's query engine (*org.mindswap.pellet.jena.PelletQueryExecution*).

The results of SPARQL query can be extracted in result sets or XML format. We are interested only in XML format because it is exploitable, since the result we obtain from SPARQL queries will be visualized with the CPE. To get the query result, there are several possible methods, but in this example we use a *ResultSetFormatter*. This method offers the possibility to obtain the result in various formats, for example XML. The query result will be stored in a file named *result.xml*. Fig.B.5 shows how to execute a SPARQL query:

```
//execute the query and write the result on "result.xml" file

QueryExecution qe = new PelletQueryExecution(query, model);

ResultSet results = qe.execSelect();

OutputStream oStream = new FileOutputStream("result.xml");

ResultSetFormatter.outputAsXML(oStream, results);
```

Fig.B. 5 Query execution

By way of this example, Fig.B.6 is an extract of the results from the above example:

```
<?xml version="1.0"?>
<sparql
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xs="http://www.w3.org/2001/XMLSchema#"
   xmlns="http://www.w3.org/2005/sparql-results#" >
 <head>
   <variable name="name"/>
   <variable name="role"/>
 </head>
 <results ordered="false" distinct="false">
   <result>
     <binding name="name">
       <literal xml:lang="en">A</literal>
     </binding>
     <binding name="role">
       <uri>http://nettyrajsiri.googlepages.com/CNKB_v5.owl#seller</uri>
     </binding>
   </result>
   <result>
     <binding name="name">
       <literal xml:lang="en">B</literal>
     </binding>
     <binding name="role">
       <uri>http://nettyrajsiri.googlepages.com/CNKB_v5.owl#buyer</uri>
     </binding>
   </result>
 </results>
</sparql>
```

Fig.B. 6 Result of the query in XML format

The XML query result is regular format and fairly simple to read:

- All of the key elements belong to a single namespace, *http://www.w3.org/2005/sparql-results#*
- The root element is *sparql*, which contains a *head* and a *results* element.
- The *head* section declares all *variables* that will be returned, which are name and role in this case.
- The *results* section lists each query *result.*
- A *result* element contains one binding for each variable. A binding is one of *literal* or *uri*. These elements contain the actual values returned.

Therefore from this XML, we extract the name and role of participants in the network. We obtain the participant named *A* plays the role of *seller*, and the participant named *B* plays the role of *buyer*. This result corresponds to the collaborative network model shown in Fig.B.3.

Annex C: XSLT

C.1 Transformation with XSLT

The XSLT (Extensible Stylesheet Language Transformation) is an official recommendation of the World Wide Web Consortium (W3C). The XSLT 1.0 W3C recommendation was published in 1999 together with XPath 1.0, and it has been widely implemented since then. XSLT 2.0 has become a W3C recommendation since January 2007.

The XSLT provides a flexible, powerful language for transforming XML documents into something else (HTML, PDF, SVG, Java code, etc). The original document is not changed but a new document is created based on the content of an existing one. Fig.C.1 shows how the XSLT processes a document and the elements involved which are: an XML source document, XSLT stylesheet, XSLT processor, and result document.

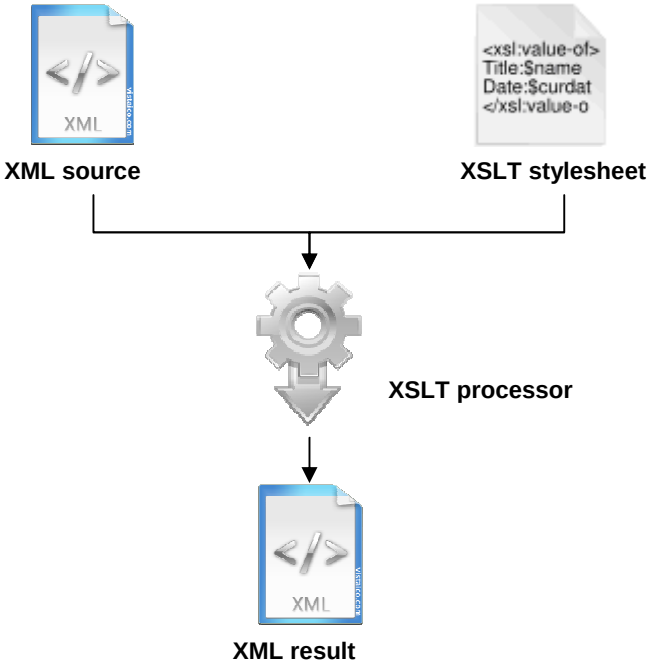


Fig.C. 1 Diagram of the basic elements and process flow of XSLT

A transformation expressed in an XSLT stylesheet describes the rules for transforming a source document into a result document. A stylesheet contains a set of template rules. A template rule has two parts: a pattern which is matched against nodes in the source document, and a template which can be instantiated to form part of the result document. This allows a stylesheet to be applicable to a wide class of documents that have similar source document structures. XSLT makes use of the expression language defined by XPath to select elements for processing, for conditional processing and for generating text.

The processor starts by processing the root node of the source document, finding in the stylesheet the best matching template for that node and evaluating the template's contents. Instructions in each template generally direct the processor to either create nodes in the result document, or process more nodes in the source document in the same way as the root node.

The structure of the result document can be completely different from the structure of the source one. In constructing the result document, elements from the source document can be filtered and reordered and an arbitrary structure can be added.

C.2 Implementation of XSLT

The uses of XSLT in the prototype take place twice as shown below:

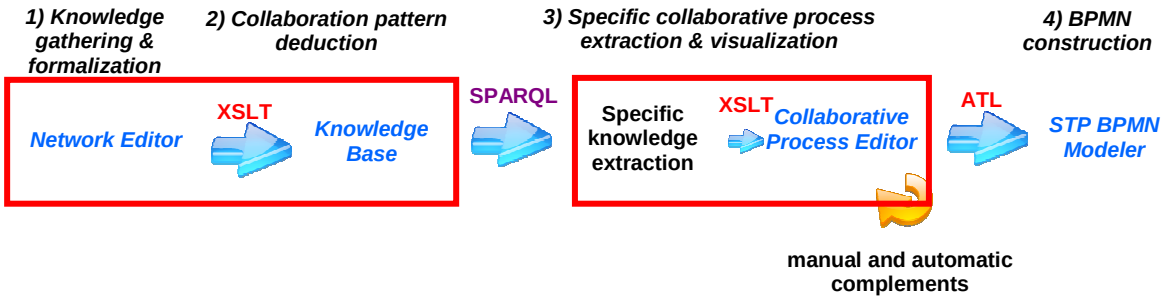


Fig.C. 2 Uses of XSLT in the prototype

Here, we present a complete example of the transformation from Network Editor to Knowledge Base. XSLT can also be used based on the transformation concept defined in Section 2.2.2.2/b of Chapter 4.

To demonstrate the XSLT transformation, we create a collaboration network of two participants named *A* and *B*. The participant *A* plays the role of *seller* which has a relationship with the role *buyer* of the participant *B*. Fig.C.3 presents the diagram and XML documents of this example:


```
<?xml version="1.0" encoding="UTF-8"?>
<CNetwork:network xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:CNetwork=http:///CNetwork.ecore id="
N10001">
  <topology duration="continuous" power="hierarchic" id=" N10007"/>
  <commonGoals description="buy products"/>
  <participants name="A">
    <role name="seller">
      <relationship target="//@participants.1/@role.0" source="//@participants.0/@role.0" label="supplier_customer"
duration="continuous" />
    </role>
  </participants>
  <participants name="B">
    <role name="buyer"/>
  </participants>
</CNetwork:network>
```

Fig.C. 3 Collaborative network diagram and its XML model

For the purposes of this example, the XML source document will be stored in a file called *input.xml*. This document use namespaces which are for instance: *CNetwork*, defined like this

http:///CNetwork.ecore. The root element is *CNetwork:network* which contains *commonGoals*, *participant*, and *topology* elements. A *participant* element contains a *role* element which itself contains a *relationship* element if its source attribute refers to its parent role.

In this transformation, we intend to obtain the result model conforming to the OWL-based CO (upper part of Fig.III.4). The source model (Fig.C.3) conforms to the meta-model of the NE (Fig.IV.6). The mapping between source and result elements is defined as follows:

XML source elements →	Result elements
CNetwork:network	CNetwork
topology	Topology
commonGoals	CommonGoal
participants	Participant
role	playRole
relationship	Relationship

Table C. 1 XML source elements and their corresponding result elements

The XSLT stylesheet has the root element stylesheet which is in the namespace *http://www.w3.org/1999/XSL/Transform*. This namespace is mapped to the *xsl* prefix so that we have to write xsl:stylesheet rather than simply stylesheet. Like the root element, all other XSLT elements also have their proper namespaces for example: *http://www.w3.org/1999/02/22-rdf-syntax-ns#* for *rdf* element, and *http:///CNetwork.ecore* for *CNetwork* element. Fig.C.4 shows the namespaces of the XSLT stylesheet:

```
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:CNetwork="http:///CNetwork.ecore">
```

Fig.C. 4 Namespaces of the XSLT stylesheet

Here, we will present only the transformation of a *CNetwork:network* element of the source document into a *CNetwork* element of the result document. We can write the template rule like this:

```
<xsl:template match="CNetwork:network">
        <rdf:RDF>
        <CNetwork rdf:ID="{@id}">
                <xsl:apply-templates select="participants" />
                <xsl:apply-templates select="commonGoals" />
                <xsl:apply-templates select="topology" />
        </CNetwork>
        </rdf:RDF>
</xsl:template>

<xsl:template match="participants">
        <hasParticipant>
        <Participant rdf:ID="{@name}">
                <name xml:lang="en"><xsl:value-of select="@name" /></name>
                <is_in_CNetwork rdf:resource="#{../@id}" />
                <xsl:apply-templates select="abstractService" />
        </Participant>
        </hasParticipant>
</xsl:template>
```

Fig.C. 5 A part of the XSLT stylesheet for transforming the elements of CNetwork:network into CNetwork

The pattern *CNetwork:network* is matched against the *CNetwork:network* element which is the root element of the source document. Then the elements found inside the template rule will be created along with their attributes, for example, the *rdf:ID* attribute will take the value of the *id* attribute of the current element *(CNetwork:network).* Then we apply templates to the children of the current element with *xsl:apply-templates*. In this example, we apply templates to select, for instance, the *participants*, *commonGoals*, and *topology* elements. Once the processor finds the template matched to the apply-templates, it will follow through the rule defined in that template. By way of this example, the processor finds the template *participants* that matches the apply-template *participants*. So the pattern participants will be matched against the *participants* element of the source document and all the elements and attributes will be created.

Processing the XSL transformations is done with Java API by using the *TransformerFactory* and *Transformer*. We have to specify the names of the XSLT stylesheet (xcode.xml), XML source file (input.xml), and the result file (output.owl). Fig.C.6 shows the processing XSL transformation by Java API:

```
TransformerFactory tFactory = TransformerFactory.newInstance();

Transformer transformer = tFactory.newTransformer(new StreamSource("xcode.xml"));

transformer.transform(new StreamSource("input.xml"), new StreamResult("output.owl"));
```

Fig.C. 6 Processing XSL transformation with Java

Once the XSLT has been processed, we will obtain the result, which is the model of the collaborative network as shown in the following figure. This result will be imported into the KB as a new instance.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:CNetwork="http:///CNetwork.ecore"
        xmlns:xmi="http://www.omg.org/XMI"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:base="http://nettyrajsiri.googlepages.com/OWL_cnetwork.xml"
        xmlns:= "http://nettyrajsiri.googlepages.com/CNKB_v5.owl#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
        <CNetwork rdf:ID="N10001">
                <hasParticipant>
                        <Participant rdf:ID="A">
                                <name xml:lang="en">A</name>
                                <is_in_CNetwork rdf:resource="#N10001" />
                                <playRole rdf:resource="#seller" />
                        </Participant>
                </hasParticipant>
                <hasParticipant>
                        <Participant rdf:ID="B">
                                <name xml:lang="en">B</name>
                                <is_in_CNetwork rdf:resource="#N10001" />
                                <playRole rdf:resource="#buyer" />
                        </Participant>
                </hasParticipant>
                <hasCommonGoal>
                        <CommonGoal rdf:ID="buy products">
                                <description xml:lang="en">buy products</description>
                                <isAchievedBy rdf:resource="#N10001" />
                        </CommonGoal>
                </hasCommonGoal>
                <hasRelationship>
                        <Relationship rdf:ID="A-B">
                                <P1 rdf:resource="#A" />
                                <P2 rdf:resource="#B" />
                                <P1.role rdf:resource="#seller" />
                                <P2.role rdf:resource="#buyer" />
                        </Relationship>
                </hasRelationship>
                <hasTopology>
                        <Topology rdf:ID="N10007">
                                <hasDuration rdf:resource="#continuous" />
                                <hasPower rdf:resource="#hierarchic" />
                        </Topology>
                </hasTopology>
        </CNetwork>
</rdf:RDF>
```

Fig.C. 7 OWL-based model to be imported into the KB

Annex D: ATL

D.1 Transformation with ATL

In this section, we will talk about the overall concept of how to do the transformation with ATL. The full discussion about ATL is in [Jouault, 2006]. Another application of this tool is in [Touzi, 2007] which addresses a particular transformation of a BPMN model into a SOA-based UML model. This application is really relevant to our work because the BPMN output model of our work will be used as an input of this application.

The Atlas Transformation Language (ATL) is the ATLAS INRIA & LINA research group's answer to the OMG MOF/QVT RFP (Query, View and Transformation). ATL is a model transformation language specified as both a meta-model and a textual concrete syntax. In the Model-Driven Engineering (MDE) field, ATL provides developers with a means of specifying the way to produce a number of target models from a set of source models. In other words, ATL introduces a set of concepts that allows model transformations to be described. An ATL transformation program is composed of rules that define how the source model elements are matched to create and initialize the elements of the target models. Fig.D.1 summarizes the full model transformation process:

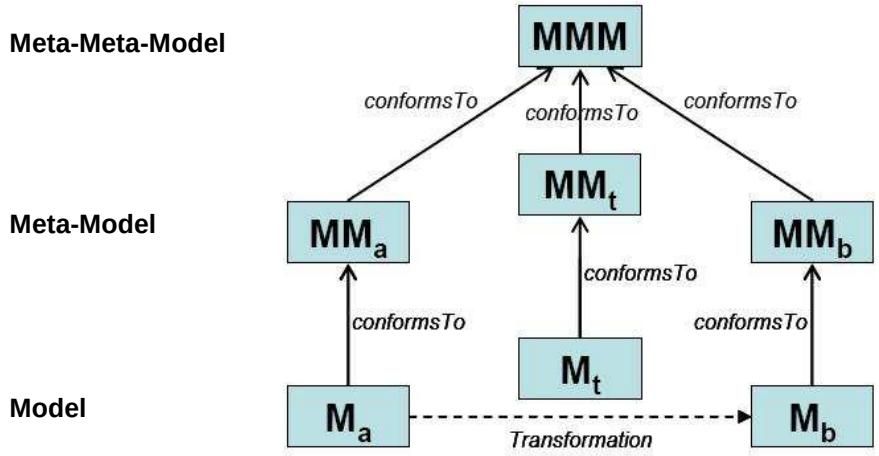


Fig.D. 1 Overview of model transformation [ATL manual, 2006]

A model M_a conforming to a meta-model MM_a, is here transformed into a model M_b that conforms to a meta-model MM_b. The transformation is defined by the model transformation M_t which itself conforms to a meta-model MM_t. Every meta-model has to conform to a meta-meta-model MMM (MOF or Ecore). Hence in our application, the MMM is Ecore and the MM_t is certainly ATL.

D.2 Implementation of ATL

As discussed above, a transformation with ATL requires a source meta-model, a source model, a target meta-model, and an ATL model. According to the development process as shown below, the transformation with ATL takes place at the last step:

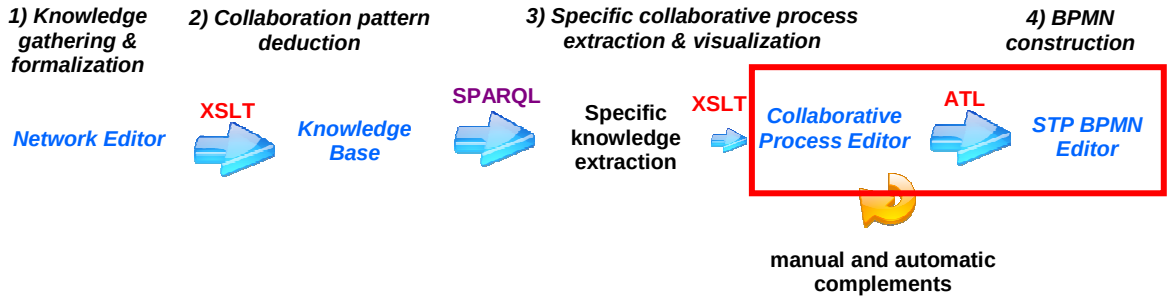


Fig.D. 2 Transformations with XSLT in our development process

We apply ATL to transform a collaborative process model (*Cprocess-CPE.xml*) into a BPMN relevant model (*Cprocess-BPMN.ecore*). The source model is generated from the CPE. The target model should conform to the BPMN meta-model of the STP BPMN Modeller. Transformation rules are quite simple due to the semantic proximity of the source and target meta-models. The implementation of this transformation will be performed as shown in Fig.D.3:

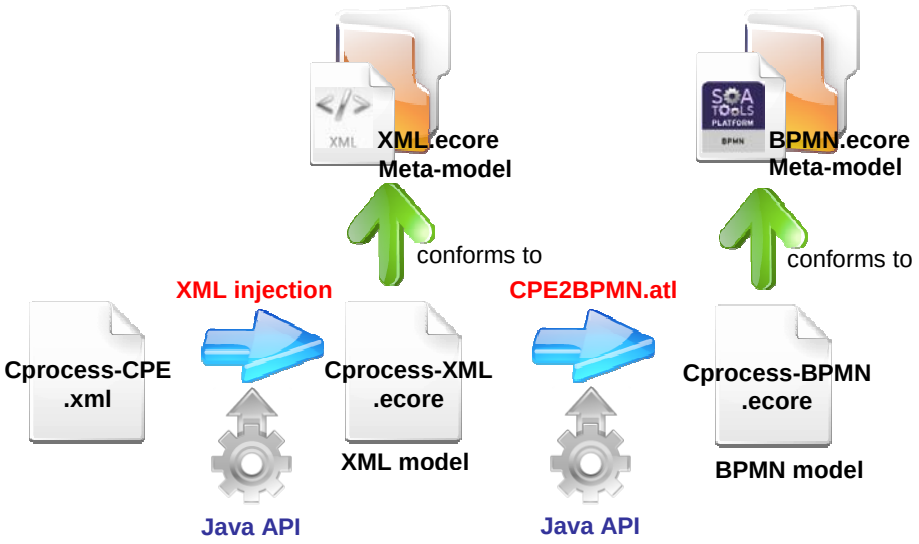


Fig.D. 3 Implementation of the transformations with ATL in our development

From the precedent schema, there are two steps of transformation:

- Injecting an XML file into an XML model (Ecore based). Or, from *Cprocess-CPE.xml* to *Cprocess-XML.ecore.*
- Transforming an XML model (Ecore based) into a BPMN model. Or, from *Cprocess-XML.ecore* to *CProcess-BPMN.ecore.*

To demonstrate the transformation with ATL, we need a collaborative process model created with the CPE. Fig.D.4 presents the diagram of the collaborative process model used in this example. Our collaborative process has two partners (*A* and *B*) which each provide their own

services corresponding to their role. They communicate with each other via the MIS which also provides its own services.

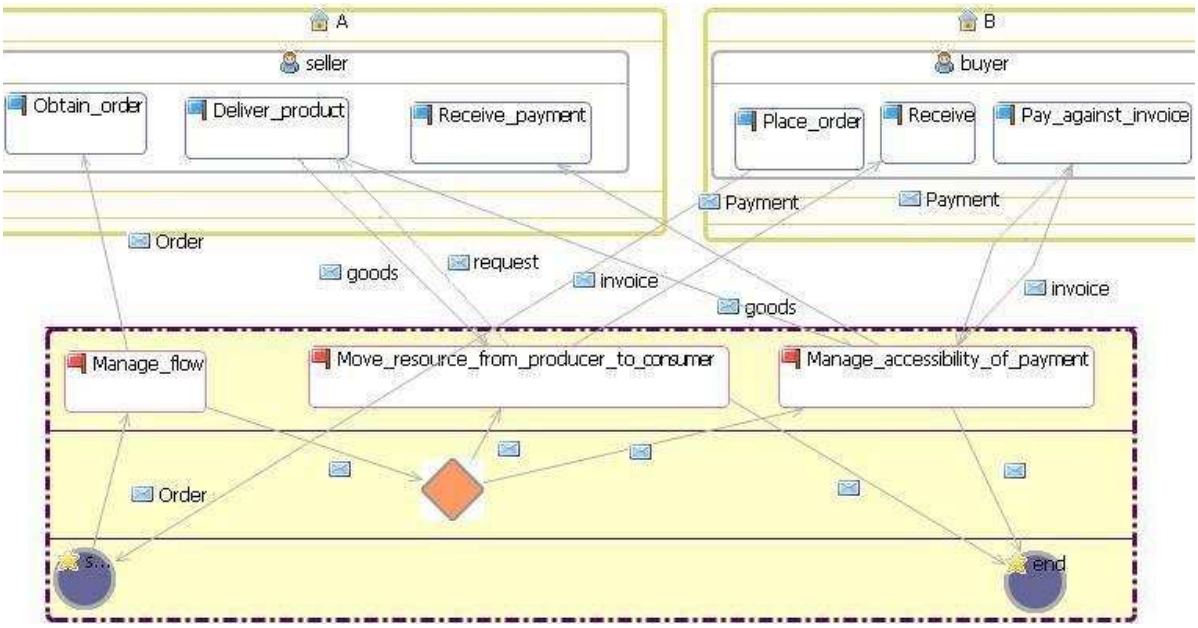


Fig.D. 4 Collaborative process diagram designed with CPE

As discussed in the CPE functionality section, besides the diagram, CPE also generates an XML schema file associated with this diagram. This XML schema conforms to the CPE's meta-model and is stored in a file named *Cprocess-CPE.xml*.

D.2.1 Injection of an XML file into an XML model

The very first step addresses an injection of an XML file into an XML model based on Ecore. There are two ways to do such an injection:

- Using the module *Inject XML file to XML model*. This module is provided under the AM3 (ATLAS MegaModel Management) perspective. For more detail, see page 138 in [Touzi, 2007].

- Automatically injecting an XML model with Java application. The *ATL Howtos* page [ATL-HowTo] in the Eclipse Wiki explains how to launch the transformations programmatically with Java API using the *AtlLauncher* class. This class will run the ATL model to transform one model into another, including the automatic injection of XML model.

In our development, we use automatic injection since it is based on Java and we can manipulate it in the Eclipse platform. The generated collaborative process model conforms to the XML meta-model. It will be stored in a file called *Cprocess-XML.ecore*.

D.2.2 Transformation of an XML model into a BPMN model

Once we have the XML model corresponding to the CPE-based collaborative process model, we can perform the real transformation with ATL. This is the most important step dealing with the transformation of an XML model *(CProcess-XML.ecore)* into a BPMN model *(CProcess-BPMN.ecore)*. The source and target meta-models are XML and BPMN respectively which both conform to Ecore. We start by conceptualising the match between the elements of these two meta-models:

N°	XML meta-model →	BPMN meta-model
1	Root	BpmnDiagram
2	Element with the name's value *participants* or *CIS*	Pool
3	Element with the name's value *role*	Lane
4	Element with the name's value *performsBusinessService*, *CISservices*, *gateways* or *events*	Activity
5	Element with the name's value *flows* and type *seqFlow*	Edge
6	Element with the name's value *flows* and type *msgFlow*	Message

Table D. 1 Matching between the elements of XML and BPMN meta-models

According to Table D.1, in this example we only present these three transformations:

N°1: Root to BPMN diagram → **rule** GenerateBPMNdiagram
N°2: Element to Pool → **rule** GenerateParticipantPools *(participants only)*
N°4: Element to Activity → **rule** GenerateBSActivities *(performsBusinessService only)*

The ATL IDE provides the ATL wizard dedicated to create ATL files *(*.atl)*. We create an ATL file to perform the three transformations listed previously, as shown in Fig.D.5:

```
-- meta-model declaration

module GMMF2BPMNeditor;
create OUT : bpmn from IN : XML;
-------------------------------------------------------------------------------------------------
-- methods

helper context XML!Element def: getName() : String =
        self.children->select(elmt | elmt.oclIsTypeOf(XML!Attribute))  ->first().value
;
helper context XML!Element def: getId() : String =
        self.children->select(elmt | elmt.oclIsTypeOf(XML!Attribute))->select(elmt|elmt.name='id')  ->first().value
;
helper context XML!Element def: getType() : String =
         if (self.name = 'flows') or (self.name='gateways')
   then
        let myVariable :String=  self.children->select(a | a.oclIsTypeOf(XML!Attribute)) ->select(a | a.name = 'type')
in myVariable ->first().value
else 'false'
        endif
;
helper context XML!Element def: isMyIncomingMsgs (f:String) : Boolean =
        if self.getToserviceId()= f and self.getType()='msgFlow'
        then true
        else false
        endif;
-------------------------------------------------------------------------------------------------
-- transformation rules

rule GenerateBPMNdiagram {
        from t:XML!Root
        to x: bpmn!BpmnDiagram
        ( pools <- t.children -> select(e|e.name='participants' or e.name='CIS')
                ,messages <- XML!Element.allInstances() -> select(e|e.name='flows') -> select(e|e.getType()='msgFlow')
                )
}
rule GenerateParticipantPools {
        from t:XML!Element (t.name ='participants' )
        to x: bpmn!Pool
        ( name <- t.getName(), iD <- t.getId()
                ,lanes <- t.children -> select(e|e.name='role')
                ,bpmnDiagram <- t.parent
                ,vertices <- t.children -> select(e|e.name='role')->collect(x|x.children)->flatten()
                                                                        -> select(x|x.name='performsBusinessService')
        )
}
rule GenerateBSActivities {
        from t:XML!Element (t.name ='performsBusinessService')
        to x: bpmn!Activity
        ( name <- t.getName(), iD <- t.getId()
                ,incomingMessages <- XML!Element.allInstances()-> select(e|e.name='flows')
                                                                        -> select(e|e.isMyIncomingMsgs(t.getId())=true)
                ,incomingEdges <- XML!Element.allInstances()-> select(e|e.name='flows')
                                                                        -> select(e|e.isMyIncomingEdges(t.getId())=true)
                ,lane<-t.parent
        )
}
```

Fig.D. 5 A part of the ATL file for transforming an XML model into a BPMN model

The ATL file has three parts:

- **Meta-model declaration** for defining the source and target meta-models. In this case, the source meta-model is defined as *OUT: bpmn*, and the target meta-model is defined as *IN: XML*.

- **Method** is not a mandatory requirement but it is helpful when we have to use the same retrieving pattern several times. It is created under the keyword *helper* which can be called via the transformation rule. This is an example of *helper* for getting a string name:

```
helper context XML!Element def: getName() : String =
    self.children->select(elmt | elmt.oclIsTypeOf(XML!Attribute)) ->first().value
```

This helper starts at finding every *element* of the XML source model. For each element, the helper goes to the *children* of this current element and gets the *first value* of *attribute* found.

- **Transformation rule** is the core of an ATL model. It is where we describe the match between the elements. It is created under the keyword *rule.* This is an example of *rule* for transforming business service of XML into activity of BPMN:

```
rule GenerateBSActivities {
    from t:XML!Element (t.name ='performsBusinessService')
    to x: bpmn!Activity
    (name <- t.getName(), iD <- t.getId()
    ,incomingMessages <- XML!Element.allInstances()-> select(e|e.name='flows')
                                    -> select(e|e.isMyIncomingMsgs(t.getId())=true)
    ,incomingEdges <- XML!Element.allInstances()-> select(e|e.name='flows')
                                -> select(e|e.isMyIncomingEdges(t.getId())=true)
    ,lane<-t.parent
    )
}
```

The rule starts by finding only the *elements* named *performsBusinessService*. Then, it calls the *helper getName()* in order to find the real name of this element and define it as the *name* of the activity. Then, it calls another *helper getId()* in order to define the *id* of the activity. The transformations of both *incomingMessages* and *incomingEdges* are more complex. The main concept is to get every child element named *flows* and verify if all the flows found belong to this business service or not. Finally, the rule transforms the *parent* element (role) of this business service element into *lane*.

The ATL file is stored in a file called *CPE2BPMN.atl*. Furthermore, we have an additional file automatically created by the system: the ASM file stored in a file called *CPE2BPMN.asm*. The ASM file contains the ATL bytecode that corresponds to the generated transformation file. This bytecode is encoded into an XML language and is updated when the transformation file is saved. In the scope of the ATL IDE, the compilation policy is based on the default Eclipse compilation policy: compilation is automatically performed in the background when an edited ATL file is saved [ATL manual, 2006]. Fig.D.6 shows the navigator view of our ATL transformation project:

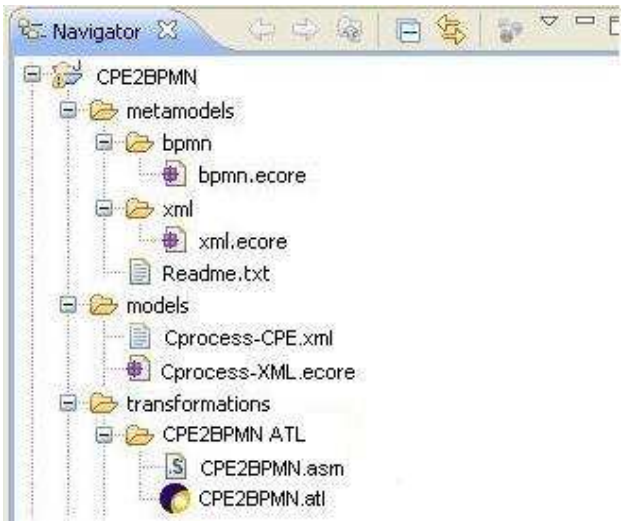


Fig.D. 6 Navigator view of the CPE2BPMN project

The execution of the transformation can be done by *the ATL run launch configuration wizard*. This wizard is composed of three distinct tabs: ATL Configuration, Model Choice, and Common. These three tabs are where we specify the source and target meta-models, the source model, and the ATL file.

Another way of execution is to launch it programmatically with Java API using the *AtlLauncher* class, as mentioned in the injection of an XML model section. This method corresponds to our development objective since we can exploit it with the Eclipse platform, and we can automatically execute the XML injection and ATL transformation at the same time. However, with this class, we manipulate the ASM file, not the ATL one.

Fig.D.7 shows the generated BPMN diagram (*Cprocess-BPMN.ecore*) after execution, conforming to the BPMN meta-model. This diagram corresponds to the input model (*Cprocess-CPE.xml*, Fig.D.4), which contains two pools (*A* and *B*), activities inside the pools, edges (sequence flows), and messages (message flows).

Fig.D. 7 BPMN diagram obtained from the transformation (Ecore view)

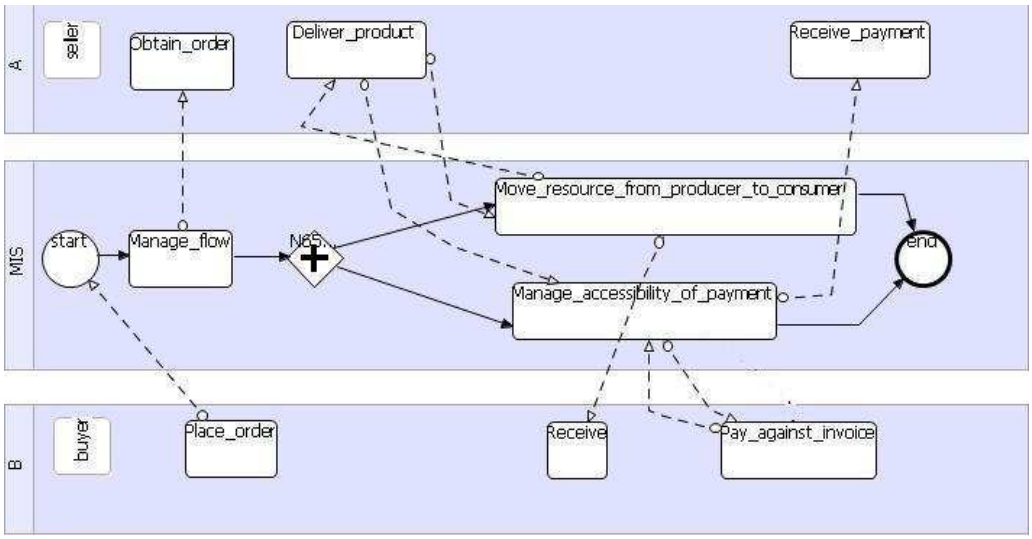


Fig.D. 8 Collaborative process between A and B

The process starts by receiving an *order* sent from the *place order* service of *B*. This *order* is entered into the *manage flow* service for transfer to the *obtain order* service of *A*. Then, the *manage flow* service informs the other two services of the *MIS* in order to advise the *deliver product* service to prepare the delivery. The *deliver product* service of *A* ships the *goods* to the *receive* service of *B* via the *move resource from producer to consumer* service of *MIS*. The

deliver produce service of A also sends the *invoice* to the *manage accessibility of payment* of *MIS* in order to forward it to the *pay against invoice* of *B*. The *pay* service of *B* sends the *payment* to *A* via the *manage accessibility of payment* service of *MIS*.

Annex E: XML and BPMN meta-models

E.1 B.1. XML meta-model (Ecore diagram)

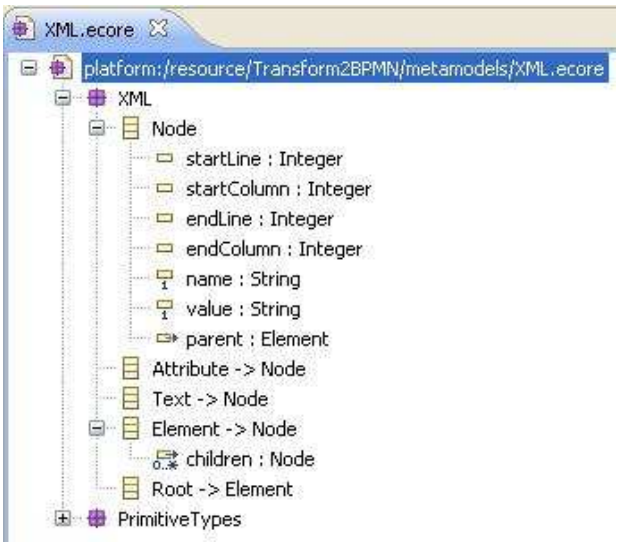


Fig.E. 1 XML meta-model

E.2 BPMN meta-model (Ecore diagram)

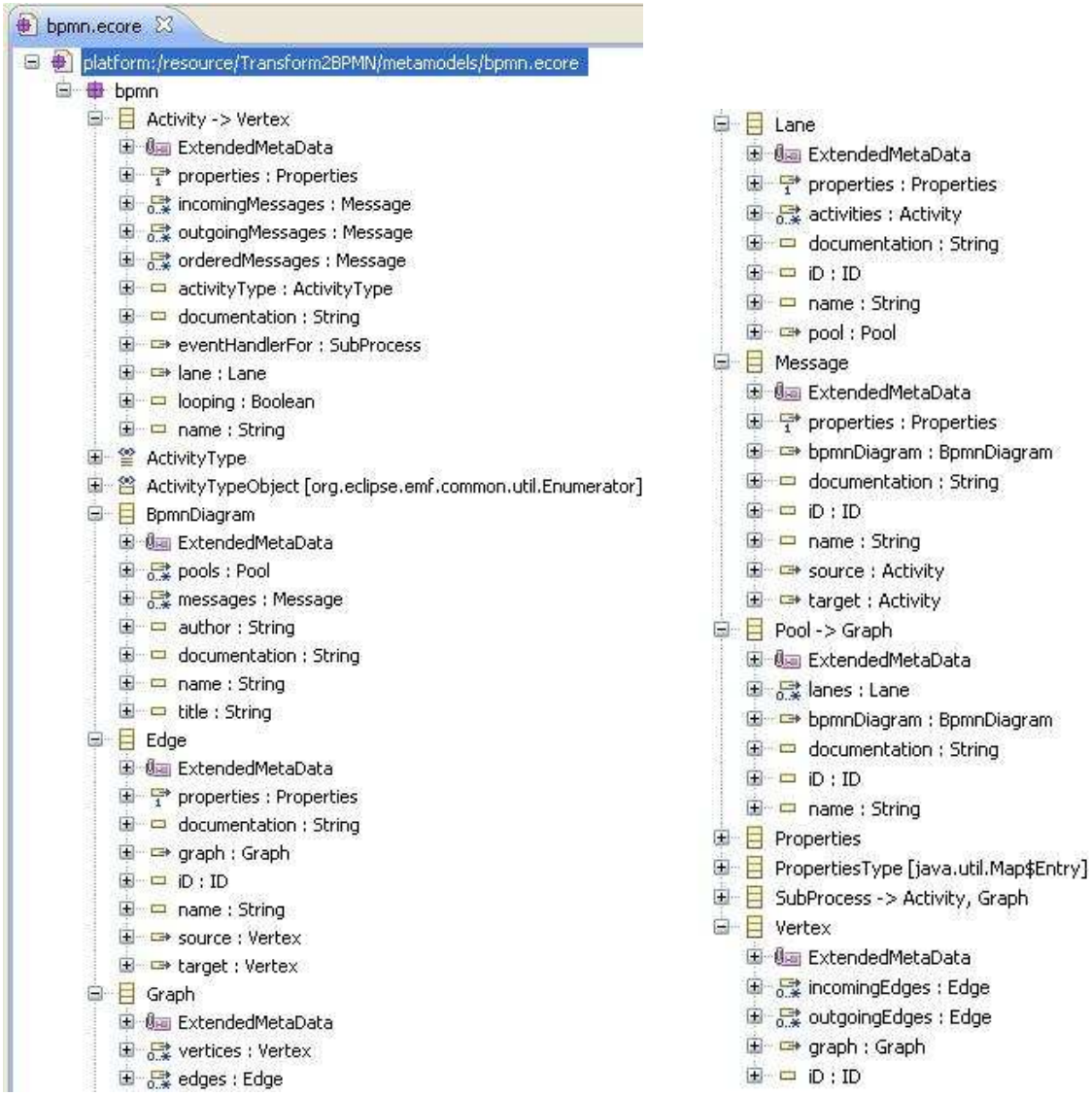


Fig.E. 2 BPMN meta-model

Annex F: SWRL Execution

The execution of SWRL rules requires an inference engine. An inference engine is a computer program that tries to derive answers from a knowledge base. It is considered as a brain of an expert system used to reason about the information in the knowledge base for the ultimate purpose of formulating new conclusions [Wiki Inference engine].

Numerous inference engines work well with Java, and many are available as open source software. Some of the most popular engines are Jess, Algernon, SweetRules, Jena, Pellet, OpenRules. At the present stage, we have no possibility to select inference engines because the only inference engine that can work with the SWRL Editor is the Jess engine. Developers of the SWRL Editor chose Jess as the first integration because it works seamlessly with Java, and has an extensive user base [O'Connor et al., 2005]. Jess provides both an interactive command line interface and a Java-based API to its rule engine. This engine can be embedded in Java applications and provides a flexible two-way runtime communication between Jess rules and Java. Even though Jess has many advantages, it is not open source but can be downloaded free for a 30-day evaluation period and is available free to academic users.

The interaction between SWRL rules and the Jess engine is user-driven. The user controls when OWL knowledge and SWRL rules are transferred to Jess. This means when inference is performed using that knowledge and those rules and when resulting Jess facts are transferred back to the Protégé as OWL new knowledge. Running the SWRL rules over the Jess engine can be done directly on the Protégé or by invoking the rules with Java.

For the direct execution on the Protégé, we have to click on the *J* button (on the top right of the SWRL Editor) to open the Jess Rules tab. Then, pressing the three following buttons in order: *OWL+SWRL* → *Jess* to transfer SWRL rules and relevant OWL knowledge to Jess, *Run Jess* to run the Jess rule engine, and *Jess→OWL* to transfer the inferred Jess knowledge to OWL knowledge. Fig.F.1 shows these four buttons:

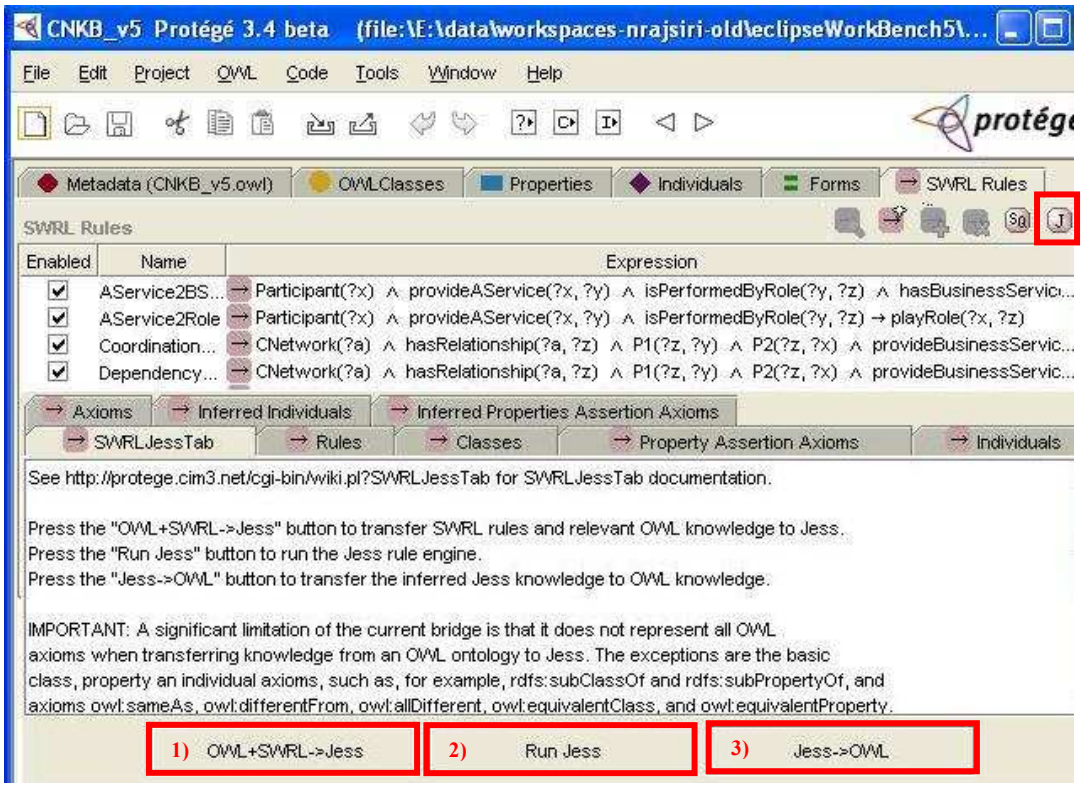


Fig.F. 1 SWRL Editor and Jess tabs

Another method of execution is to do it programmatically with Java API by using *SWRLRuleEngineBridge*. The SWRL Rule Engine Bridge is a subcomponent of the SWRL Editor that provides a bridge between an OWL model with SWRL rules and a rule engine. Its goal is to provide the infrastructure necessary to incorporate rule engines into the Protégé to execute SWRL rules [SWRL-FAQ]. Executing in this way corresponds to our development objective because we can do it automatically and exploit it with the Eclipse platform. Fig.F.2 shows the use of Java API for programmatically executing SWRL:

```
SWRLRuleEngineBridge bridge = BridgeFactory.createBridge("SWRLJessBridge", owlModel);
```

Fig.F. 2 SWRL Rule Engine Bridge

Annex G: Ontology development tools

This annex is focused on studying tools that allow developing OWL ontologies. In the last years, the number of environments and tools dedicated to ontology development has grown exponentially. [Gomez-Perez, 2004] distinguished two groups of ontology development tools:

- Tools whose knowledge model maps directly to an ontology language. These tools were developed as ontology editors for a specific language. For example, Ontolingua Server with Ontolingua and KIF, Ontosaurus with Loom, WebOnt with OCML, and OilEd with OIL and DAML+OIL.

- Integrated tool suites which have an extensible architecture, and whose knowledge model is usually independent of an ontology language. These tools provide a core set of ontology related services and are easily extended with other modules to provide more functions. For example, Protégé, WebODE, OntoEdit, TopBraid, and KAON.

In this section, we present the most well known and widely used ontology tools available on the market: Protégé, TopBraid, and KAON. These three tools are classified under the second group.

G.1 Protégé

Protégé[21] is a free, open source ontology editor and knowledge-base framework. It is the most widely-used ontology creation tool in the market. The first Protégé was created in 1987 by the Stanford Medical Informatics (SMI) group at Stanford University in order to simplify the knowledge acquisition process for expert systems.

According to [Noy et al., 2001], Protégé is based on Java. It is extensible and provides a plug-and-play environment that makes it a flexible base for creating and integrating easily new extensions like plug-ins. It implements a rich set of knowledge modelling structures that support the creation, visualization, and manipulation of ontologies in various representation formats. Ontologies can be edited and created using RDF/OWL script language (including OWL Lite, DL, and Full). Protégé ontologies can be exported into RDF(S), XML Schema, OWL, N-Triple, and TURTLE formats.

The Protégé platform supports two main ways of modelling ontologies:

- The Protégé-Frames editor enables users to build and populate ontologies. It implements a knowledge model which is compatible with the Open Knowledge Base Connectivity protocol (OKBC). Users can organize tabs (e.g. class, form, instance tabs) that each focuses on a particular aspect of ontology building. The features of Protégé-Frames include: 1) a set of user interface elements for modelling knowledge and entering data, 2) an extensible plug-in architecture such as ontology visualisation and reasoning, and various storage formats (e.g. RDF, XML, etc.), 3) a Java API for

[21] http://protege.stanford.edu/

plug-ins and other applications to access, use, and display ontologies created with Protégé-Frames.

- The Protégé-OWL editor enables users to define logical class characteristics as OWL expressions. Users can also load and save OWL and RDF ontologies, as well as edit and visualise classes, properties, and SWRL rules. This editor makes it possible to execute reasoners such as description logic classifiers, and edit OWL individuals for Semantic Web markup. Protégé-OWL provides a flexible architecture which makes it easy to configure and extend the tool. It is tightly integrated with Jena and has an open source Java API for the development of user interface components or arbitrary Semantic Web services.

G.2 TopBraid

TopBraid Composer[22], a component of TopBraid Suite, is a modelling tool for the creation and maintenance of ontologies. It is a professional development environment for W3C's Semantic Web standards RDF(S), OWL, SPARQL Query Language, and the Semantic Web Rule Language (SWRL).

TopBraid Composer enables individual users and communities to collaborate effectively in developing Semantic Web ontologies. It provides a comprehensive set of features to cover the whole life cycle of semantic application development. In addition to being a complete ontology editor with refactoring support, the Composer also can be used as a run-time environment to execute rules, queries, reasoners, and mash-ups. The current release is version 2.6.2. The key features[23] include:

- Standards-based, syntax directed development of RDFS, and OWL ontologies, SPARQL queries, and rules
- Re-use of the legacy models and data through XML, UML, spreadsheet, and database schema imports
- Visualisation and diagramming
- Support for re-factoring within and across ontologies
- Consistency checking and debugging

The Composer is built upon the Eclipse platform and uses Jena as its underlying API. It is a very flexible platform which can be extended with custom Java plug-ins. As it is implemented as an Eclipse plug-in, users can use a single tooling environment for many different modelling tasks with other languages such as UML and XML. The Composer can be used to edit RDFS and OWL files in various formats. It also provides scalable database backends **(**e.g. Jena**,** AllegroGraph, Oracle 10g, and Sesame**),** as well as multi-user support.

TopBraid Composer is delivered with the OWL DL and Pellet reasoner. Additional inference engines can be integrated and specified in the configuration preferences.

[22] http://www.topquadrant.com/topbraid/composer/index.html
[23] http://www.semantic-web.at/1.11.resource.371.topbraid-composer.htm

Although the software can be downloaded free of charge, we need to purchase a license key to use it after a 30 day evaluation period.

G.3 KAON

According to [Gomez-Perez, 2004], KAON (KArlsruhe ONtology) tool suite[24] has been developed by the Institutes AIFB and FZI at the University of Karlsruhe [Maedche et al., 2003]. The KAON tool is provided under an open source licence at http://sourceforge.net/projects/kaon. It was developed with a flexible and scalable architecture. KAON is completely implemented in Java and can be extended with plug-ins. It has three different layers as shown in Fig.G.1:

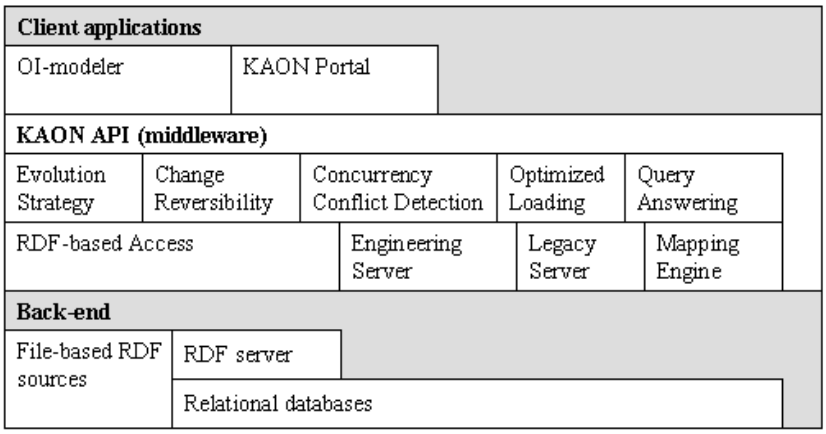


Fig.G. 1 KAON tool suite architecture [Maedche et al., 2003]

- Backends layer is for ontology storage. There are three different backends but with the same knowledge: RDF(S) files, RDF(S) models stored in a relational database with the RDF server, and KAON models stored in a relational database with the KAON Engineering server. The RDF server and the Engineering server run on top of the JBOSS application server.

- Ontology middleware layer provides the most important KAON services. One of them is the KAON API for accessing KAON ontologies. Other middleware services are, for example, an ontology evolution service that detects the implications of changes in ontology terms, a change reversibility service that keeps track of ontology changes, a concurrency conflict detection tool for collaborative ontology engineering, a query answering service, etc.

- A client application layer accesses the ontology middleware and provides end-user applications. Client applications have been already created, such as OI-modeller which is an ontology editor for KAON ontologies, and KAON Portal which allows creation of ontology-based web portals.

[24] http://kaon.semanticweb.org

The knowledge model underlying KAON is based on an extension of RDF(S). We can model ontologies with concepts, properties, and instances of concepts and of properties. The knowledge model also attaches labels, documentation, synonyms, and word stems in different nature languages to concepts, properties, and instances. However, KAON cannot support more complex components like formal axioms.

G.4 Conclusion

The study of different ontology development tools is aimed at choosing an appropriate tool for developing our OWL ontology. Numerous ontology development tools have evolved over the last decades. Two groups of tools have been distinguished by [Gomez-Perez, 2004]: tools for editing ontologies in a specific ontology language, and tools created as integrated extensible tool suites. We focus more in the second group because it is possible to extend them and add new functions.

The selection of an appropriate tool for developing our ontology was based on certain criteria. Some of them have been delivered by [Corcho et al., 2002] and [Gomez-Perez, 2004]. They are listed as follows:

- Availability: license required for using the tools.
- Extensibility capability: possibility to extend, develop or add more functions.
- Ontology storage: how the tools store ontology in simple text files or in database?
- Language support: what ontology languages the tools support?
- Collaborative editing: possibility to edit the same ontology with other users simultaneously.
- Inference engine: engines provided by the tools in order to perform constraint checking on the ontologies built.
- Consistency checking: capability of the tools to check and ensure, by using the inference engines, that the ontologies built with them have no errors.
- Import / Export: the interoperability of the tools with other tools. It presents how we can use the ontologies developed with each tool and implemented in other languages or other tools.

The Table G.1 compares these three ontology development tools (Protégé, TopBraid, and KAON) by basing on the above criteria:

Tools/ Criteria	**Protégé**	**TopBraid**	**KAON**
Availability	Open source	License needed	Open source
Extensibility	Yes (plug-ins)	Yes (plug-ins)	Yes (application servers)
Ontology storage	Files (.rdf, .owl) DBMS	Files (.rdf, .owl) DBMS	Files (.rdf, .owl) DBMS
Editing	OWL, RDF(S)	OWL, RDF(S)	OWL, RDF(S)
Collaborative editing	No	Yes	Yes
Inference engine	Jess, FaCT, Racer	Pellet	N/A

Consistency checking	Yes	Yes	Yes
Import	XML, XSD, XMI, RDF(S)	UML, XSD, Spreadsheet	KAON, RDF(S)
Export	XML, XSD, XMI, RDF(S), Flogic, Java	XML, Spreadsheet, display on map or calendar	KAON, RDF(S)

Table G. 1 Comparative of three ontology development tools

As shown in the table, Protégé and KAON are open source, while TopBraid is distributed under license. All three tools have extensibility capabilities. Protégé and TopBraid allow easy extension by means of plug-ins. KAON is based on application server architectures which also provide good extensibility features and multi-user support. These three tools store their ontologies in text files and also are able to store in a database. Thus, this makes it possible to manage larger ontologies. They have all been developed in Java. The editing capabilities in terms of language that each tool supports, as well as editing ontologies collaboratively are mostly the same, except Protégé which is not possible for collaborative editing. Some inference engines are listed in the table for each tool. Protégé provides more inference engines than others. TopBraid is delivered with Pellet in its package. All of them can check the consistency of ontologies created. The import and export features concern the interoperability aspect of each tool. Protégé is able to import and export more formats than the others, while KAON format can only be used with KAON tool.

From the above discussion, Protégé and TopBraid are not too different, except for the availability and collaborative editing issues. Since TopBraid is a commercial platform, it is more stable than Protégé [Knublauch, 2006]. Based on our objectives and point of view, the tool we will use will have to interoperate with the other tools in our prototype (see Chapter 4), especially GMF (Graphical Modeling Framework). Thus, the import and export capabilities are also the outstanding criteria for us. Protégé seems to be a better choice than the others because it can deal with XML and XMI formats which are the main formats we use in interoperating between the different tools in our prototype.

Abstract:

Enterprises are now operating in an environment where market is more open, globalized, and competitive. Changes in market conditions are obliging enterprises to become involved in various kinds of industrial networks in order to maintain their business efficiency. The integration of business partners depends deeply on the ability to capture and share information seamlessly amongst the information systems (ISs) of different enterprises. The MISE (Mediation Information System Engineering) project was evolved in order to tackle this problem by providing an information technology solution for supporting the enterprise interoperability through ISs. It is developed on the basis of the MDE (Model Driven Engineering). This dissertation addresses the business level of the interoperability, and the CIM (Computer Independent Model) of the MDE. Its main objective is to develop a knowledge-based system for supporting the design of collaborative processes that conform to the BPMN (Business Process Modeling Notation). We propose to work at the upper level of the CIM to capture knowledge that allows us to characterize collaboration by basing on the perspectives and experiences of business partners. We use this knowledge together with the existing knowledge (instances about business processes) from the MIT Process Handbook for moving down to the CIM level. The prototype of our knowledge-based system is also developed in order to validate and evaluate the approach.

KEYWORDS: Process, Collaboration, Interoperability, Information system, Knowledge-based system, Ontology, Rule, Deduction, Model-driven architecture

Résumé:

Le marché industriel est aujourd'hui de plus en plus dynamique et compétitif. Cette tendance évolutive de l'écosystème amène les entreprises à prendre part à un nombre croissant de réseaux industriels, dans l'optique de maintenir leur activité et d'accroître leur compétitivité. La qualité d'interaction et de collaboration de partenaires de ces réseaux dépend grandement de la capacité de leurs systèmes d'information (SIs) respectifs à gérer et à partager les informations. Le projet MISE (Mediation Information System Engineering) relève pleinement de cette problématique en proposant une approche de conception d'une solution (conceptuelle et technologique) pour le support de l'interopérabilité d'entreprises au travers de leurs SIs. Ce projet s'appuie sur la notion de MDE (Model-Driven Engineering) et s'articule autour de trois niveaux : métier, logique et technologique. Les travaux de thèse dont il est ici question relèvent du niveau métier en présentant une démarche d'obtention d'un modèle indépendant de toute implémentation (CIM pour Computer Independent Model). Il s'agit en particulier de s'appuyer sur un système basé sur la gestion de connaissance pour concevoir des processus collaboratifs en BPMN (Business Process Modelling Notation). En se positionnant à un niveau d'abstraction au dessus de celui du CIM, on peut capitaliser, manipuler et raisonner une connaissance permettant d'une part de caractériser des collaborations et d'autre part de mettre en place des mécanismes de déduction pour descendre au niveau de CIM. Ces principes sont en outre illustrés par le biais d'un prototype développé pour valider l'approche.

MOTS-CLES : Processus, Collaboration, Interopérabilité, Système d'information, Système basé sur connaissances, Ontologie, Règle, Déduction, Model-driven architecture