A MDA-based Development Process for Collaborative Business Processes

Pablo David Villarreal¹, Enrique Salomone¹², Omar Chiotti¹²

¹CIDISI, Universidad Tecnológica Nacional - Facultad Regional Santa Fe, Lavaisse 610, 3000, Santa Fe, Argentina
(pvillarr@frsf.utn.edu.ar)
²INGAR-CONICET, Avellaneda 3657, 3000, Santa Fe, Argentina
({salomone,chiotti}@ceride.gov.ar)

Abstract. The modeling and specification of collaborative business processes is a key issue in order to enterprises can implement Business-to-Business collaborations with their partners. These processes have to be defined first in a business level using a technology-independent language and then in a technological level using a Business-to-Business standard. Both definitions must have a mutual correspondence. To support the above issues, the Model-Driven Architecture (MDA) Initiative has been identified as a key enabler to build a model-driven development method. In this paper, we describe a MDA approach for collaborative business processes. We present the phases, artifacts (models and code), languages, mappings and transformation techniques that make up the development process of this MDA approach. We identify the issues to be taken into account in a MDA approach for collaborative processes, in order to guarantee that the generated B2B specifications fulfill the collaborative processes agreed between the partners in a business level.

1 Introduction

To compete in the current global markets, enterprises are focusing on the setting up of Business-to-Business (B2B) collaborations with their partners. A B2B collaboration implies the integration of enterprises in two levels: a business level and a technological level [22, 23]. In order to accomplish inter-enterprise integration at both levels, one of the main challenges is the modeling and specification of collaborative business processes. Through these processes, partners undertake to jointly carry out decisions to achieve common goals, coordinate their actions and exchange information from a common global point of view.

In the business level, partners have to focus on the definition of collaborative business processes, which describe the explicit behavior of the collaboration along with the partners’ responsibilities. In this level, collaborative processes have to be defined in an independent way of the implementation technology, because business analysts and system designers do not want to deal with the technical details.

In the technological level, enterprises have to focus on the implementation of the B2B information system that supports the collaborative processes. This system is
made up of autonomous, heterogeneous and distributed components owned by enterprises to support the joint execution of collaborative processes. Interoperability between these components can be achieved by using B2B protocols [3]. Two important interoperability aspects are supported by these protocols: the specification of the collaborative processes and the specification of the partners’ interfaces that compose the B2B information system, both based on a technological-specific standard language. Currently, there are two main technologies proposed to specify B2B protocols [3]: standards based on Web Services Composition, such as BPEL (Business Process Execution Language) [4] and WS-CDL (Web Services Choreography Description Language) [14]; and standards based on Business Transactions, such as ebXML [20].

Thus, collaborative processes must be defined first in a business level using a technology-independent language and then in a technological level using a B2B standard. However, the development of these B2B solutions is complex and costly [1]. In addition, both definitions must have a mutual correspondence [1, 3]. Another complexity is B2B standards provide or are based on different languages to specify collaborative processes and partners’ interfaces, but these standards do not provide techniques to support the transformation or verification of correctness between the collaborative process specifications and their corresponding partners’ interfaces. Hence, it is necessary a mechanism that also ensures, in the technological level, consistence between the collaborative process specifications and the partners’ interfaces specifications.

To accomplish the above issues, the Model-Driven Architecture (MDA) Initiative [17] has been identified as a key enabler to build a model-driven development method for collaborative processes [21, 23]. In this way, we propose a MDA approach for collaborative processes, which supports: the design of collaborative processes independent of the idiosyncrasies of particular B2B standards; and the automatic generation of specifications based on a B2B standard from conceptual collaborative process models. The main benefits of this approach are:

- Increase of the abstraction level. The main artifacts of the development are the technology-independent collaborative process models.
- Reduction of development time and costs because solutions are generated automatically from conceptual models of collaborative processes.
- Guarantee of consistency in the generated technological solutions because they are generated automatically from a well-defined transformation procedure. On one hand, they are consistent with the processes defined in the business level. On the other hand, the specifications of processes and their corresponding partners’ interfaces specifications are also consistent.
- Independence of the collaborative process models from the B2B standards, which increases the reuse of these models.

To support the design of collaborative processes we have proposed the UML Profile for Collaborative Business Processes based on Interaction Protocols (UP-ColBP) [21, 22, 23]. This is a modeling language defined as UML2 Profile. In addition, in previous works, we have described the generation of technological solutions based on the current different standards by using a MDA approach: ebXML [22], BPEL [24] and WS-CDL [25]. In this way, we had applied a MDA approach to generate B2B specifications and we have also validated the UP-ColBP language against
standard languages. The conclusions were most of the elements of the B2B standards can be derived from UP-ColBPIP models.

The purpose of this paper is to describe a complete MDA-based development process for the modeling and specification of the collaborative processes and the partners' interfaces that make up the B2B information system. We present the artifacts (models and code), the languages and techniques to build and transform these artifacts, and the phases and activities of the MDA approach for collaborative processes. We also identify the issues to be considered in each phase of this MDA approach and the patterns of transformations to be applied in order to guarantee the generated B2B specifications fulfill the collaborative processes agreed by the partners in a business level.

This paper is organized as follows. Section 2 describes the phases and the generic pattern of transformations of the MDA approach we propose for collaborative processes. For each phase we describe: the artifacts to be used and built, the techniques to be used and the transformations (mappings) to be defined and applied. Section 3 describes the transformation model tool we have developed to support the transformations required in this MDA approach. Section 4 discusses related work. Section 5 presents conclusions and outlines future research directions.

# 2 MDA Approach for Collaborative Business Processes

The OMG’s MDA Initiative [17] proposes a conceptual framework along with a set of standards (UML, MOF, XMI, etc.) to build model-driven development methods. In MDA, the generic development process consists of: defining platform-independent models (PIMs), selecting platform-specific models (PSMs) and executing transformations that generate PSMs from PIMs, and finally generating Code that can be executed from the PSMs. A platform makes reference to the technology to be used to implement the system. In this way, MDA provides the guidelines and the general artifacts that a model-driven development method has to contain, but it does not provide the concrete development process, the artifacts to be built and the techniques to be used for a particular domain. Furthermore, although UML can be considered as a generic standard modeling language that can also be used to model organizational aspects, it is important to use languages that are more suitable and closer to the application domain.

Therefore, to support the development of collaborative processes and B2B information systems, it is necessary to define a MDA-based development process with the phases, activities, artifacts and techniques required for this application domain.

We propose a MDA approach for collaborative processes in which the development process consists of three phases (Figure 1):
1. Analysis and design of collaborative processes
2. Verification of collaborative processes
3. Implementation of collaborative processes

The artifacts to be produced in the first phase are the technology-independent collaborative process models (Figure 1), which are the inputs for the next phases. In the second phase, the output artifacts are the specifications of collaborative processes in a formal language, in such a way that they can be verified. In this case, according to the
results of the verification, it is possible to go back to the first phase in order to correct the processes. Several cycles of analysis/design and verification can occur. Finally, once the collaborative processes were agreed, the third phase consists of selecting the technology of implementation (i.e. the B2B standard to be used) and generating the B2B specifications that fulfill the collaborative processes defined in the first phase.

![Phases of the MDA approach for collaborative business processes](image)

Fig. 1. Phases of the MDA approach for collaborative business processes

To produce the output artifacts of the second and third phases, a generic pattern of transformations is applied according to the principles of MDA, which is shown in Figure 2. This figure indicates the artifacts (components) and the techniques used to build them. The input artifacts are the technology-independent collaborative process models, which are defined in the first phase. The intermediate artifacts are the technology-specific models and the output artifacts are the XML-based specifications.

Two types of transformations have to be defined and executed: transformations of technology-independent models into technology-specific models, and transformations of technology-specific models into XML-based specifications. The first one requires the use of a model transformation tool that enables the definition and execution of these transformations. In addition, it requires the building of the metamodels corresponding to the technology-specific models to be generated. These metamodels can be derived from the XML schemas provided by the technology-specific languages. In this way, the metamodel of a language corresponds to its XML schema; and the models correspond to the XML documents that contain the XML-based specifications. Although the XML documents may be generated directly from the technology-independent models, this intermediate representation allows a more modular and maintainable process transformation and it is in line with the principles of MDA.

The second type of transformations supports the generation of XML documents corresponding to the B2B specifications, according to the technology-specific language selected. These transformations are almost direct through the applications of XML production rules that convert a UML class model into a XML version.

In the next subsections we describe the different phases of the MDA approach for collaborative processes. Subsections 2.2 and 2.3 describe the application of the generic pattern of transformations to implement the transformations required in the second and third phases. Section 3 describes the model transformation tool we have developed and applied to support the definition and execution of these transformations.
2.1. Phase 1: Analysis and Design of Collaborative Processes

This phase is about the modeling of collaborative processes from a business perspective of the B2B collaboration. People involved in this phase are business analysts and system designers, who are the responsible of defining the business aspects of the B2B collaboration. This phase focuses on the analysis and design of the collaborative business processes. The analysis stage treats with the identification of the business requirements, i.e. details of the collaboration agreement, common business goals, etc., and the identification of the collaborative processes required to achieve the common business goals agreed by the partners. This stage is defined according with the collaborative business model that partners have agreed for the management of the B2B collaboration. The design stage treats with the detailed definition of the behavior of the identified collaborative processes, with the purpose of establishing explicitly the way in that partners will both coordinate their actions and exchange information. In addition, the design also implies the definition of the business interfaces of the partners, which can be derived from the collaborative processes.

The analysis and design of collaborative processes require the building of collaborative process models using concepts that are much less bound to the implementation technology and are much closer to the B2B collaborations domain. This is due to several reasons:

- Business aspects have to be considered in this phase.
- Business aspects are defined by business analysts or system designers who are more interested on the problem domain that in the technical details of implementation.
- People involved in this phase are not acquainted about the different B2B standards and technology-specific languages that can be used to implement the collaboration.

![Diagram of Generic Pattern of Transformations]

**Fig. 2.** Generic Pattern of Transformations
In this way, the analysis and design of collaborative processes imply the modeling of these processes in an independent way of the technology. To support this requirement we have proposed the UML Profile for Collaborative Business Processes based on Interaction Protocols (UP-ColBPIP). This modeling language has been defined as a UML Profile that extends the semantics of UML2 to model technology-independent collaborative processes. This language encourages a top-down approach and provides the conceptual elements to support the modeling of four views (Figure 3):

- **B2B Collaboration View.** This view captures the participants (partners and the roles they fulfill) and their communication relationships in order to provide an overview of the B2B collaboration. UP-ColBPIP extends the semantics of UML2 collaborations to represent B2B collaborations. Furthermore, this view defines the collaborative agreement parameters and the hierarchy of common business goals to be fulfilled by the partners in a B2B collaboration. They are represented extending the semantics of classes and objects of UML.

- **Collaborative Processes View,** which identifies the collaborative processes required to achieve the common business goals. To define this, UP-ColBPIP extends the semantics of use cases to represent collaborative processes as informal specifications of a set of actions performed by trading partners to achieve a common goal.

- **Interaction Protocols View,** which defines the explicit behavior of the collaborative processes through the use of interaction protocols. UP-ColBPIP extends the semantics of the UML2 interactions to model interaction protocols.

- **Business Interfaces View,** which offers a static view of the collaboration through the definition of the business interfaces of the roles performed by the partners. The business interfaces contains the business services (operations) that support the exchange of messages of the interaction protocols. To define this view, UP-ColBPIP extends the semantics of the composite structure and interfaces of UML2.

The first and second views correspond to the analysis stage. The last views correspond to the design stage. The output of this phase is a UP-ColBPIP model with the definition of the above four views, which will contain the description of the collaborative business processes to be carried out in a B2B collaboration.

Due to space limitations, we do not describe how the different views can be defined with the UP-ColBPIP language. More details about this language can be found in [21, 22, 23]. Following, we describe the most important view: Interaction Protocols.

In addition to support a methodological guide for the analysis and design of collaborative processes, the main contribution of this language is the use of interaction protocols to represent the behavior of collaborative processes, along with the application of the speech act theory [18]. In the B2B collaborations domain, an interaction protocol describes a high-level communication pattern through a choreography of business messages between partners playing different roles [23].

The purpose of modeling collaborative processes based on interaction protocols is to fulfill the identified requirements for B2B collaborations [22, 23]: enterprise autonomy, decentralization, peer-to-peer interactions, global view of the collaboration and support for negotiations.

In contrast to activity-oriented business process languages, the modeling of interaction protocols focus on the exchange of business messages representing interactions between partners. In this way, interaction protocols support the autonomy required by
the enterprises because their internal activities, services and decisions cannot be defined in interaction protocols and hence, they are kept hidden to its other partners.

Fig. 3. Analysis and Design Phase of the MDA Approach for Collaborative Processes

Modeling interaction protocols, the focus is on representing the global view of the interactions between the partners. The message choreography describes the global control flow of the peer-to-peer interactions between the partners, as well as the responsibilities of the roles they fulfill. This is the main different with activity-oriented business process languages such as UML Activity Diagrams or BPMN [16], which are more suitable to describe private processes or public (abstract) processes but from the viewpoint of one partner. Although they can also be used to define collaborative processes, to express the global view and the parallel processing of the parties, additional semantics is required for UML2 Activity diagrams and using BPMN, the definition of public processes of each partner is required in order to understand this global view.

In addition, B2B interactions cannot be restricted to mere information transfer [10] but they have to focus on the communicative actions between the parties. These communicative aspects are not also considered in many of the business process modeling languages. In UP-ColBPIP, communicative aspects of B2B interactions are represented in interaction protocols through the use of speech acts. In an interaction protocol, a business message has a speech act associated, which represents the intention that a partner has with respect to an exchanged business document through the message. Thus, decisions and commitments done by the partners can be known from the speech acts. This enables the definition of complex negotiations in collaborative processes.

As an example, we describe the interaction protocol Demand Forecast Request, which realizes a simplified process to manage collaborative demand forecasts. Figure
4 shows the sequence diagram of this protocol, in which partner “A” plays the supplier role and partner “B” plays the customer role. They are represented by lifelines.

![Sequence Diagram]

**Fig. 4. Demand Forecast Request Interaction Protocol**

The purpose of this protocol is the supplier and the customer agrees on the type of forecast demand to be defined. The protocol starts with the supplier, which requests a forecast demand, as it is indicated by the message *request(DemandForecast)*. The business document of this message contains the information about the products, periods, and the time horizon of the forecast. Then, the customer processes the received message with its contents and makes a decision. The customer can respond of two different forms, as it is expressed by the control flow segment with the Xor operator. This operator indicates that only one of the two paths can occur according to the condition defined in each path. One alternative is the customer accepts the supplier’s request and commits to carry out the demand forecast in the future. It is expressed by the message *agree(ResponseToForecastRequest)*. The business document of this message contains more information about the customer’s response. Then, the execution of the protocol continues. Another alternative is the customer sends a message *refuse(ResponseToForecastRequest)* to indicate that it rejects the type of forecast requested by the supplier. The business document of this message contains the reasons of the rejection. In this case, the protocol finishes with a failure, as it is indicated by the explicit termination event *failure*. This is only an indication that the protocol does not finish in a correct way from the point of view of the business logic. This failure event does not indicate a technical failure on the communication.

If the customer agrees on carry out the forecast, then the nested protocol *Collaborative Demand Forecast* is invoked in order to partners agree on a common demand forecast. After of the execution of this nested protocol, the protocol finishes. In this case the termination of the protocol is defined in an implicit way.
Furthermore, the business messages of the control flow segment have defined time constraints representing deadlines, which indicate these messages have to be sent before two days, after of the occurrence of the first message.

2.2. Phase 2: Verification of Collaborative Processes

Due to collaborative processes describe the behavior of the partners in the collaboration, as well as the form in which the partners’ systems will interact, the verification of functional requirements of these processes, such as the absence of deadlocks and livelocks, is required.

Currently, there are different proposals to verify business processes in technology-specific languages such as BPEL [8] and WS-CDL [9]. Thus, applying a model-driven approach it is possible to generate technological solutions and then verify the generated B2B specifications. However, the main disadvantage of this procedure is that the verification of processes is done in a late stage of the development, when the technological solution has been already defined. Hence, if results of the verification indicate modifications should be done to fix the B2B specifications, these modifications also should be propagated automatically to the collaborative process models. However, these modifications have to be done using the B2B standard languages selected instead of using a technology-independent collaborative process language.

The use of technology-independent models to verify system’s properties in an early stage of the development has been recognized as one of main aspects to be supported by a model-driven method [19], because in an early stage of the development is when business analysts and system designers make most of the fundamental decisions.

Therefore, we propose that the verification of the collaborative processes should be done in the second phase of the development process, after of the definition of the technology-independent collaborative process models. This implies that the verification should be applied on these models instead of the B2B specifications.

To achieve this, the generic pattern of transformations we have defined above can be applied for generating the specifications of the collaborative processes in a formal language. In our MDA approach we are using the Petri net formalism to express the semantics of the interaction protocols and enable the verification of their properties.

Figure 5 shows the pattern of transformations to be applied in this phase. The input is the UP-ColBPIP model that contains the defined collaborative processes based on interaction protocols. Model-to-Model Transformations are applied to generate Petri net models. These models are defined according to the language PNML (Petri Net Marking Language) [5], which is a XML-based standard language for Petri nets. Hence, the PNML metamodel has to be generated from the XML schema of PNML to enable the building of PNML models.

Finally, from these models, XML documents that contain the Petri nets specifications corresponding to the interaction protocols are generated. In this way, Petri Nets tools can then be used to read the generated Petri nets specifications and to carry out the verification procedures in order to determine if the collaborative processes based on interaction protocols are well-defined and fulfill the desirable properties. The properties to be verified on Petri nets representing protocols are those typical such as
the absence of deadlocks and livelocks. Also, analyzing Petri nets of the protocols enables the detection of structural errors, such as the absence of conditions or deadlines on loop control flow segments or the absence of acknowledgments on messages.

### Components

<table>
<thead>
<tr>
<th>Components</th>
<th>Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>UP-ColBPIP Model (Collaborative Processes based on Interaction Protocols)</td>
<td>Modelling Language UP-ColBPIP</td>
</tr>
<tr>
<td>UP-ColBPIP-to-PNML Transformations</td>
<td>Tool for Model Transformations</td>
</tr>
<tr>
<td>PNML Models</td>
<td>PNML Metamodel</td>
</tr>
<tr>
<td>Model to XML Code Transformations</td>
<td>Production Rules of XML Code</td>
</tr>
<tr>
<td>PNML Specifications</td>
<td>PNML Language</td>
</tr>
</tbody>
</table>

**Fig. 5. Pattern of Transformations to generate Petri Nets Specifications**

### 2.3. Phase 3: Implementation of Collaborative Processes

The last phase corresponds to the implementation of collaborative processes, i.e. the generation of specifications of both the processes and the partners’ interfaces by using a particular B2B standard. The first activity to be carried out is to select the B2B standards to be used. This has to be agreed by the partners according to their technological requirements.

As we have described above, the generation of B2B specifications requires the use of different languages: one to specify business processes and another one to specify partners’ interfaces. In this way, we need to manipulate technology-specific business process models and technology-specific partners’ interfaces models.

Figure 6 shows the pattern of transformations to be applied in this phase. The input again is the UP-ColBPIP model that contains the defined collaborative processes based on interaction protocols and the business interfaces of the partners. Two types of models have to be generated from a UP-ColBPIP model: technology-specific business process models and technology-specific partners’ interfaces models.

On one hand, the number of technology-specific business process models to be generated depends of the language used. For example, by using ebXML BPSS only one model have to be generated because a XML document of an ebXML BPSS specification can have several binary collaborations representing collaborative processes [22]. This is the same with WS-CDL that allows the definition of several choreo-
graphs in a WS-CDL specification [25]. But it is different with BPEL. By using
BPEL, the semantics of a collaborative process based on interaction protocol is repre-
sented by means of BPEL abstract processes (also known as public processes), one by
each partner involved in the protocol [24]. This means that a BPEL abstract process
represents the behavior of the role fulfilled by one partner in a collaborative process.
Hence, the number of BPEL abstract processes models to be generated from a UP-
ColBPIP model is the sum of all of the roles involved in all the interaction protocols.

<table>
<thead>
<tr>
<th>Components</th>
<th>Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>UP-ColBPIP Model (Collaborative Processes based on Interaction Protocols)</td>
<td>Modeling Language UP-ColBPIP</td>
</tr>
<tr>
<td>Transformations of UP-ColBPIP into Technology-Specific Process Models</td>
<td>Tool for Model Transformations</td>
</tr>
<tr>
<td>Technology-Specific Business Process Models</td>
<td>Technology-Specific Business Process Metamodel</td>
</tr>
<tr>
<td>Model to XML Code Transformations</td>
<td>Technology-Specific Partner’s Interface Metamodel</td>
</tr>
<tr>
<td>Transformations of UP-ColBPIP into Technology-Specific Interface Models</td>
<td>Production Rules of XML Code</td>
</tr>
<tr>
<td>Technology-Specific Partner’s Interface Models</td>
<td>B2B Standard Languages</td>
</tr>
<tr>
<td>Technology-Specific Business Process Specifications</td>
<td></td>
</tr>
<tr>
<td>Technology-Specific Partner’s Interface Specifications</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6. Pattern of Transformations to generate B2B Specifications

On the other hand, the number of technology-specific partners’ interfaces models
depends of the number of partners defined in the UP-ColBPIP model. For each part-
ner, a technology-specific partner’s interface model is generated. To define partners’
interfaces, ebXML provides the CPPA (Collaboration-Protocol Profile and Agree-
ment) language, and standards based on web services composition depend on the Web
Services Description Language (WSDL).

Finally, from these technology-specific models, the XML-based specifications of
business processes and partners’ interfaces are generated.

3 Tool for Model Transformations

The transformation of UP-ColBPIP models into formal or B2B specifications can be
implemented by using the model transformation tool prototype proposed in [21]. This
tool enables the definition of model transformations for generating specifications
based on a XML-language from UP-ColBPIP models.
The architecture of this tool is shown in Figure 7. The Transformation Editor (TE) implements a model transformation language that supports the declarative definition of transformation rules and their composition in rule modules consisting of a hierarchy of rules. Model transformations represent the transformations of UP-ColBPIP models into technology-specific models. A model transformation contains a set of transformation rules and rule modules. For each mapping between elements of the source and target languages, a transformation rule is defined. Transformation rules consist of LHS (Left-Hand Side) and RHS (Right-Hand Side) patterns defined according to the metamodels of the source language and the target language. These patterns are specified in an imperative way using Java code. An API is provided to define these patterns. The execution order of the rules is defined in the rule modules. A rule defined in a rule module can call to another module in order to allow recursive rules. Model Transformations are saved in a XML format. The tool contains a repository for the storage and retrieval of model transformations for the subsequent execution.

![Fig. 7. Architecture of the Model Transformation Tool.](image)

To support the execution of model transformations and generate the XML code of the specifications, the tool uses the Transformation Model Engine that takes as input a UP-ColBPIP model in an XMI format and generates technology-specific models according to the technology-specific language used. To do this, the transformation model engine has two subcomponents: the Rule Sequencer, which interprets the rule models to determine the order of execution of the rules; and the Rule Executor, which interprets each transformation rule and executes its corresponding LHS and RHS patterns. To generate XML documents (corresponding to the specifications) from the above models, the tool has a XML Code Generation Engine, which implements production rules of XML code from object models.

To manipulate UP-ColBPIP models and import them in a XMI format, the tool uses the APIs of the Eclipse UML 2 plug-in [7], which is an implementation of the UML 2 metamodel based on the Eclipse Modeling Framework (EMF) [6]. Hence, UP-ColBPIP models can be defined with any UML Case tool that implements the Eclipse
UML2 Metamodel. Also, for manipulating technology-specific models, EMF was also used to generate the metamodel and APIs that allow the manipulation of these models.

The Tool has a 3-tier architecture. The Facade separates the user interfaces tier from the main tier that contains the components described above. In this way, the user interfaces can be implemented to be used from HTML browsers or traditional clients. Currently, we have implemented this last type of client.

4 Related Work

The application of model-driven development (MDD) approaches has been exploited in the domain of web services to generate technological solutions based on web services composition [2, 15]. However, as we have appointed in previous works [24, 25], they support the modeling and specification of the behavior of collaborative processes but only from the point of view of one partner. As a result, they do not support a global view in the modeling of collaborative processes. From the methodological point of view these approaches do not also define a complete development process. In addition, they are not based on a standard conceptual framework such as MDA.

In [13], a complete development process for the generation of web services compositions was defined. However, the principles of MDA are only applied in the phase of generation of the process definitions. The verification of technology-independent business process models is not considered.

The above mentioned MDD approaches are only focused on the domain of web services and therefore, they do not also consider the application of a technology-independent modeling language for collaborative processes.

In [11,12], the transformation of technology-independent collaborative processes defined with the UN/CEFACT Modeling Methodology (UMM) to B2B specifications has been proposed. However, UMM do not provide a complete development process to generate B2B specifications. It only provides a development process for modeling technology-independent collaborative processes. In addition, the proposal defined in [11, 12] does not consider neither the application of a model-driven approach nor the verification of the processes in an early stage of the development.

Different to the above works mentioned, the development process defined in this paper exploits the principles of MDA in each phase of the development, through the application of different patterns of transformations according with the type of XML-based specification to be generated.

5 Conclusions and Future Work

In this work, we have described the development process of a MDA approach we propose for the modeling and specification of collaborative business processes. As part of this MDA approach, we have described the phases and activities to be carried out, the artifacts to be produced and the techniques to be used for building these arti-
facts. Through this MDA approach we contribute with a model-driven development method for enabling enterprises to establish B2B collaborations with their partners.

First, as a start point of the development, we have highlighted the importance of using a technology-independent modeling language for the analysis and design of collaborative processes. Hence, we propose the use of the UP-ColBPIP language. This language supports a top-down approach and encourages the use of interaction protocols for representing the behavior of collaborative processes. The use of interaction protocols allows fulfilling with the requirements of the B2B collaborations.

Second, the verification of collaborative processes previous to the generation of the technological solutions is very important in order to determine, in an early stage of the development, if collaborative processes defined in a business level are well-formed. To do this, we have indicated the pattern of transformations required to generate Petri Nets specifications based on the PNML language from UP-ColBPIP models. PNML-based specifications can be used by most of the Petri Nets Tools to verify properties of the Petri Nets representing the interaction protocols.

Third, we have indicated the pattern of transformations required to generate B2B specifications that allow the implementation of collaborative processes. Two types of technology-specific models and specifications have to be generated: those representing business processes and those representing interfaces of the partners. The number of models and specifications to be generated depends on the B2B standard selected.

Finally, we have described a model transformation tool developed to generate B2B specifications. This tool implements a particular model transformation language that allows the definition of rules that can be reused in different rule modules to specify the ordering, composition and recursive definition of the transformation rules. The patterns of the rules are defined in an imperative way. The advantage of using this tool is it was developed specifically for the application domain of collaborative processes. However, the main disadvantage of this tool is that patterns of the rules are defined in an imperative way, and hence, it is necessary developers with a well-known of the metamodels of the languages and the Java APIs that allow the manipulation of the models. Furthermore, correctness in the definition of the transformations cannot also be guaranteed. To overcome these disadvantages, we are extending this model transformation tool to define and execute rules based on graph transformations. The purpose is to reuse as model transformation engine a tool that provides a graph transformation engine for the execution of transformations and enables the graphical definition of the patterns of the rules as graphs.

Our ongoing work is also about the building of a software development environment to support the management of this complete MDA-based development process.

References