Int. J. Production Economics 121 (2009) 5–20

Contents lists available at ScienceDirect

Int. J. Production Economics

journal homepage: www.elsevier.com/locate/ijpe

A model-driven approach for collaborative service-oriented architecture design

Jihed Touzi [a,*], Fréderick Benaben [a], Hervé Pingaud [a], Jean Pierre Lorré [b]

[a] Ecole des Mines d'Albi-Carmaux, Campus Jarlard, 81013 Albi Cedex 09, France
[b] EBM WebSourcing, 10 av. de l'Europe, parc technologique du canal, 31520 Ramonville St Agne, France

ARTICLE INFO

Article history:
Received 11 December 2007
Accepted 23 September 2008
Available online 18 April 2009

Keywords:
Process Modelling
BPMN
Information system
Transformation rule
MDA
SOA

ABSTRACT

In a collaborative context, the integration of industrial partners deeply depends on the ability to use a collaborative architecture to interact efficiently. In this paper, we propose to tackle this point according to the fact that partners of the collaboration respect the Service-Oriented Architecture (SOA) paradigm. We propose to design such a collaborative architecture according to Model-Driven Architecture (MDA) principles. We aim at using business models to design a logical model of a solution (logical architecture) as a principal step to reach the final collaborative solution. This paper presents the theoretical aspects of this subject and the dedicated transformation rules.

1. Introduction

It is now widely recognized that for sustainability reasons, small- and medium-sized enterprises (SMEs) should be involved in many kinds of industrial networks in order to maintain their business efficiency. Such collaborations can be described in many ways, such as:

- in a given value chain, a group of specialized organizations with complementarities decides to develop jobs together in order to achieve a common goal (supply chain model);
- a group of relatively similar organizations decides to sign an alliance in order to achieve the critical capacity required by an offer;
- etc…

In parallel to these networked business strategies, new requirements are specified for the definition of the collaborative platform that will support collaboration between organizations. The diversity of business process categories to develop inside the network is as large as the variety of types of collaboration between those business organizations. A network is a living, open system that evolves and adapts its processes regularly, as does a single organization. Thus, using the term "collaboration" we seek to describe the widest of industrial network configurations. For each partner, the basic problem is to be able to establish fruitful connections with others at low transaction costs and as quickly as possible.

Abstracting from the IEC TC 65/290/DC standard (IEC, 2005; Kosanke, 2005), we adapt the different levels of collaborative maturity that can be used to characterize an organization: *communicating* (capable of exchanging and sharing information), *open* (capable of sharing business services and functionalities with others), *federated* (capable of working with others according to a set of collaborative processes that have a common objective and to assure its own objectives) and *interoperable* (capable of working with others without a special effort so that, from the external point of view, the set of enterprises appear as a homogeneous and seamless system). Interoperability, which is the ultimate rung of

* Corresponding author. Tel.: +33563493138; fax: +33563493183.
E-mail address: jihed.touzi@enstimac.fr (J. Touzi).

0925-5273/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2008.09.019

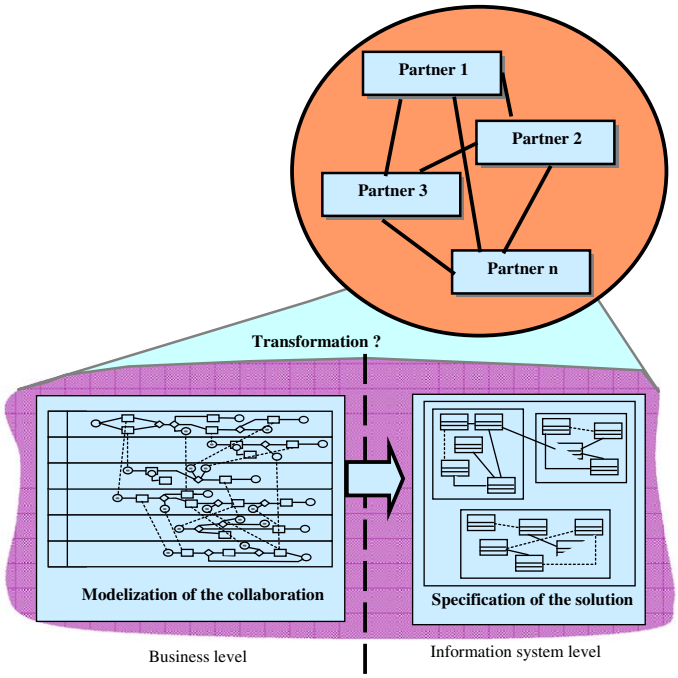

Fig. 1. From a description of the collaboration to a specification of the solution.

the collaborative maturity ladder, appears as a concept that facilitates the ease of partner connectivity.

An information system is based on a set of software applications that allows one organization to manage and progress in its business. The efficiency of exchange of information and documents with new partners deeply depends on the capability of the information system to be interoperable.

Because of the organization's heterogeneity at cultural, linguistic, business and technological levels, the design of solutions for interoperability of heterogeneous information systems is a quite complex problem. The interoperability of an organization through its information system has been the subject of intensive research recently. The problem tackled in this article is about how to bridge the gap between the business level (a set description of how partners in the network collaborate) and the information system level (to find, configure and assemble components of the partner's information systems). The question is about transforming a business knowledge about the collaboration itself to a technical knowledge about how to make information systems of partners interoperable (Fig. 1).

The role of specific models describing, from the one hand, the collaboration and from the other hand, the solution should be to enable the transition between the two levels, i.e., it should be driven by models (model-driven). The first kind of models show business aspects of the collaboration like roles implied, synchronization of activities, messages exchanged, etc. The second kind of models describe the technical solution based on a logical choice of well-defined architecture: components, services, etc.

Models should be considered according to semantic and syntactic points of view. From the syntactic point of view, models allow to represent a knowledge needed in the different steps of the design of the final software. From the semantic point of view, models must be well understood and semantically agreed. If the semantic point of view is crucial to share different models provided by heterogeneous partners, it exists today in a number of architectures, proposals and design processes that help to formulate correctly models at the semantic level. Some are released with international standards (e.g., ISO), others are developed at regional or national level (e.g., CEN) and others are developed by independent project teams and groups (e.g., OMG, W3C, IAI). Most of the standards cited, have been developed in strong contact with industry needs.

Nowadays, the model driven approach is followed by numerous projects and communities like INTEROP (2007) in the European Union and model-driven architecture (MDA) (OMG, 2003a, b), which is carried out by the Object Management Group (OMG).[1] MDA, for instance, intends to promote the use of models as fundamental way of designing and implementing different kinds of systems. This article intends to provide an innovative methodology to develop a collaborative architecture (that provides interoperability capacity to partners) following the MDA approach. The article is structured as follows: In Section 2 we present an overview of approaches and architectures that facilitate the establishment of the interoperability. The Service-Oriented Approach (SOA) will be presented in

[1] www.omg.org.

this section. The theoretical aspects of this subject and the dedicated formalized transformation rules are detailed in Section 3. Section 4 describes the developed prototype to illustrate our work. Section 5 gives an example of application of the presented transformation rules. Section 6 presents the conclusion of this work and areas of future research.

2. Overview of approaches and architectures for interoperability

Interoperability can be defined as "achieved only if the interaction between two systems can, at least, take place at the three levels: data, resource and business process with the semantics de-fined in a business context" (Chen and Doumeingts, 2003). Interoperability is one possibility for realizing an integration, not the only one (Vernadat, 2006), but it promotes the idea that integration has to be prepared using standards, reference frameworks or specific architectures and approaches so that the act of connecting to others appears to be as much as possible as a plug-and-play action.

As cited above, the problem of enterprise interoperability concerns three levels: data, resources and business processes. Different research works define frameworks to characterize interoperability levels: *European Interoperability Framework* (EIF) (EIF, 2004), *ATHENA Interoperability Framework* (AIF) (ATHENA, 2004), *Interoperability Development for Enterprise Applications and Software* (IDEAS) (IDEAS, 2003) and *e-Government Interoperability Framework* (e-GIF) (e-Gov, 2005).

EIF and e-GIF focus on interoperability in the e-Government/e-Administration domain but the levels they present (organisational, semantic, technical) are compatible with the industrial domain. The IDEAS framework defines three levels: *business* (business context and processes of organizations), *knowledge* (definition of products, competencies, etc. in the organization) and *ICT systems* (applications and communication infrastructure) and a transversal level of semantics to assure a mutual understanding of the three levels mentioned above. AIF adopts a holistic approach of interoperability that allows a good analyse of interoperability needs: it concerns meta-models, concepts, formalisms and standards that help to formalize the different levels of interoperability (i.e., a process model presents interoperability characteristics on an organizational level).

According to these frameworks, we can deduce that the problem of interoperability deals globally with *organizational*, *conceptual*, and *technical* issues:

- at the organizational level, the business context of the collaboration must be explained: how do partners interact? Which data are exchanged? Which resources do they expose to others? Process and data models are examples of solutions for modelizing interoperability at this level;
- at the conceptual level, data, resources and business processes of different information systems must be linked in spite of their heterogeneous structures and different interpretations. The problem is both syntactic and semantic and
- at the technical level, the aim is to reconcile the different applications, technologies, systems and communication infrastructures used by the partners.

Defining the final collaborative solution that meets the interoperability requirements is not an easy task. In a distributed environment of a collaboration, technical components (database, ERP, web service, etc.) of partner's information systems should work together to answer the business needs expressed by partners. The selection and the configuration of these components is not only a problem at technological level but it should be aligned with conceptual and organizational levels. Despite the fact that interoperability problems usually occur at a horizontal level (partners' heterogeneity of processes, data and applications), the problem tackled in this article could be seen as a vertical interoperability problem. Indeed, our contribution allows to go down from the organizational level to the technical level, according to the MDA principles.

A critical choice to do in the development of interoperability solutions according to the MDA paradigm is the definition of a target logical architecture independently of platform considerations. "Interoperability is achieved if two (or more) systems can exchange information and use the information in manner for which they have the basic capability" (IEEE, 1990). If according to the last definition, interoperability seems to allow an easy and open access to information system resources, it is important that interoperability must be controlled. We need to manage interaction between the organization's collaborative (public) and internal (private) processes. Only a public part of an organization's information system will be visible to other partners, most of the other part remains invisible for competitive and strategic reasons. Service-oriented architecture is a perfect solution to answer these expectations. SOA allows organizations to achieve the necessary wide integration through software interfaces. These interfaces called "services" can be easily adaptable, reconfigurable and reusable in new collaborations.

If services represent a good answer to technical and syntactical interoperability issues, they fail in the semantic one: witch service is needed exactly to answer this specific business need? Research works in enterprise ontology and semantic web services (Missikoff and Taglino, 2006) try to propose some tracks of solution. If this subject is not the heart of our contribution, we believe that a pragmatic way to tackle the semantic correspondences between business needs and IS specifications is based on the definition of an architectural framework which defines related formalisms, meta-models and the linked transformation mechanisms. The work presented in this paper has this objective.

The collaborative architecture that we aim to develop, conforming to the MDA principles, respects the SOA vision of designing collaborative systems. In the following, we present briefly the basic characteristics and principles of

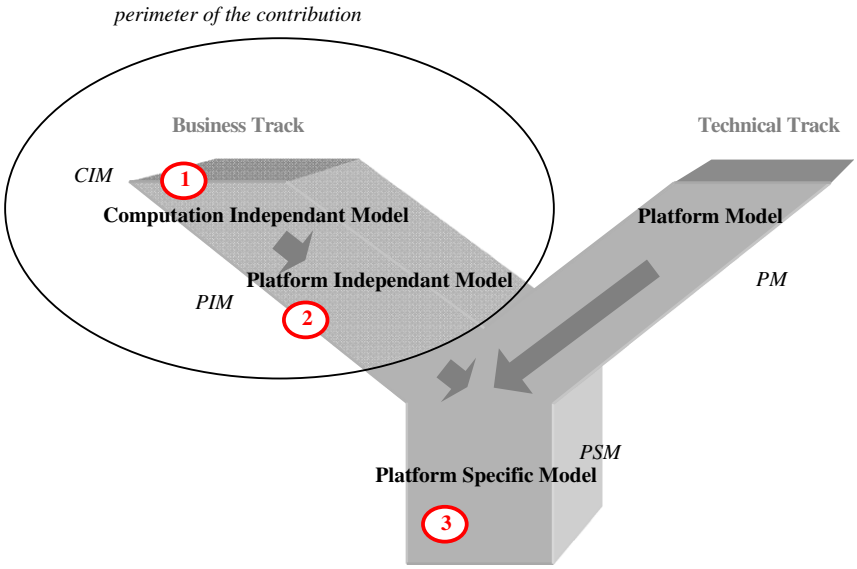

Fig. 2. Model-driven architecture.

MDA and SOA and how we define our contribution according to these two architectures.

2.1. Model-driven architecture

The Object Management Group has been proposing the MDA approach as a reference to achieve wide interoperability of enterprise models and software applications. Two main aspects are essential in the engineering principles promoted by MDA:

- use of different models at each abstraction levels: from conceptual (CIM, or Computer-Independent Model) to logical (PIM, or Platform-Independent Model), and from logical to physical layers (PSM, or Platform-Specific Model). The models are in closed connections and transformation mechanisms facilitate passage from one layer to another and
- separation of concerns by segregating implementation choices from business needs specifications (business track). Technology is defined by the choice of the implementation platform in a generic way (technical track). In fact, the ultimate solution is a mix of information coming from these two tracks, processed to produce the PSM.

The Y symbol is frequently used to summarize these principles, as shown in Fig. 2.

As cited above the transition from one level to another is based on model transformations. A model transformation can be seen as morphism between elements of two models. A meta-model allows fixing the syntax and the semantic of the different elements that compose a model. Morphism between two models is explained as a mapping between the elements of two related meta-models. On the basis of the defined mappings, a transformation can be done to link two models. By executing a model transformation, models conforming to the source meta-model are transformed to models conforming to the target meta-model. This is crucial in our problematic of transforming a collaborative process model into an information system model: firstly, we have to define the two meta-models of the collaborative process and of the collaborative architecture model and secondly, we have to define the transformation rules based on established mappings between the different elements of the two meta-models. The model-driven interoperability (MDI) proposal (Grangel Seguer et al., 2007) attempts to provide solutions that, following the MDA approach, can help enterprises to transform models at different levels of abstraction in order to generate enterprise software applications (ESA) from enterprise models[2] and how a model-driven approach could be a useful way to solve interoperability problems. An application of the MDI approach is described in Grangel Seguer et al. (2007). Authors explain how GRAI (Doumeingts et al., 1998) extended actigrams can be transformed into unified modelling language (UML) activity diagrams at the CIM level. If the MDI proposal defines meta-models needed to represent the transition (enterprise model/ESA), there are not transformation rules explicitly defined and the propositions still be without implementations of prototype to show the feasibility of the approach. Our contribution presents clearly a formalized set of transformation rules (under a set of preliminary assumptions).We have developed also, in our research work, a prototype using a transformation model tool to illustrate our work.

[2] Enterprise modelling aims to describe practices in enterprises from several points of view: functional, physical, business process, decisions, information, etc.

2.2. Service-oriented architecture

SOA is based on the fundamental idea that an information system is no more than a collection of easily accessible services that can be dynamically connected in order to provide the desired solution (Vernadat, 2006; Maamar et al., 2005). Choosing a SOA approach seems to be a suitable candidate for tackling the complexity of interoperability establishment. SOA allows 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 using services and have independent lifecycles. Indeed, enterprise applications and internal processes can be encapsulated as services. A service is the key concept of the SOA paradigm. It is a discrete piece of functionality (of the enterprise) that appears to be atomic and self-contained from the point of view of the service consumer. Services communicate using a set of messages as input and output. Each message has a particular structure. It can be a complex business object (purchase order, invoice, etc.).

Schematically, 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 changing business priorities. The consumer of a service has to ask a third-party 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.

While web-services technology provides support for many SOA concepts, it does not implement all of them. Moreover, service consumers can execute web services directly if they know the service's address and contract.

The design of collaborative solutions respecting SOA considerations has become one of the major topics in the domain of interoperability. As example, the platform-independent model for service-oriented architecture (PIM4SOA) project (Benguria et al., 2006) aims to develop a meta-model for SOA. This meta-model consists of a set of essential aspects for SOA. PIM4SOA addresses four system aspects (views): processes (logical order in terms of actions, control flows and interactions between services), information (related to the messages or structures exchanged by services), services (description of services: access, operations and types) and quality of services (extra-functional qualities that can be applied to services, information and processes). The project also provides a set of transformations that link the meta-model with specific platforms (agents, web services, etc.) following the MDA approach. However, transformation rules and mappings between PIM and PSM levels are not explicitly explained in the project.

Our contribution is presented as follow: from a collaborative process model (CIM level), we want to deduce, using transformation rules, a SOA model (PIM level) related to a services collaborative solution, a vertical transformation in MDA vocabulary. Our approach is close to the MDI approach cited. Indeed, on the one hand, a collaborative process describes in a disproportionate way views of enterprise modelling. We consider that the most powerful means to tackle one collaboration of partners is to handle the associated collaborative process. The increasing interest in the field of business process management (BPM) shows the central position of processes in the definition of collaborations. On the other hand, the SOA model generated represents a logical solution (independent of technical considerations). The interest of the model obtained is that it can be used to generate others specific platform assets (agents architecture, components architecture, etc.). In our work, the SOA model generated is the fundamental part of a wider solution that addresses implementation of an enterprise service bus (ESB).[3] A number of questions have been done: Which process modelling formalism to represent collaborative process? Which language to represent generated SOA models? What about meta-models definition and the requisite transformation rules?

3. Model-driven approach for collaborative service-oriented architecture design

The transformation from a business requirement level (collaborative process model) to a SOA infrastructure requirement level (information system model) is not an easy task. We need to specify languages and formalisms needed for the definition of each level. A meta-model for each level has to be defined later. The main entities of the steps of our approach are described below.

3.1. Collaborative business process modelling

The aim of a process model is to depict interactions between two or more business entities. Currently, there are scores of business process modelling languages, tools and methodologies. They can be classed according to defined maturity levels. In a collaborative context and due to the complexity of interactions between partners, an adapted process modelling language must be used. For example, specific attention must be paid to the private/public considerations in the modelling of the collaboration. The business process modelling notation (BPMN) (BPMI, 2004) is an adapted answer to current needs in the field of the collaborative process modelling. The adoption of BPMN standard notation will help unify the expression of basic business process concepts (e.g., public and private processes, choreographies) as well as advanced modelling concepts (e.g., exception handling, transaction compensation).

The objective of the BPMN formalism is to support process management by both technical and business users. Interactions in BPMN are represented using the "*message flow*" concept which shows an exchange of data between two actors of the process. These actors are

[3] ESB is a technology which implements a SOA pattern based on a distributed lightweight web services approach.

represented using "*pool*" concept. Pools can be divided in many "*lanes*" (different roles of an actor). There are many synchronization mechanisms in BPMN: sequencing ("*sequence flow*" concept), events ("*start event*", "*intermediate event*" and "*end event*" concepts), forking ("*parallel gateway*" concept), conditioning ("*data-based gateway*" and "*event-based gateway*" concepts), etc. The reasons why we have chosen BPMN are because this formalism is sufficiently rich and expressive and provides a notation that is intuitive to business users yet able to represent complex process semantics.

In the collaborative processes that we consider in our work, a special pool called "*collaborative information system*" (CIS) plays the role of a mediator[4] between different partner's information systems. This central pool contains the big part of the collaborative process and orchestrates synchronization between the different collaborative tasks of partners. This method of representation respects the public/private paradigm. Indeed, organizations are represented by their public part (collaborative tasks) in the process. They are able to interact in a different context without changing their internal processes.

3.2. Collaborative service-oriented architecture modelling

The collaborative SOA which we aim to define can be modelled using the unified modelling language (OMG, 2003a, b) which is a standard for software modelling. It is able to represent many views of the system design like functional view (or user view), structural view and behavioural view. Functional view describes competencies of the system in use context, while structural view models its global organization in terms of logical components and their interfaces. Finally, behavioural view describes scenarios, operating modes and performance of part or whole of the system. Different diagrams, gathered, give a complete description of the system. A first approach for modelizing SOA consists in representing everything as class: a service is a class, an exchanged message is a class, etc. This could make the models difficult to understand and to use. For this reason, we have developed a specific profile (based on a meta-model) to represent collaborative SOA aspects. This profile is inspired by the results of the PIM4SOA project (Benguria et al., 2006).

The collaborative architecture that we propose is an extension of the classical SOA paradigm (PIM4SOA). It contains an intermediate entity (called mediator) that manages partner's services and the execution of the collaborative process. This mediator provides also a set of "added value" services that cannot be provided by the partners in the collaboration (e.g., payment check, supplier selection). The generation of a model that represents an instantiation of the collaborative architecture defined according to a given BPMN collaborative process is the aim of this contribution.

[4] This article does not focus on the mediator concept. For more detailed information, please see Touzi (2007).

3.3. Feasibility of the BPMN–UML transformation

It is an important question to know if the BPMN model will give enough information to specify the SOA model. A BPMN model is a process-centric view of a system. In comparison with the four points of view of the ISO19440 (ISO 19440, 2005) standard (functional, resources, informational and organizational views), a BPMN model mainly covers the functional view, and the informational and organizational views only partially. The result is that the transformation will not completely provide all information needed by the SOA model. A data structure deficit is evident, because in BPMN the concept of message-flow is not well supported by data models. The data models have to be studied in parallel to the transformation of process models. Considering the resource view of the ISO 19440, services are software resources supposed to be qualified and available.

Fig. 3 shows the coverage of the different ISO 19440 views by the BPMN formalism. BPMN models allow the construction of diagrams of the behavioural and functional views (arrows A and B). For the others views (arrows C, D, E and F), we need an additional knowledge to obtain complete UML diagrams. That is the reason why we have to define a well structured collaborative architecture (the target collaborative SOA meta-model) in the MDA approach which starts from the BPMN model.

Consequently, a major part of the specification seems to be provided by the transformation of BPMN collaborative models according to the prevailing set of assumptions.

3.4. Meta-models definition and formalization

In this section, we present a definition and a formalization of the needed meta-models to perform the CIM–PIM transformation. A graphical model (UML class diagram) joined to a formal definition of the meta-model will be presented.

3.4.1. Collaborative process meta-model

The first meta-model is of the collaborative process. The BPMN language is used with a systematic approach into which pools of partners form a matrix of containers showing coordinated entities. The main BPMN formalism components appear on the class diagram of Fig. 4. The definition of the collaborative process respects two critical constraints:

- a mediator pool (called "CIS pool") must be entirely represented in the process model. This choice is interesting because the collaborative process may contain tasks that refer to collaborative or technical "added-value" services provided by a mediator entity and
- for competitive reasons, partners do not want to show their internal processes and applications. In the meta-model, partners are represented by their collaborative tasks that refer to a set of communication interfaces.

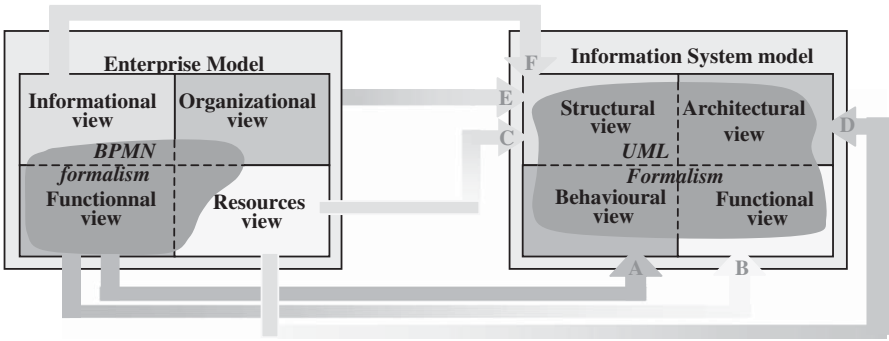

Fig. 3. BPMN–UML covers.

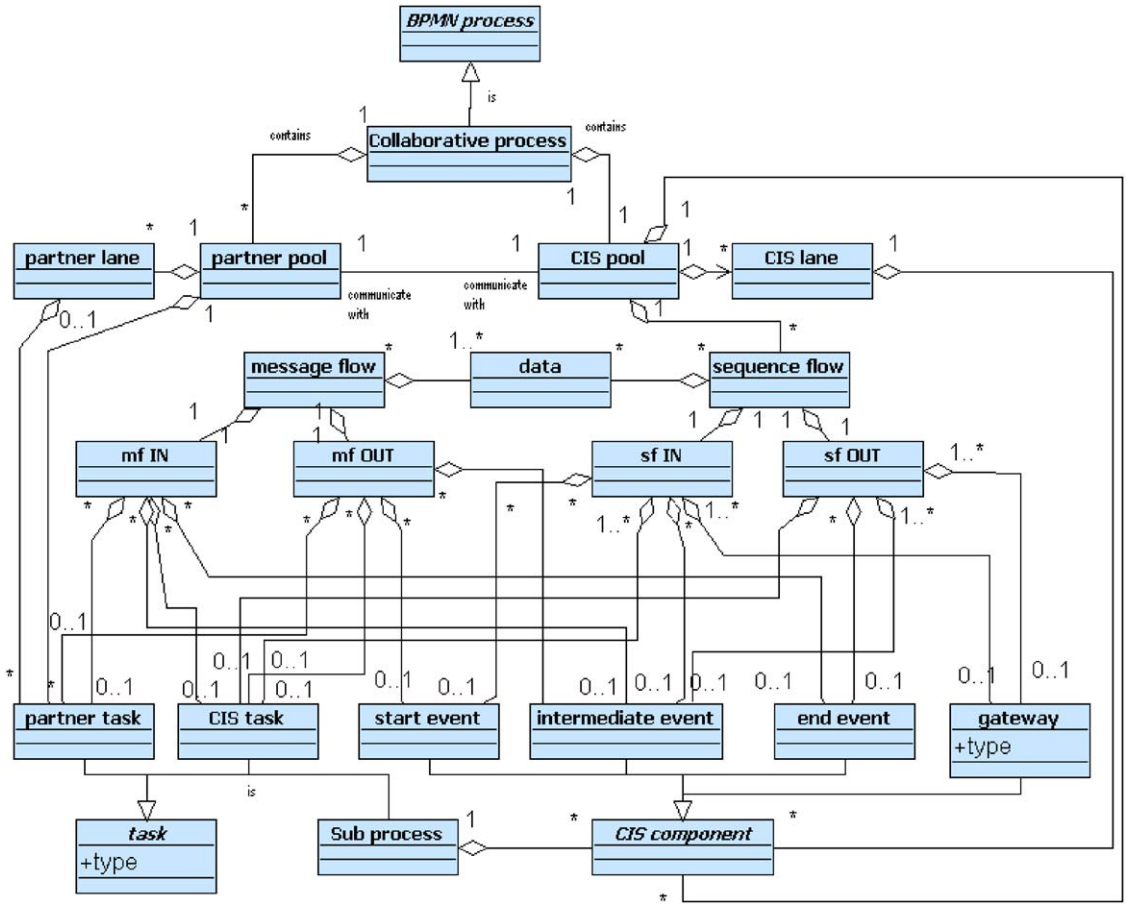

Fig. 4. Collaborative process meta-model.

A formal definition of the meta-model is described below:

Definition 1. One collaborative process model *CPM contains*

- one "*CIS pool*" p^{CIS}: the orchestration container of the process, managed by the mediator entity,
- a set of "*CIS lane*" L^{CIS} container to represent the functional divisions of the mediator of the collaboration,
- a set of "*partner pool*" P^{PAR}: containers to represent partners of the collaboration,
- a set of "*partner lane*" L^{PAR}: containers (optional) to represent functional divisions of one partner of the collaboration,
- a set of "*partner task*" T^{PAR}: interfaces of partner's information systems in the collaboration. These tasks can be of three types: Send Task T^{PARs} (when a partner sends a message to the CIS), Receive Task T^{PARr} (when a partner waits for a message from the CIS) and Service

Task T^{PARse} (when the task represents a service). T^{PARs}, T^{PARr}, $T^{PARse} \subset T^{PAR}$,

- a set of "*CIS task*" T^{CIS}: orchestration task of the collaborative process,
- a set of "*sub-process*" *Sp*: a part of a process,
- a set of "Event" *E*, an event can be partitioned into "start event" E^s, "*intermediate event*" E^i, "*end event*" E^e. E^s, E^i, $E^e \subset E$,
- E^i is composed of the subsets "*intermediate message event*" E^{im} and "*intermediate timer event*". E^{it}. E^{im}, $E^{it} \subset E^i$,
- a set of "*gateway*" *G*, composed of the subsets: "*parallel gateway*" G^p, "*data-based inclusive gateway*" G^{dbi}, "*event-based exclusive gateway*" G^{ebe} and "*data-based exclusive gateway*" G^{dbe}. G^p, G^{dbi}, G^{ebe}, $G^{dbe} \subset G$,
- a set of relations "*sequence flow*" *Sf*, where *x.Sf.y*, $x \in sfIN$ and $y \in sfOUT$ are, respectively, the source and the target element of the relation *Sf*:
  - $sfIN \subset (E^s \cup E^i \cup T^{CIS} \cup G)$, a source of a "*sequence flow*" must be "*start event*" or "*intermediate event*" or "*CIS task*" or "*gateway*",
  - $sfOUT \subset (E^e \cup E^i \cup T^{CIS} \cup G)$, a target of a "*sequence flow*" must be "*end event*" or "*intermediate event*" or "*CIS task*" or "*gateway*",
  - *Sf* may be linked to an element "*data*" *d* which presents a business object exchanged.
- a set of relation "*message flow*" *Mf*, where *x.Mf.y*, $x \in mfIN$ and $y \in mfOUT$ are, respectively, the source and the target object of the relation *Mf*:
  - $mfIN \subset (T^{PAR} \cup E^i \cup T^{CIS} \cup E^e)$, a source of a "*message flow*" must be "*partner task*" or "*intermediate event*" or "*CIS task*" or "*end event*"
  - $mfOUT \subset (T^{PAR} \cup E^i \cup T^{CIS} \cup E^s)$, a source of a "*message flow*" must be "*partner task*" or "*intermediate event*" or "*CIS task*" or "*start event*"
  - *Mf* is linked obligatory to at least an element "*data*" *d* which presents a business object exchanged.

3.4.2. Collaborative SOA meta-model

The collaborative architecture meta-model is described in Fig. 5. Three packages are proposed corresponding to three views where specific concerns of the collaboration, respecting SOA considerations, can be addressed:

- *Services view*: services that are used in the collaboration are described; they are business reachable computing functionalities with a known location on the communication network. In this view, information about addresses, operations and descriptions of partner's services are provided.
- *Information view*: data are exchanged by messages between services; they are defined here in the structure by a data model, and also as a communication utility by identification of the emission and reception services. These messages refer to business objects (invoice, order, etc.).
- *Process view*: interaction between services and coordination aspects are specified by the control of processes described here. This view deals with a specification of the orchestration of invoking services in the collaborative process.

Fig. 5 shows that in the services view, *services registry* describes a set of *partner services*. It is a container used by the CIS to find information needed about a partner service. The *CIS services* sub-package deals with a set of added value *CIS services*. In the Information view, each exchanged *message* in the collaboration has its own *format* and is described by a *semantic definition*. In the process view, traditional process modelling concepts are retained. A *collaborative process* is composed of a set of constructs that refer to the business process execution language (BPEL) standard (OASIS, 2003). *Basic activities* refer to how to deal with services of the collaboration: to invoke a service (*invoke*), to wait for a new message (*receive*) and to reply to a previous invocation (*reply*). *Structured activities* refer to how to structure the execution of the process (the logical order): parallel (*flow*), sequence (*sequence*), loop (*while*), etc. *Event handlers* manage the different events that characterize the execution of the process.

Each view is closely linked to the others two views using UML associations: in order to operate, a service (service view) needs and produces messages (information view), and the execution of one activity of the collaborative process (process view) needs to call one service (service view) to be performed.

A formal definition of the meta-model is described below:

Definition 2. An collaborative SOA Model is composed of:

- one "Services Package" pa^{ser} which contains:
  - two sub-packages "*partners services*" pa^{par} to describe partner's services and "*CIS services*" pa^{CIS} to describe business services provided by the mediator
  - one "*registry class*" c^{reg}: to manage and subscribe partner's services. It is a container used by the CIS to find information needed about a partner service
  - a set of "*services class*" C^{ser} to represent abstract services
  - a set of "*partner_service class*"C^{psr} to represent partner's services
  - a set of "*partner_service_description class*"C^{psd} to describe partner's services
  - a set of "*enterprise_division attribute*" A^{edi}
  - a set of "*generic_service class*"C^{gsr}
  - a set of "*specific_service class*"C^{ssr}
  - a set of "*service_category attribute*" A^{sca}.
- one "*Information Package*" pa^{Inf},
  - a set of "*business_object class*" C^{bob} which are linked with two classes: "*format class*" C^{for} and "*semantic_definition class*" C^{sde}.
- one "*Process Package*" pa^{pro} which contains:
- two sub-packages "*basic activity*" pa^{bac} to describe the basic synchronization activities of the process and "*structured activity*" pa^{sac} to describe activities which control the flow of the process,
  - a set of "*invoke class*" C^{inv}
  - a set of "*receive class*" C^{rec}
  - a set of "*reply class*" C^{rep}

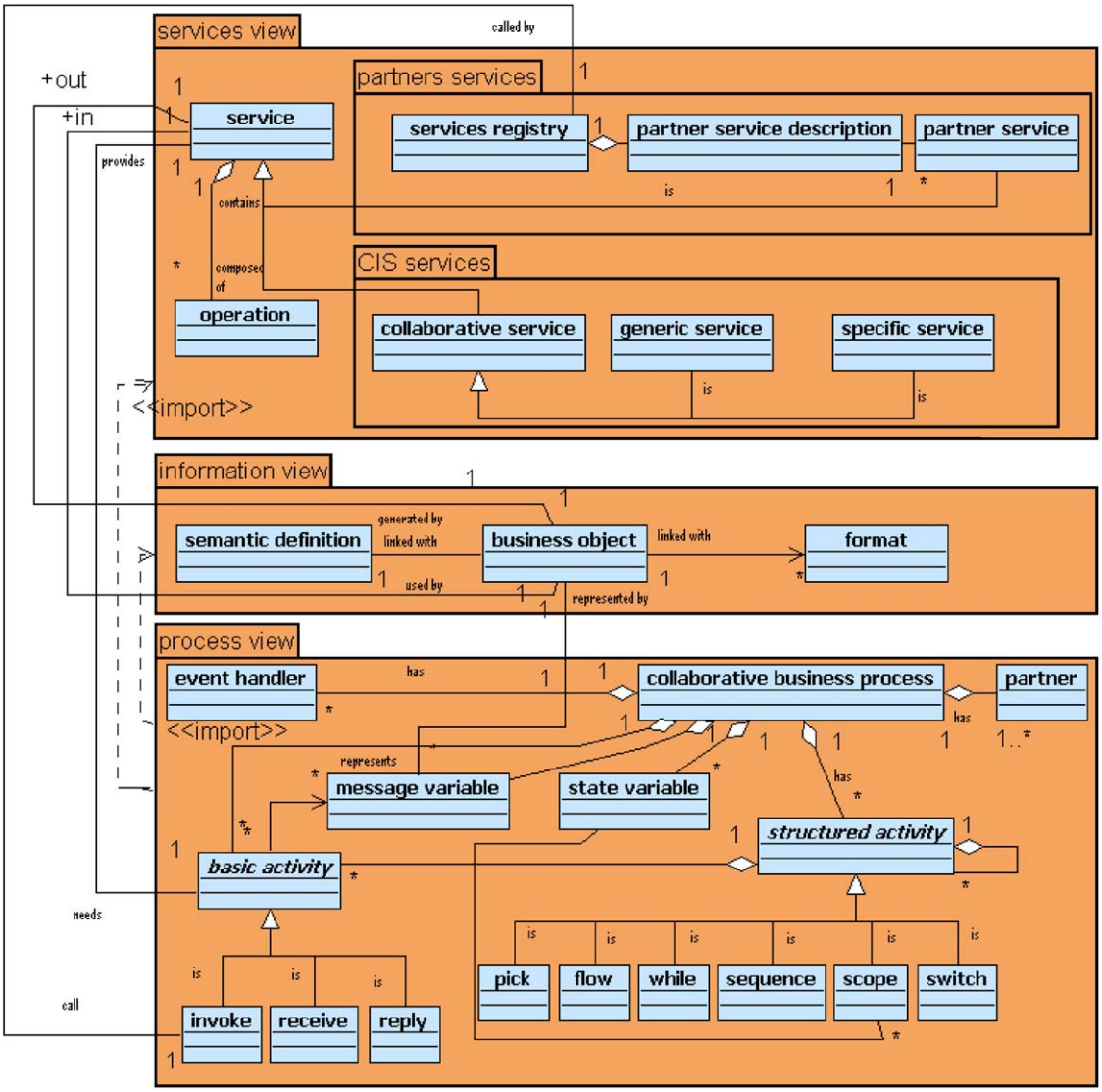

Fig. 5. Collaborative process meta-model.

- A set of "*pick class*" C^{pik}
- A set of "*flow class*" C^{flo}
- A set of "*while class*" C^{whi}
- a set of "*sequence class*" C^{seq}
- A set of "*scope class*" C^{sco}
- A set of "*switch class*" C^{swi}
- A set of "*message variable class*" C^{mva}
- A set of "*partner class*" C^{par}
- A set of "*event handler class*" C^{eha}
- A set of "*association*" *Asso*:*x.Asso.y* , *x, y* are classes of the collaborative SOA model.

3.5. Transformation rules

Transformation rules are classified into two categories:

- *basic generation rules* are used at first to create elements of the target model. Most of these rules are defined by a direct mapping between meta-model elements and
- *binding rules* are then applied to generate the links between the elements resulting from the previous phase. Existing relations in the source model are transformed into relations in the target model.

3.5.1. Preliminary assumptions

The rules we present in the following section are made under a set of assumptions that we show here. The CIMOSA enterprise modelling methodology presents for the majority of rules the basis of deduction:

- A functional part of an organization (or network of organizations) which composed of a set of activities is strongly connected to a resources part of an organization (or network of organizations). An activity needs (or is based on) an applicative resource to operate.

- Every exchange between two partners of the collaboration can be characterized by the description of a business object (structure, semantic definition, etc.).
- For each functional part (a set of activities), there is an organizational part which is responsible for.

Other rules are simply inspired on theone hand from our expertise, in BPM and information system domains and on the other hand from the expertise of our industrial partners (EBMWebsourcing) in the domain of the design of collaborative solutions:

3.5.2. Basic generation rules

Figs. 6–8 show a graphical representation of the set of rules that are applied during transformation to generate the three views of the SOA model. Circles located in the middle of two class diagrams represent the rules. The class diagrams are sub-graphs, which are parts of the presented meta-models. On the left part of each Figure is the sub-graph of the source meta-model, and on the right part is the sub-graph of the target meta-model. The rules have to be interpreted in the following manner: "When an object is identified in the collaborative process model (belongs to the left side sub-graph linked to the rule), it will be transformed into an object instantiated from the class on right side of the figure. We mean that it will become an object in the collaborative information system of the network."

Based on Definitions 1 and 2, the following presents a formal representation of these rules. We consider the function *gen* where $x \xrightarrow{gen} y$, x is a subset of the collaborative process meta-model (Definition 1) and y is a subset of the SOA meta-model (Definition 2). This function must be interpreted as follow: "for every x, detected in the source model, y elements are generated in the target model". Fig. 6 shows the rules needed to generate UML classes of the services view from the collaborative process.

- *Rs1 rule*:

  $\forall x \in T^{CIS}, \quad x \xrightarrow{gen} y / y \in \{C^{gsr} \cup C^{ssr}\}.$

  For each *CIS task* in the collaborative process model a *CIS service is generated*, either specific or generic. An annotation (*generic*) is added to the process model task to make it easier to identify generic CIS services;
- *Rs2 rule*:

  $\forall x \in L^{CIS}, \quad x \xrightarrow{gen} y / y \in A^{sca}.$

  The *CIS lane* of the collaborative process corresponds to an attribute of the *collaborative service* class which defines the organization of services of the CIS according to different categories;

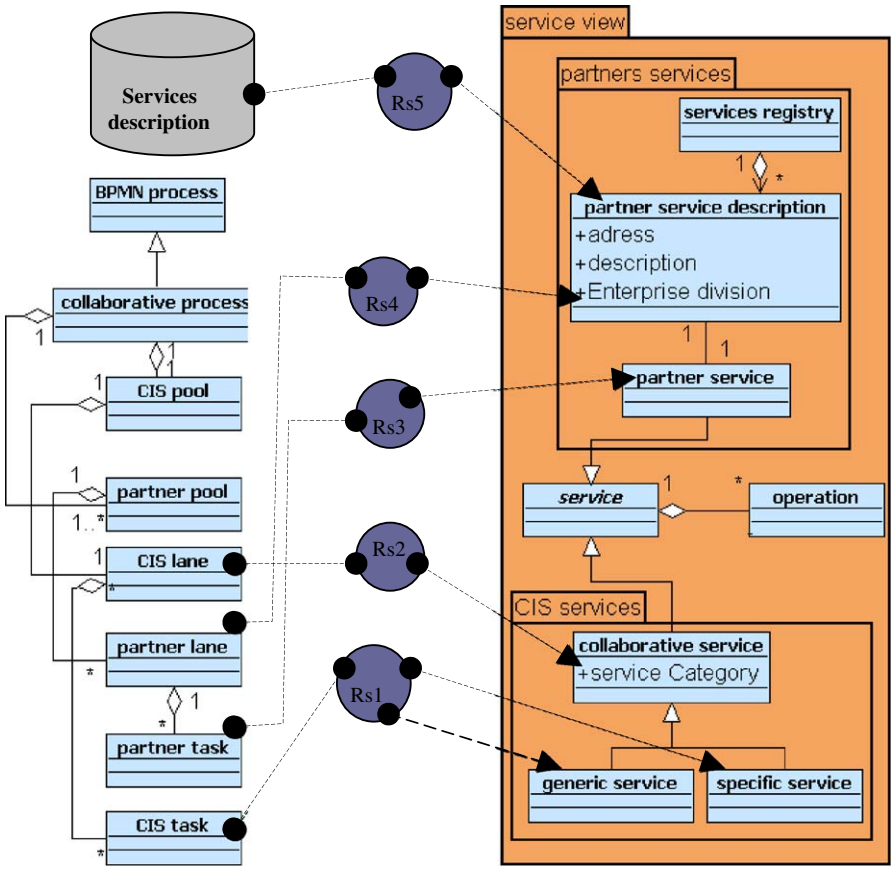

Fig. 6. Transformation rules for generating the services view.

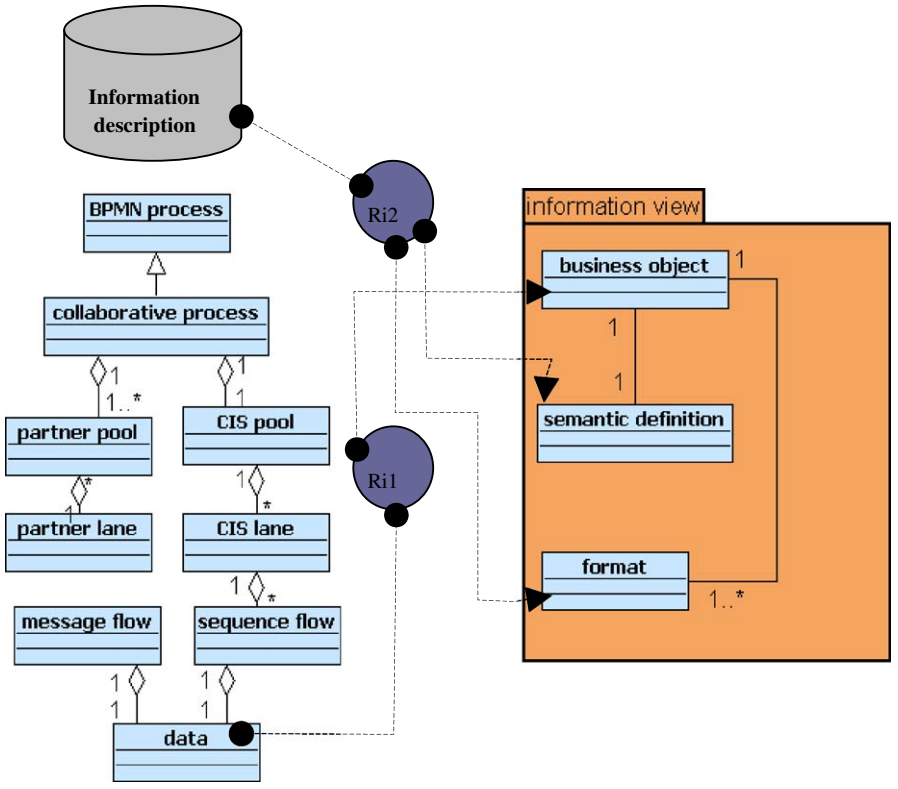

Fig. 7. Transformation rules for generating the information view.

- *Rs*3*rule*:

  $\forall x \in T^{par}, \quad x \xrightarrow{gen} y / y \in C^{psr}.$

  This rule is similar to *Rs*1 but concerns the deduction of a *partner service* from a *partner task*;
- *Rs*4 *rule*:

  $\forall x \in L^{par}, \quad x \xrightarrow{gen} y / y \in A^{edi}.$

  This rule expresses the organization of the partners' services. An attribute (enterprise division) shows the partner division to which the service belongs;
- *Rs*5 *rule is not a rule to implement but it* shows the need for additional knowledge to obtain a complete and useful view of services. This additional knowledge concerns a description of service implementations (address, access protocols, etc.).

Following the same logic, Fig. 7 introduces two transformation rules applied to the information view. Transformation rules provide syntactic indications that help to create business objects:

- *Ri*1 rule:

  $\forall x \in d, \quad d \xrightarrow{gen} (y, z, w) / y \in C^{bob}, \ z \in C^{for}, \ w \in C^{sde}.$

  This rule concerns the *data* element that is associated with the *message flow* element. The deduced *business object* elements refer to the messages (*data*) exchanged between partners in the collaboration;
- *Ri*2 *rule is not a rule to implement but it* shows the limits of the BPMN model in describing exchanged business objects (invoice, order, etc.). As previously stated, the transformation is not sufficiently developed in this view. Additional knowledge is needed to describe structure of information.

In contrast, Fig. 8 is the most developed part of the transformation procedure, with nine rules. Some of the rules in Fig. 8 are adaptations of recommendations provided by BPMI (BPMI, 04) where they address the problem of BPMN graph conversion to BPEL, well-defined XML phrases, and the work on BPMN-BPEL mapping by Ouyang et al., (2006):

- *Rp*1 *rule*:

  $\forall x \in L^{par}, \quad x \xrightarrow{gen} y / y \in C^{par}.$

  This rule concerns the deduction of *partner* element that is important to specify the holder of one activity from BPMN *partner lane* element;
- *Rp*2 *rule*:

  $\forall x \in d, \quad x \xrightarrow{gen} y / y \in C^{mva}.$

  This rule represents one *business object* of the collaborative process using specific *message variables* in the process view;
- *Rp*3 *rule*:

  $\forall x \in Sf, \quad x \xrightarrow{gen} y / y \in C^{seq}.$

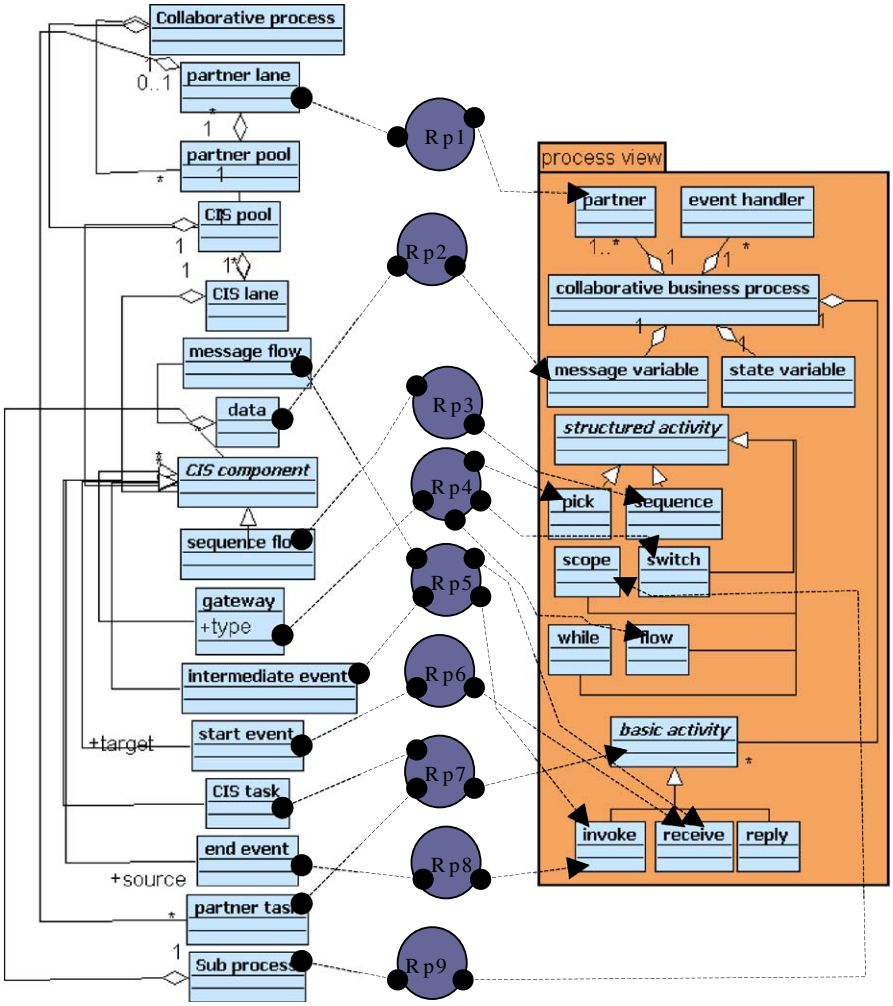

Fig. 8. Transformation rules for generating the process view.

This rule concerns the deduction of *sequence* elements (logical sequence of basic activities) from BPMN *sequence flow*;

- *Rp4 rule*:

$$\forall x \in G^{p}, \quad x \xrightarrow{gen} y / y \in C^{flo}$$

$$\forall x \in G^{dbi}, \quad x \xrightarrow{gen} y / y \in C^{flo} \in zC^{swi}$$

$$\forall x \in G^{ebe}, \quad x \xrightarrow{gen} y / y \in C^{pik}$$

$$\forall x \in G^{dbe}, \quad x \xrightarrow{gen} y / y \in C^{swi}.$$

This rule allows the transformation of BPMN gateways into different BPEL elements (*pick*, *flow* and *switch*) depending on the type of gateway:

- if it is a *parallel gateway*, a *flow* class will be generated to express a parallel execution of activities;
- if it is a *data-based inclusive gateway*, a *flow* class will be generated, associated with a *switch* class for each set of activities linked to the gateway;
- if it is an *event-based exclusive gateway*, a *pick* class will be generated to express that an event must be produced to continue the execution of the process;
- if it is a *data-based exclusive gateway*, a *switch* class will be generated to express that the continuation of the execution of the process depends on the value of a variable;

- *Rp5 rule*:

$$\forall x \in E^{i}, \quad x \in MfIn, \quad x \xrightarrow{gen} y / y \in C^{rec}$$

$$\forall x \in E^{i}, \quad x \in MfOut, \quad x \xrightarrow{gen} y / y \in C^{inv}.$$

- *This rule* concerns the transformation of *intermediate events* into *basic activities*. This transformation depends on the type of the message flow connected to the event;
  - if it is an inbound *message flow*, a *receive* class will be generated because a new message is received;
  - if it is a outbound *message flow*, an *invoke* class will be generated because a new message is sent.

- *Rp*6 *rule*:

  $\forall x \in E^s, \quad x \xrightarrow{gen} y/y \in C^{rec}.$

  This rule concerns the transformation of *start events* into *receive* classes. The process receives a message that produces a *start event* to start the process;
- *Rp*7 *rule*:

  $\forall x \in T^{par} \cup T^{cis}$:

  $x \in MfIN \wedge x \notin MfOUT, \quad x \xrightarrow{gen} y/y \in C^{inv}$

  $x \in MfOUT \wedge x \notin MfIN, \quad x \xrightarrow{gen} y/y \in C^{rec}$

  $x \in MfIN \wedge x \notin MfOUT, \quad x \xrightarrow{gen} y/y \in C^{rep}.$

  This rule shows that BPMN tasks will be transformed into *basic activities.* Depending on the type of the BPMN class, a *receive*, *reply* or *invoke* activity is generated. The type of the BPMN task can be defined according to inbound and outbound *message flows* connected to the task;
- *Rp*8 *rule*:

  $\forall x \in E^e, \quad x \xrightarrow{gen} y/y \in C^{inv}.$

  This rule concerns the transformation of *end events* into *invoke* classes. The process sends a message that signals its end;
- *Rp*9 *rule*:

  $\forall x \in Sp, \quad x \xrightarrow{gen} y/y \in C^{sco}.$

- *This rule* shows that a BPMN sub-process must be transformed into a *scope* element. This element defines a limited part of the execution of the process (activities, gateways, message variables, etc.).

3.5.3. Binding rules

Binding rules can be used to build interactions between the generated elements of the CIS model (results from the application of the first category of rules). These links could be inside one CIS package or between two different packages (dependence). The goal is to define, in the target model, the relations needed in accordance with the existing relations in the source model. The relations are of type, *association.* We define the function $Y = Equivalent\ (X, pa)$, where X belongs to the BPMN model and Y is the result of the transformation rules defined, belongs to the information system model. *pa* is the target package of the generated element (services, information, process).[5]

Three binding rules, *Rb*1 to *Rb*3, are given:

- *Rb*1 *rule* (sequence ordering):

  $x \in SfIN, \quad y \in SfOUT$

  $r \in Sf, x.r.y$

  $x' = Equivalent\ (x, process)$

  $y' = Equivalent\ (y, process)$

  $r' = Equivalent\ (r, process)(r' \in C^{seq})$

  $r \xrightarrow{gen} (from, to)/from,\ to \in Ass^2, \quad r'.from.x',\ r'.to.y'$

  a *sequence* element issued from rule Rp3 is associated with two *basic activities* into the same process package;
- *Rb*2 *rule* (information processing): we define the function $y = isManipulatedBy(x)$ where y is a task and x is a business object, manipulated (sended or received) by x (i.e. there is a message flow outgoing or ingoing x)

  $y \in \{T^{par} \cup T^{CIS}\}, \quad x \in d,\ y = isManipulatedBy\ (x)$

  $y' = Equivalent\ (y, services)(y' \in C^{par} \cup C^{gsr} \cup C^{ssr})$

  $x' = Equivalent\ (x, information), \quad (x' \in C^{bob})$

  $d \xrightarrow{gen} use/use \in Asso, \quad x'.use.y'$

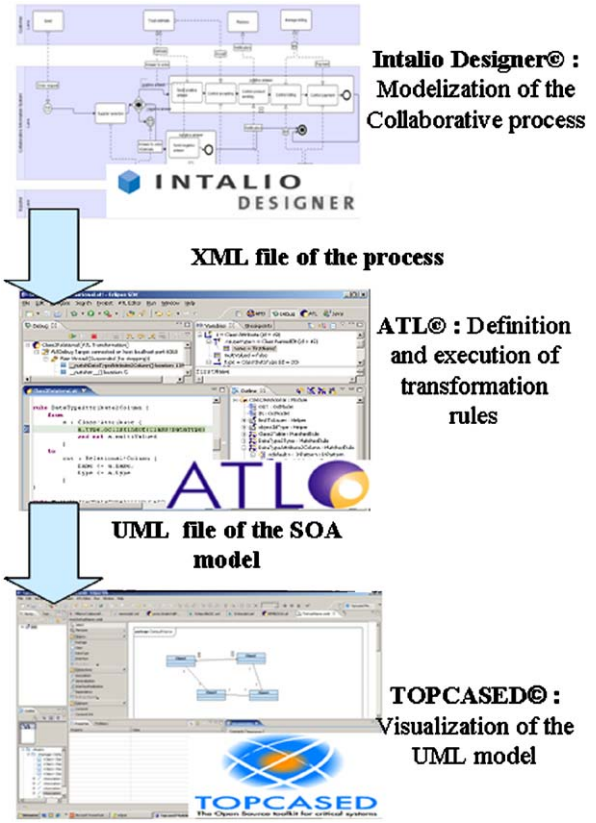

Fig. 9. Technical architecture of the developed prototype.

[5] One BPMN element can be mapped onto one different element of different package.

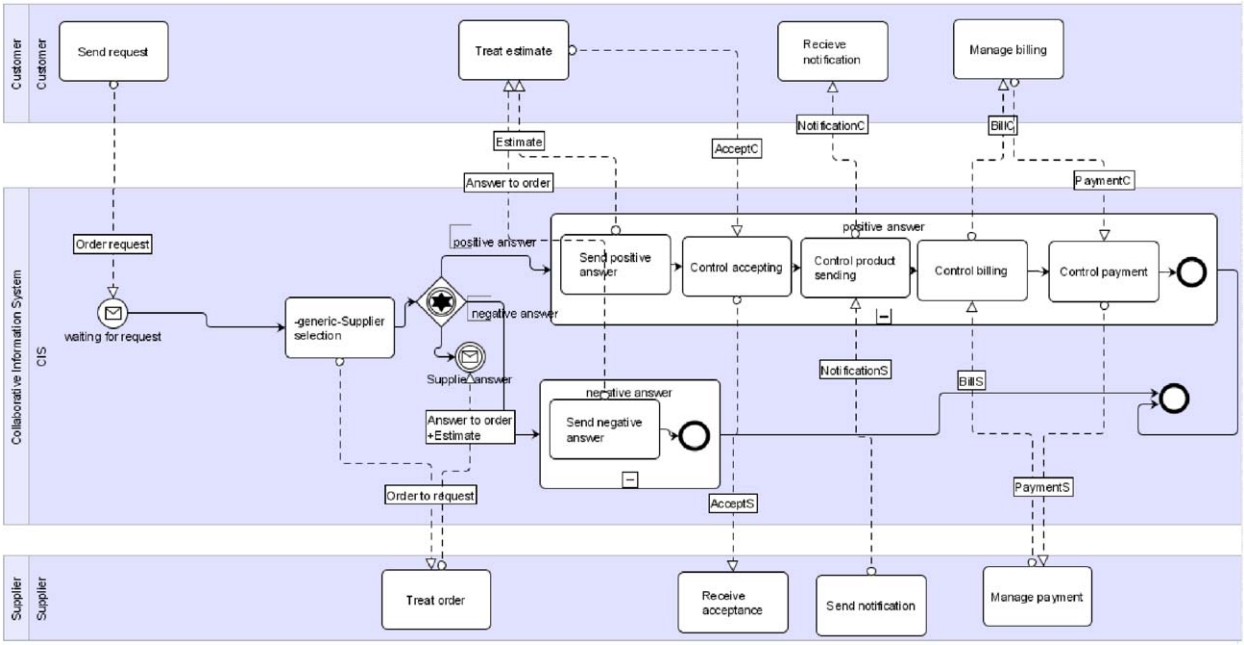

Fig. 10. Example of a collaborative process.

a *service* from a service package is related to a *business object* from the information package;

- *Rb3 rule* (service identification):

$$x \in \{T^{par} \cup T^{CIS}\}$$

$$x' = Equivalent\ (x, process)\ (x' \in C^{inv} \cup C^{rec} \cup C^{rep})$$

$$x'' = Equivalent\ (x, services)\ (y' \in C^{ser})$$

$$x \xrightarrow{gen} call/call \in Asso, \quad x'.call.x''$$

a *basic activity* from the process package is linked to a *service* from the service package.

4. Prototype development

A prototype transformation tool has been developed to implement our proposition. It is based on three open source tools that run on the IDE *Eclipse*© platform. *Intalio designer*© is a BPM tool that helps users to specify a BPMN model. The *Atlas transformation language* (ATL)© (Jouault and Kurtev, 2006) can use a process model in XML format coming from Intalio designer© in input, and produces the UML model in output (applying the transformation rules mentioned into this paper). ATL is QVT-compatible. Query, view and transformation (QVT) is a specialized language that is being developed under the guidance of the OMG. One of the purposes of this language is to allow transformations between models. The ATL tool is the cornerstone of our transformation system. The *TOP-CASED*© tool is a computer-aided software environment that can create a graphical representation of the UML model. Fig. 9 shows the technical architecture of the prototype.

Meta-models are created using the Eclipse Modelling Framework (EMF)[6] which allows to create an *ecore* file (*.ecore*) for each meta-model. ATL can deal directly with *ecore* files as input and output of the transformations.

The formalized rules presented in the previous section are the cornerstone of the deduction of the ATL code needed to perform the models transformations. As a simple example of the ATL code,

*The Rs*3 *rule*: $\forall x \in T^{par}$, $x \xrightarrow{gen} y/y \in C^{psr}$ corresponds to this ATL code:

```
rule generatePartnerservices
{
    from
        a: BPMN!PartnerTask
        to
        service: UML2!Class
    (
    name <- a.name
    )
}
```

The *from* and *to* parts of the ATL rule correspond respectively to the left and the right sides of the formalized rule *Rs*3. The whole ATL code is more complicated than the example presented, especially concerning the imperative rules (not declarative) which are not based on a direct mapping between elements.

[6] www.eclipse.org/emf.

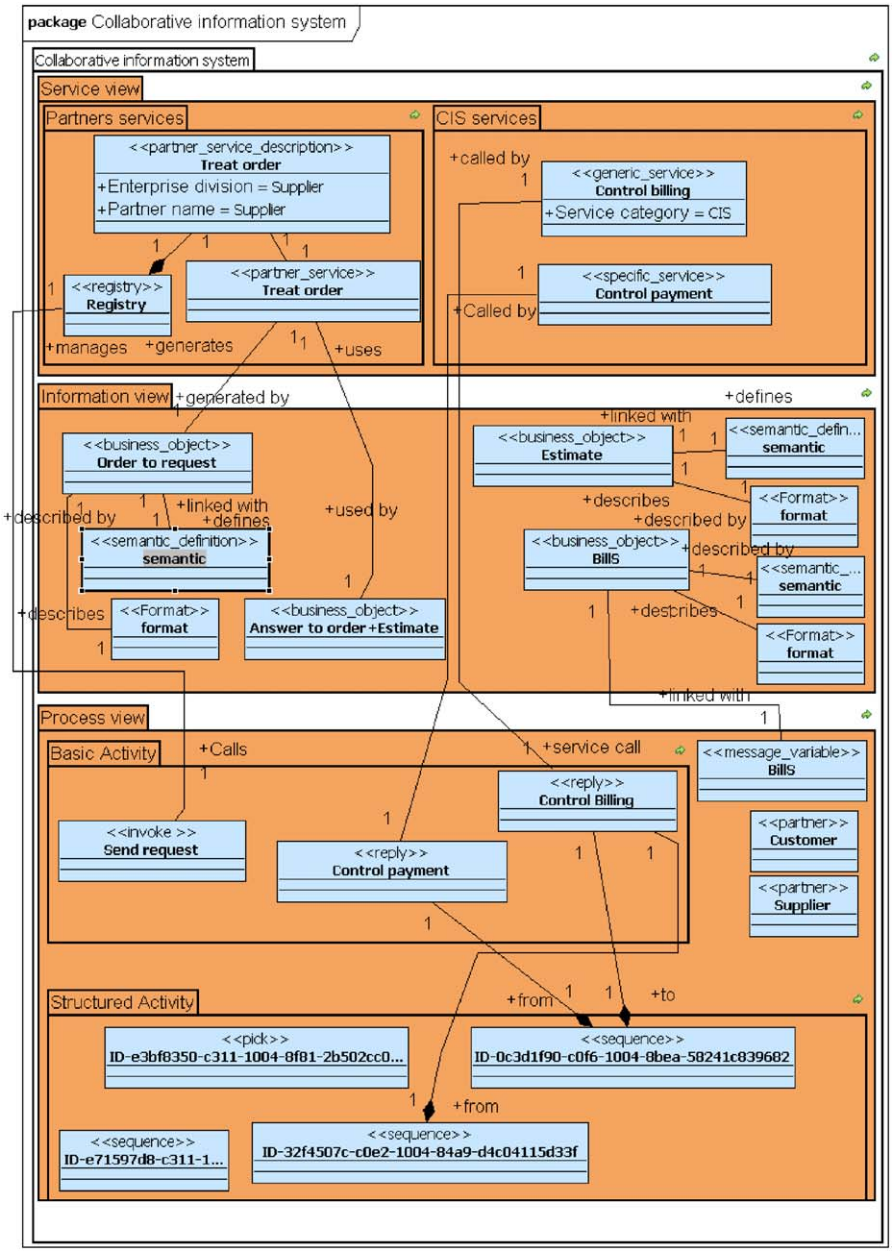

Fig. 11. Result of the transformation using the developed prototype.

5. Example of transformation

A series of simple case studies have been defined and examined in order to begin the validation of the approach. A simple example of a collaborative process is proposed in Fig. 10.

The collaboration takes place between a customer and a set of suppliers for a trading transaction. The customer sends an order to the mediator (CIS pool). The CIS must find a supplier corresponding to the customer order characteristics. The contacted supplier has to analyse the order and to answer the customer. If the answer is positive, the supplier has to inform the customer when the product is ready for dispatch. Then, the two partners have to perform payment and billing operations.

Fig. 11 shows the result of the transformation of the collaborative process of Fig. 10 using the developed prototype.[7]

The model obtained is useful for managing message, service and process definitions in the CIS. In the *services view package*, a registry of services is linked with all partners' services involved in the collaboration.

[7] For reasons of clarity in the model, we show only a few relevant UML classes.

"treat order" task is mapped into "treat order" service which is linked to the registry. The CIS services subpackage contains all collaborative services managed by the CIS. "control billing" and "control payment" services are deduced from the BPMN model. In the *information view package*, business objects that refer to supplier and customer are defined for each message exchanged in the collaboration. "order to request" and "estimate" are examples of business objects but without details about their structure. In the *process view package*, synchronizations between different partners' activities are established. The "event-based gateway" of the BPMN process is mapped into a "pick" element. BPMN tasks are mapped into activities according to their type. The strong point of this ATL-generated model is that using UML associations it clearly shows, on the one hand, links between messages and services and, on the other hand, links between activities and services. "Control payment" activity needs the service with the same name to run. The "treat order" service deals with "order to request" business object. This kind of knowledge is crucial in SOA context.

However, the model obtained is incomplete. For example, we do not have information about the specific format of the business objects. Therefore, partners must provide this information. This information is crucial to allow partners to exchange messages with a structure that these partners can understand.

6. Conclusion and prospects

The presented work intends to enrich frameworks which define interoperability at the three levels (CIM, PIM, PSM) with the definition and the formalization of transformation rules between models that belong to CIM and PIM levels, under a set of assumptions, inspired by the actual practices in the development of systems integration solutions.

Our MDA methodology bridges the gap between the business analyst level (BPMN collaborative process model) and the IT developer level (collaborative SOA model). The principal limitation of our approach is the difficulty to semantically prove the correctness of the rules and its specification. The SOA model, obtained should be used as an intermediate step when the final objective is to obtain ESB artefacts (XSD7, BPEL, WSDL8, etc.), needed to configure an ESB solution according to a given BPMN collaborative process. EBM WebSourcing (our industrial partner) currently develops an ESB tool inside the OW2 open source community; the project is called PETALS (see http://petals.objectweb.org).

We are aware that it is relatively uncommon to have networks of organizations that are able to design a collaborative process for their projected shared activities. In (Rajsiri et al., 2007), we study the contribution of a knowledge-based methodology to help in the process model design using ontology based approach;

Collaborative processes may dynamically evolve and the collaboration may also change with time. CIS supporting the partnership should mirror such change. Lastly, in order to improve the solution, we are also involved in the *ISyCri* Project (French project: ANR/CSOSG2006). The problem to solve concerns the development of interoperability between actors in a crisis context.

References

Athena Consortium, 2004. Public document: ATHENA General description v10, 〈http://www.athena-ip.org/〉.

Benguria, G., Larrucea, X., Elveseater, B., Neple, T., Beardsmore, A., Friess, M., 2006. Platform Independent Model for Service Oriented Architectures. Enterprise Interoperability: New Challenges and Approaches. Springer, Berlin, pp. 23–32, ISBN-10: 1846287138.

Business Process Management Initiative (BPMI), 2004. Business Process Modeling Notation (BPMN), Version 1.0.

Chen, D., Doumeingts, G., 2003. European initiatives to develop interoperability of enterprise applications, basic concepts, framework and roadmap. Annual Reviews in Control 27, 153–162.

Doumeingts, G., Vallespir, B., Chen, D., 1998. Decisional modelling GRAI grid. In: Bernus, P., Mertins, K., Schmidt, G. (Eds.), International Handbook on Information Systems. Springer, Berlin.

e-Gov, 2005. e-Government Unit, e-Government Interoperability framework, Version 6.1, 2005.

EIF, 2004. European Interoperability Framework, White Paper, Brussels, 〈http://www.comptia.org〉.

Grangel Seguer, R., Ben Salem, R., Bourey, J.-P., Daclin, N., Ducq, Y., 2007. Transforming GRAI Extended Actigrams into UML Activity Diagrams: A First Step to Model Driven Interoperability, Enterprise Interoperability: New Challenges and Approaches II. Springer, Berlin, pp. 447–458, ISBN: 978-1-84628-857-9.

IDEAS, 2003. A Gap Analysis–Required activities in Research, Technology and Standardisation to Close the RTS Gap—Roadmaps and Recommendations on RTS Activites, IDEAS Deliverables.

IEC, 2005. IEC TC 65/290/DC, Common automation device. Device Profile Guideline, TC65: Industrial Process Measurement and Control, IEC, Geneva, Switzerland.

IEEE, 1990. IEEE: Standard Computer Dictionary—A Compilation of IEEE Standard Computer Glossaries.

INTEROP, 2007. Interoperability Research for Networked Enterprises Applications and Software NoE (IST-2003-508011), 〈http://www.interop-noe.org〉.

ISO/DIS 19440.2., 2005. Enterprise Integration—Constructs for Enterprise Modeling. ISO, Genève.

Jouault, F., Kurtev, I., 2006. On the Architectural Alignment of ATL and QVT. In: Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 06), Chapter Model transformation (MT 2006), ACM Press, Dijon, France, pp. 1188–1195.

Kosanke K., 2005. ISO standards for interoperability: a comparison. In: Proceedings of the First International Conference on Interoperability of Enterprise Software and Applications INTEROP-ESA'05 (IFIP/ACM SIGAPP INTEROP-ESA'2005). Springer, Berlin, pp. 55–64, ISBN 1-84628-151-2.

Maamar, Z., Kouadri Mostefaoui, S,. Mahmoud, Q.H., 2005. On personalizing web services using context. International Journal of E-Business Research 1(3), Special Issue on E-Services, The Idea Group Inc.

Missikoff, M., Taglino, F., 2006. Ontologies for interoperability: a systematic overview. In: Lecture in ECI Workshop, Paris.

OASIS, 2003. Technical Committee: OASIS Web Services Business Process Execution Language. 〈http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel〉.

OMG, 2003a. MDA Guide Version 1.0.1. Object Management Group. Document number: omg/2003-06-01 edn.

OMG, 2003b. OMG Unified Modeling Language Specification, version 1.5. Object Management Group. formal/03-03-01 edn.

Ouyang, C., Van Der Aalst, W., Dumas, M., Hofstede, A., 2006. Translating BPMN to BPEL. Technical Report, BPM group of Queensland University of Technology Brisbane (QUTB).

Rajsiri, V., Lorré, J.-P., Bénaben, F., Pingaud H., 2007. Cartography for Designing Collaborative Process, Enterprise Interoperability: New challenges and approaches II, Springer, Berlin, ISBN: 978-1-84628-857-9.

Touzi, J., 2007. Aide à la conception de système d'information collaboratif, support de l'interopérabilité des entreprises. Ph.D. Thesis, Ecole des Mines d'Albi Carmaux, France.

Vernadat, F., 2006. Interoperable enterprise systems: architecture and methods. In: Plenary Lecture, IFAC/INCOM Conference, Saint-Etienne, France.