Applying Model-Driven Development to Collaborative Business Processes

Pablo David Villarreal\textsuperscript{1}, Enrique Salomone\textsuperscript{1,2}, Omar Chiotti\textsuperscript{1,2}

\textsuperscript{1}Centro de Investigación y Desarrollo de Ingeniería en Sistemas de Información, Universidad Tecnológica Nacional-Facultad Regional Santa Fe, Lavai
ta 610, 3000, Santa Fe, Argentina

\textsuperscript{2}INGAR-CONICET, Avellaneda 3657, 3000, Santa Fe, Argentina

pvillarr@frsf.utn.edu.ar, {salomone,chiotti}@ceride.gov.ar

Abstract

Currently enterprises are focusing on the setting up of Business-to-Business collaborative relationships with their partners. One of the main challenges in these relationships is the definition of the Collaborative Business Processes. 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. Therefore, in this paper we present a model-driven development method to support the design of collaborative processes. As part of this method we describe the modeling language UP-ColBPIP to design technology-independent collaborative processes. This language supports the definition of these processes through interaction protocols. In addition, we provide a model transformation method that defines how from UP-ColBPIP collaborative process models we can generate specifications based on the ebXML standard for implementing collaborative processes.

1. Introduction

Currently enterprises are focusing in the setting up of Business-to-Business (B2B) collaborative relationships with their partners in order to be competitive. In these B2B Collaborations, enterprises require a process-oriented integration between heterogeneous and autonomous partners involved in a supply chain or a collaborative network. One of the main challenges in this integration is the definition of the Collaborative Business Processes. Through the execution of collaborative processes, enterprises undertake to jointly carry out decisions to achieve common goals, coordinate their actions and exchange information [MFG00].

Two points of view of this process-oriented integration have to be considered: a technological point of view and a business point of view. The former refers on the support to collaborative business processes through the implementation of B2B information systems. These systems are composed by business interfaces that each partner implements to jointly execute collaborative processes with other partners. Interoperability between these interfaces is an important issue and requires: the explicit specification of the partners’ interfaces, and the semantics of collaborative processes being understood by all the partners’ interfaces [MBB03]. These interoperability requirements of the B2B information systems are supported by B2B standards. They allow the specification of B2B protocols [BUS02], which define both the partners’ interfaces and the collaborative processes in a format that can be processable by the applications.

Currently, there are two main technologies proposed for the specification of B2B protocols [BKK03]: standards based on Web Services Composition, such as BPEL (Business Process Execution Language) and WSCI (Web Service Choreography Interface); and standards based on Business Transactions, such as ebXML [UO] and RosettaNet. However, due to enterprises require engaging in independent B2B relationships with different partners, the above issue of interoperability has shifted from the level of applications to the level of standards [MBB03]. A partner should be able to collaborate with several partners using different standards. Moreover, a partner may require implementing a same collaborative process with several partners using different standards.

From the business point of view, the process-oriented integration between partners is addressed through the use of collaborative business models. Examples of these models are Vendor-Managed Inventory (VMI)
Initiatives [CompTIA]; Collaborative Planning, Forecasting and Replenishment (CPFR), and the Partner-to-
Partner Collaborative Model [VCZ03]. Collaborative business processes are derived from these models. In the
analysis and design phases of these processes several requirements have to be fulfilled. First, enterprises must
focus on designing collaborative processes (i.e., roles, actions and information to be exchanged) without
considering the technology used for their implementation. A collaborative process model should be
understandable and easy to read by business analysts and systems developers, who are not acquainted with the
technical details of the languages used to execute collaborative processes. Second, collaborative process
models have to provide a common semantics for the partners to understand in the same way these processes.
Finally, these models have to express suitable abstractions for the domain of B2B collaborations. Hence, the
use of conceptual models to support the definition of these processes is an important requirement.

In this way, 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. Both definitions must have a mutual
correspondence. Several works [BAG04, BKK03] have recognized the necessity of providing guidelines for:
the design of collaborative processes independent of the idiosyncrasies of particular B2B standards; and the

Therefore, in this work we propose a Model-Driven Development (MDD) method to support the design stages
of collaborative business processes. On one hand, this method allows the technology used to implement them
can be decided later. On the other hand, this method intends to reduce the inherent complexity and costs that
partners have to incur during the development of B2B systems [BAG04]. Furthermore, a model-driven
development approach allows to the models play an important role [MCF03]. Systems developers can build
and transform collaborative process models in order to generate XML code. In this work, we focus on the
generation of ebXML specifications from the technology-independent models of collaborative processes.

The proposed method for MDD is based on the principles and components proposed by the Model-Driven
Architecture (MDA) Initiative of the Object Management Group (OMG) [OMGb]. This paper is organized as
follow: Section 2 describes the MDD method along with the techniques proposed for the design of
collaborative processes. Section 3 gives an overview of UP-ColBPIP, the proposed language to define
collaborative processes in an independent way of the B2B standards. Section 4 describes the transformation
method used to generate ebXML specifications from collaborative process models defined with UP-ColBPIP.
Section 5 discusses related works. Section 6 gives conclusions and outlines future research directions.

2. Model-Driven Development of Collaborative Business Processes

The OMG’s MDA initiative [OMGb] proposes a conceptual framework along with a set of standards (UML,
MOF, XML, etc.) to build model-driven development methods. One principle of MDA is that the building of
systems can be organized around a set of models by imposing a series of transformations between models,
organized into an architectural framework of layers and transformations. In MDA, the concept of system does
not only refer to the software but also refer to an enterprise, a set of enterprises, some combination of
parts of different systems and so on. In the context of this work, the system to be built includes the partners’
interfaces of the B2B information system and the collaborative processes in an executable language.

The components of MDA are: Platform Independent Models (PIMs), Platform Specific Models (PSMs),
Transformations from PIMs to PSMs, and Transformations from PSMs to Source Code (Figure 1). A platform
makes reference to the technology that supports the system being built. However, MDA provides the
components that have to contain a MDD method, but does not provide the concrete techniques to build them.
Therefore, following the MDA approach, the techniques and components of our MDD method for designing
collaborative processes are (Figure 1):

- The UML Profile for Collaborative Business Processes based on Interaction Protocols (UP-
  ColBPIP). This is the modeling language proposed to build collaborative business process models that
  are independent of the technology. This UML Profile is based on UML 2 [OMGa].

- One or more modeling languages or metamodels used for building Collaborative Business Processes
  based on a specific B2B Standard, such as ebXML or BPEL. For the specification of collaborative
  processes in a B2B standard, it is possible to use a specific UML Profile or to directly create a UML
  class model version of the XML Schema of the B2B standard, that is, the corresponding metamodel.
• **Transformations from UP-ColBPIP models to models based on a B2B standard.** These transformations define how a UP-ColBPIP model can be converted into a model of a B2B standard. To support these transformations, a transformation method has been defined and implemented using a prototype tool for model transformation that we are developing. In this tool, model transformation rules are defined by input and output patterns according to the source and target metamodels. The tool also implements a metamodel for model transformation that allows the declarative definition of the transformation rules and the composition of them in rule modules that consist in a hierarchy of rules. The rule patterns are specified in an imperative way using Java code. The tool has a transformation engine that takes as an input a UP-ColBPIP model in a XMI format, and generates a model in a B2B standard according to the metamodel of the B2B standard. We are interested in two types of transformations. Transformations of UP-ColBPIP models to B2B standards based on business transactions, such as ebXML, and transformations of UP-ColBPIP models to B2B standards based on Web Services Composition, such as BPEL. In this work we focus on the first type of transformations.

• **PSM to XML code transformations.** The final output of the transformations is one or more XML files representing the partners’ interfaces and the collaborative processes in the XML languages provided by the B2B standard. The model transformation tool is also used for this purpose. The tool supports the generation of the XML code from the B2B standard models in direct way, according to production rules that convert a UML class diagram into a XML version.

![Figure 1. MDA and the Model-Driven Development Method for Collaborative Processes](image)

3 The UML Profile for Modeling Collaborative Business Processes based on Interaction Protocols

The UML Profile for Collaborative Business Processes based on Interaction Protocols (UP-ColBPIP) has been defined as a technology-independent modeling language to design collaborative processes [VSC04]. Following, we describe the UP-ColBPIP stereotypes that represent the main conceptual modeling primitives along with their semantics. UP-ColBPIP supports the modeling of three views of the B2B collaborations: the B2B Collaborations Context View, the Collaborative Processes View and the Interaction Protocols View. Following, the modeling elements of the first two views are briefly described. Then, we focus on the last view, from which we can generate specifications in a B2B standard. More details about this UML Profile can be found in [VSC04].

3.1. The B2B Collaborations Context View and the Collaborative Processes View

The B2B Collaboration Context View is used to define the requirements of a B2B Collaboration that partners agree on carrying out. In this view, B2B Collaborations extends the semantics of UML 2 Collaborations and are defined using Collaboration Diagrams. A B2B collaboration consists of a set of Trading Partners cooperating through binary B2B Relationships. Trading partners own Public Business Interfaces that allow them to engage in a B2B relationship and fulfill a specific Partner Role. This view also supports the definition of the Collaborative Agreement, which specifies the rules that will govern the collaboration between partners.
and a hierarchy of Business Goals to be fulfilled in the collaboration. The collaborative agreement’s attributes and the business goals are defined by using Class Diagrams.

In the Collaborative Processes View, the Collaborative Business Processes that partners have to perform in a B2B collaboration are identified. They are represented extending the semantics of Use Cases. Hence, they are defined as an informal specification of a set of actions performed by trading partners to achieve a common goal. A collaborative process can have as attributes Business Documents, which represent the information to be exchanged and shared between the partners. Also, in this view, subprocesses and exception processes associated to a collaborative process are identified. An Exception Process is a type of collaborative process that specifies how another collaborative process is finalized or corrected in case an exception occurs. In addition, the business goal that each collaborative process has to achieve is specified. The modeling elements of this view are defined through Use Case Diagrams.

### 3.2. The Interaction Protocols View

This view supports the detailed design of the collaborative processes identified in the above view. The behavior of the collaborative processes is realized through Interaction Protocols. Interaction Protocols have been used in the area of multi-agent systems for representing interactions between software agents [BMO01]. In the context of B2B collaborations, an Interaction Protocol describes a high-level communication pattern through an admissible sequence of messages between enterprises playing different roles [VSCb03]. Its main elements are: partner roles, a choreography of sending and receiving business messages between roles, speech acts, control flow operators, conditions and deadlines.

The main objective of modeling collaborative processes through interaction protocols is to fulfill the requirements of B2B interactions [VSCa03] [VSCb03]: enterprise autonomy, decentralization, peer-to-peer interactions and negotiation supports. In contrast to the traditional activity-oriented process models, used mainly by workflow approaches, a process model based on interaction protocols focuses just on the message exchange between enterprises. The activities/services that each partner performs for processing the received information or producing the information to be sent are not explicitly defined in the interaction protocols. They are kept hidden to its other partners. Internal activities/services and decisions of the enterprises are also not exposed. Hence, the enterprise autonomy is supported in the modeling of collaborative processes.

With the modeling of interaction protocols, we are providing a communication-oriented modeling approach to define collaborative processes. Communication-oriented approaches are based on the Language/Action Perspective theory (LAP) [WF86] and the Speech Acts Theory. LAP assumes that communication is a kind of action that creates commitments between the communicating parties. A speech act is the main concept of LAP and represents an intention that a speaker makes in a communicative act. Through the use of speech acts to describe the messages, the interaction protocols provide an intentional perspective to process models. The designers of a collaborative process focus on the semantics of each message selecting the most appropriate speech act. That is, speech acts allow the representation of the intentions that enterprises communicate in a collaborative process. Decisions and commitments done by the enterprises can be known from the speech acts. This enables the definition of complex negotiations in collaborative processes.

The Interaction Protocol view of UP-ColBPIP supports the definition of Interaction Protocols using the Interaction Diagrams of UML2. Therefore, interaction protocols are defined extending the semantics and constraints of the Interactions of UML2. Figure 2 shows the package of UP-ColBPIP that specifies the stereotypes along with the metaclasses they are extending. In this diagram, we also include the associations between the metaclasses to understand the UML2 metamodel as well as the implicit relationships between the stereotypes. However, these associations are not part of the profile but are part of the UML2 metamodel.

The InteractionProtocol stereotype extends the Interaction metaclass, which is the main element to represent interactions in UML. A ProtocolTemplate stereotype is derived from InteractionProtocol and it represents reusable patterns of interactions used to define interaction protocols.

In addition to the parameters in, out and inout of a UML interaction, interaction protocols require the declaration of three specific parameters indicated by the ProtocolParameterKind enumeration. The businessDocumentsOntology indicates the ontology used to semantically describe business documents. The contentLanguage indicates the language used to syntactically describe business documents. The LanguageActionLibrary indicates the speech act library used in the protocol, which indicates the semantics of
the speech acts, e.g. FIPA ACL (Agent Communication Language). These parameters are defined in the ProtocolParameter stereotype.

The Trading Partners participating in an Interaction Protocol and the Partner Role they play are represented by the Lifelines. The connectableElement attribute of a Lifeline references to a Partner Role defined in the collaborative process that the Interaction Protocol is realizing.

An interaction protocol is also composed by business messages. A BusinessMessage describes an interaction between two roles. The direction of a message, i.e. the initiator role and the recipient role, is defined by event occurrences associated to the message. EventOccurrences describe the sending and receiving events of a message on the lifelines of the partner roles. In a business message, when the isReceiptAcknowledgementRequired attribute is true, the recipient has to send a receipt acknowledgement when the message is received. When the isReadAcknowledgementRequired attribute is true, the recipient has to send a read acknowledgement if the message is received and understood.

![Figure 2. InteractionProtocols Package of UP-ColBPIP.](image)

A BusinessMessage has to indicate the speech act representing the intention associated to the business document that a partner communicates to another partner. Therefore, a SpeechAct stereotype is included, which extends the Signal metaclass. In UML, the signature association attribute of a Message can refer to a signal or an operation. However, in our modeling domain, trading partners cannot invoke private behaviors of other trading partners but they can only manage reception signals used to internally activate their behaviors. Therefore, the signature of a BusinessMessage must only refer to a signal, but not an operation.

A business document is represented by the BusinessDocument stereotype that extends the Class metaclass. The argument association attribute of a BusinessMessage must only refer to an instance of a Business Document for representing the business information being transported by the message.

The choreography of an interaction protocol is specified by the metaclasses: InteractionOccurrence, CombinedFragment, InteractionOperand, InteractionConstraint, EventOccurrence and Stop. An Interaction Occurrence is used to represent nested and interleaved interaction protocols, which are called within another interaction protocol. In UML 2, combined fragments are represented by boxes that define the different paths of an interaction protocol depending on the interaction operator used. The InteractionOperand metaclass represents each path included in a CombinedFragment.

With the objective of providing well-known control flow operators for defining collaborative processes, the CombinedFragment metaclass is extended by the ControlFlowSegment stereotype. The attribute of this stereotype refers to the ControlFlowOperators enumeration that defines the control flow operators. The XOR
operator represents that only one path, of a set of alternative paths, can be executed if its condition is evaluated to true. The \textit{AND} operator represents the execution of parallel paths in any order. The \textit{OR} operator represents the selection of several paths from several alternatives. A path is executed in case its condition is evaluated to true. This operator is different to the \textit{Option} operator in UML, which considers only one alternative. The remaining operators are those of UML. The \textit{Break} operator can be used to define paths that manage exceptions and require the abrupt termination of the protocol. The \textit{Negative} operator can be used to manage exceptions about non-understood messages. The \textit{CriticalRegion} operator can be used to define business transactions, where a message sequence has to be done atomically.

\textit{InteractionConstraints} are used to define conditions on the paths of a \textit{ControlFlowSegment} and on the \textit{BusinessMessages}. The \textit{Success} and \textit{Failure} stereotypes indicate a success or failed termination of a protocol.

To graphically define interaction protocols, we can use the UML2 Interaction Diagrams: Sequence Diagram, Communication Diagram, Interaction Overview Diagram and Timing Diagram. However, the former is the most known and provides a more expressive concrete syntax.

\subsection*{3.3. Example of a Collaborative Business Process defined with UP-ColBPIP}

As an example we describe the collaborative process \textit{Blanket Purchase Order}, which is performed as part of the collaborative model \textit{Forecast-based Supplier-Managed Inventory (SMI)}\cite{CompTIA}. This model has been proposed to carry out collaborative replenishments between a retailer and a manufacturer [CompTIA]. The collaborative process \textit{Blanket Purchase Order} is a negotiation process between a \textit{customer} and a \textit{supplier} for agreeing on an order forecast for the agreement period against which multiple short-term releases will be generated to satisfy the customer’s requirements. This is defined within the business document \textit{Blanket Purchase Order (BPO)} that also contains total quantity, conditions and pricing terms on items.

Figure 3 describes the Sequence Diagram that specifies the behavior of this process by the \textit{Blanket Purchase Order} interaction protocol. The protocol starts with the \textit{customer} proposing a \textit{BPO}. Then, a loop of different message sequences can occur more than once while negotiation is required (control flow segment...
BPONegotiation). In this loop, the supplier internally evaluates the BPO and two paths can occur. This is specified by the control flow segment BPOChanges with a Xor operator.

If changes to the BPO are required by the supplier, the first path is executed. In this path, the supplier proposes changes to the BPO, and the customer can respond in three different ways as it is indicated by the control flow segment BPOChangeResponses. One alternative is when the customer rejects the proposed changes because it does not agree with the supplier’s proposal. Another alternative is when the customer accepts the supplier’s proposal. In both alternatives the protocol finishes. The third alternative is when the customer makes a counterproposal.

If the supplier does not require changes, the second path is executed. The supplier responds with an accept-proposal that represents its acceptance to the BPO proposed by the customer. In addition, if the customer wants to initiate changes, it makes a counterproposal sending the propose(BPOChangeRequest) message.

In both paths, the interaction protocol finishes except the customer sends a propose message, when it makes a counterproposal. In these cases, the protocol continues according to the loop segment. Therefore, the supplier can again make a counterproposal or accept the BPO proposed by the customer. In this way, the negotiation can go through n times iterations. However, the duration constraint {0..6d} on the protocol defines it cannot be executed more than six days. In addition, deadlines are associated to the messages or segments in this protocol through time constraints.

4 Transforming UP-ColBPIP Models to ebXML Specifications

In this section we describe how models of collaborative processes defined with UP-ColBPIP can be used to generate ebXML specifications to enable the execution of these processes. The ebXML standard consists of a set of languages for enabling B2B interactions between enterprises through the exchange of XML-based messages. Two of these languages are relevant for our purpose: ebXML Business Process Specification Schema (BPSS) [UO] and ebXML Collaboration-Protocol Profile and Agreement (CPPA). BPSS is used to specify collaborative processes and CPPA is used to specify the operational details of the partners’ interfaces. In this work, we only describe the generation of BPSS specifications. CPPA is out of the scope of this paper.

The transformation of UP-ColBPIP models to ebXML BPSS models has been implemented using the model transformation method and the model transformation tool commented in the section 2. UP-ColBPIP models are transformed in ebXML BPSS models defined as UML class diagrams based on the BPSS metamodel. Then, these ebXML models are transformed into their corresponding XML version. The method and the tool for model transformations are out of the scope of this paper due to space limitations and because the objective of this work is to show how we can derive ebXML specifications from UP-ColBPIP models. Therefore, in this section we first provide an overview of BPSS and then we describe the transformation method showing the correspondence between the UP-ColBPIP conceptual elements and the elements of BPSS.

4.1. Overview of ebXML BPSS

BPSS [UO] uses the concept of Business Collaborations to represent collaborative processes. A Business Collaboration consists of roles that interact through Business Transactions by exchanging Business Documents. Collaborations can be; Multiparty, which involve more than two roles; or Binary, i.e. between two roles. A Binary Collaboration defines a choreography in terms of Business States, which can be Business Activities or control flow elements as Fork, Join, Decision, Transition, Start, Success and Failure. Also it contains exactly two roles specifying the partners. A Business Activity can be either a Collaboration Activity or a Business Transaction Activity. The former represents the execution of a Binary Collaboration within another Binary Collaboration. The latter represents the execution of a Business Transaction, which consists of a Requesting Activity and the Responding Activity executing one or two Business Document flows. It may also support one or more Business Signals (Acknowledgments).

4.2. The UP-ColBPIP-to-ebXML Transformation Method

The transformation of UP-ColBPIP collaborative process models into ebXML BPSS is accomplished through an ordered set of transformation rules:
I. **B2BCollaboration-to-ProcessSpecification**: This rule defines for each B2B Collaboration (B2B Collaborations Context View), a Process Specification element in BPSS is generated, which is the root of a BPSS file. This will contain all the processes defined in a B2B Collaboration.

II. **BusinessDocuments-to-BusinessDocuments**: In BPSS, Business Documents have to be defined before the definition of Business Collaborations because they can be reused in different Business Collaborations within the Process Specification defined in the BPSS file. In UP-ColBPIP, Business Documents also are separately defined of the collaborative processes because they can be reused in different processes (Collaborative Processes View). Hence, with this rule each of them is mapped to a BPSS Business Documents.

III. **SpeechAct-to-BusinessTransaction**: This rule indicates that for each Speech Act, a Business Transaction is generated. Business Transactions also can be reused in different Business Collaborations, so they also are defined before the Business Collaborations. In UP-ColBPIP, speech acts are separately defined from the interaction protocols and they can be reused in the same one or in different interaction protocols. However, a Speech Act is attached to a Business Message having only one direction, to represent the intention of a sender to a recipient when the former sends a business document. Therefore, for representing Business Messages having a Speech Act, Business Transactions are derived from the Speech Acts and defined with only the Requesting Activity indicating the Speech Act. The Responding Activity is not generated. This represents the asychronic feature of the Business Messages. The Requesting Activity is defined with a Document Envelope referencing the Business Document to be sent. This is generated from the Business Document associated to the Speech Act. Business Transactions can be derived from Speech Acts because both are defined without considering the roles, and the Requesting Activity can represent the speech act and the business document attached to it.

IV. **InteractionProtocol-to-BinaryCollaboration**: Once generated the Business Documents and the Business Transactions to be used in the Business Collaborations, we start transforming the Interaction Protocols. Those with more than two roles are transformed in Binary Collaborations related into a Multi-Party Collaboration. Those with only two roles are defined by a Binary Collaboration. In brief, we describe the transformation rules for Interaction Protocols that have only two roles. Therefore, this rule indicates each Interaction Protocol is transformed into a Binary Collaboration. For each execution of this rule, the subrules IV.1, IV.2 and IV.3 are executed having as input parameters the Interaction Protocol found and the Binary Collaboration generated.

IV.1. **PartnerRole-to-Role**: This rule is applied for each Partner Role founded in the Interaction Protocol. Roles of a Binary Collaboration are generated from the Partner Roles that the lifelines of the Protocol are referencing. The initiating role is obtained from the Partner Role referenced by the Lifeline that has the first Event Occurrence of the Protocol.

IV.2. **Fragment-to-BusinessState**: Each Fragment of an Interaction Protocol is mapped to a Business State in the Binary Collaboration generated. This rule is applied to determine if a Fragment exist in the Interaction Protocol. Therefore, this rule is executed for each Fragment of the Interaction Protocol. However, the transformation of this Fragment according to its type is done by the following subrules.

IV.2.1. **BusinessMessage-to-BusinessTransactionActivity**: This rule indicates when a Fragment is an Event Occurrence with the sendMessage association attribute referencing a Business Message and the next Fragment is an Event Occurrence with the sendMessage association attribute referencing the same Business Message. The Business Message found is transformed into a Business Transaction Activity. This is because the order of a Business Message within the Interaction Protocol’s choreography is obtained from pairs of Event Occurrences, one representing the sending of it and another representing the reception of it. The fromRole and the toRole attributes of a Business Transaction Activity are derived from the Partner Roles referenced by the Lifelines associated to the send Event Occurrence and the receive Event Occurrence respectively. A Business Transaction Activity only represents the order of the Business Message into the choreography. The one-way interaction between two roles modeled in a Business Message is represented by the Business Transaction associated to the Business Transaction Activity that is realizing the message. Hence, in the Business Transaction Activity, the reference to the Business Transaction is generated from the Speech Act associated to the Business Message. Only the reference to the Business Transaction is generated because it was created before. In addition, if the Business Message has a Time Constraint, the timeToPerform attribute is generated.
IV.2.2. **InteractionOccurrence-to-CollaborationActivity**: This rule is used to transform sub Protocols into sub Binary Collaborations. This rules indicates if the Fragment is an Interaction Occurrence, it is transformed into a Collaboration Activity. This rule also creates from the sub Interaction Protocol, the corresponding new Binary Collaboration. The reference to the new Binary Collaboration that the Collaboration Activity invokes is generated from the Interaction Occurrence’ attribute referTo that has a reference to the sub Protocol. This rule then invokes the PartnerRole-to-Role rule described above and invokes in a recursive way the Fragment-to-BusinessState to carry out the transformation of each Fragment of the sub Interaction Protocol into the Business States of the new Binary Collaboration.

IV.2.3. **Success-to-Success** and **Failure-to-Failure**: These rules indicate that a Fragment being a Success or Failure event is respectively mapped to Success or Failure state in the Binary Collaboration.

IV.2.4. **ControlFlowSegment-to-BinaryCollaboration**: The protocol’s choreography in UP-ColBPIP is supported by Control Flow Segments that contains Interaction Operands (paths), which can be composed of other ordered Fragments. Instead, in BPSS the choreography is supported by Business States and Transitions similar to state machines. Therefore, to transform the Interaction Protocol’s choreography of Business Messages into a Binary Collaboration’s choreography of Business States, this rule defines that Control Flow Segments (CFSs) are transformed into sub Binary Collaborations. In the Binary Collaboration realizing the Interaction Protocol, a Collaboration Activity is added with a reference to the new Binary Collaboration created from the CFS. The reasons for this decision are:

1. Because BPSS does not support complex control flows, as loops.
2. Because operators of CFSs have not a direct counterpart in BPSS.
3. Because CFSs can contain sub CFSs, the mapping of them to Binary Collaborations reduces the complexity of the Binary Collaboration that is realizing the Interaction Protocol.

The mapping of a CFS to a Binary Collaboration is performed according to the operator of the CFS. Table 1 summarizes this mapping. As example, a CFS with a XOR operator is mapped to a Binary Collaboration having a Fork state with an XOR value in the type attribute and a Join state with the attribute waitForAll settled in False. A CFS with a Loop operator is mapped to a Binary Collaboration having a Decision element as the first business state in case a Loop (0,n). Else, the Decision is defined before the success and failure states.

<table>
<thead>
<tr>
<th>ControlFlowSegment operator</th>
<th>Binary Collaboration with the following Business States</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xor</td>
<td>XOR</td>
<td>False</td>
</tr>
<tr>
<td>Or</td>
<td>OR</td>
<td>False</td>
</tr>
<tr>
<td>And</td>
<td>OR</td>
<td>True</td>
</tr>
<tr>
<td>Option</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Loop (0,n)</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Loop (1,n)</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

Once created the Binary Collaboration elements representing a CFS, the Fragments of each Interaction Operand of the CFS are transformed invoking in a recursive way the Fragment-to-BusinessState rule.

IV.3. **CreateBPSSTransitions**: In UP-ColBPIP there is not explicit transitions between the Fragments of an Interaction Protocol, but they are implicitly linked by the order in which they are defined. This order is indicated by the fragment association attribute of the Interaction and InteractionOperand metaclass. Therefore, due to each Fragment is mapped to a Business State, the Transitions between these states are generated after the transformation of a Fragment. That is, once the Fragments-to-BusinessState rule and one of their subrules are executed, this rule is invoked to create the Transitions that link the created Business States in the Binary Collaboration. Three types of transitions can be generated: the transition from the Start state to the Business State created when it is the first state; the transition from a previous business state to the created business state; and the transition from a previous states to a success and failure states when the Interaction Protocol does not have more Fragments and there is not a explicit termination with a Stop event. Finally, the conditions associated to each Interaction Operand are mapped to Conditions of the Transitions between the Business States generated. These conditions are generated...
According to how they were defined in the protocols, i.e. in natural language or in OCL (Object Constraint Language). However, they must be redefined later by developers to express them in XPATH.

### 4.3. Example of the UP-ColBPIP-to-ebXML Transformation Method

Following, we show an example of the above transformation method using the Blanket Purchase Order (BPO) collaborative process defined as an Interaction Protocol (IP) in section 3. The output of this transformation (the BPSS file) is shown in Figure 4, which contains some parts of the BPSS specification that is generated. As input model, we take the UP-ColBPIP model that defines the B2B Collaboration SMI Replenishment, because the BPO process has been defined as part of this B2B Collaboration. Applying the first rule, the Process Specification defined in Line 2 is generated from the B2B Collaboration SMI Replenishment. The execution of the second rule generates the business documents from those defined in the Collaborative Processes View. Lines 4-7 only define the business documents used in the IP Blanket Purchase Order.

```xml
<!-- BusinessDocuments -->
<ProcessSpecification name="SMI Replenishment" version="1.1" nameID="SMI ReplenishmentID"/>
<!-- Below are the BusinessTransactions representing derived from the SpeechActs -->
<BusinessTransaction name="propose_BlkPcktPrchseOrdRqst">
  <RequestingBusinessActivity name="propose" isIntelligentCheckRequired="true" timeToAcknowledgeReceipt="PT1H" timeToAcknowledgeAcceptance="PT1H" />
  <DocumentEnvelope businessDocument="Blanket Purchase Order"/>
</BusinessTransaction>

<BinaryCollaboration name="BC:Blanket Purchase Order" timeToPerform="P6D" initialRoleIDRF="CustomerID" />
<Role name="Customer" nameID="CustomerID"/>
<Role name="Supplier" nameID="SupplierID"/>  
<BusinessTransactionActivity name="BTA:propose_BlkPcktPrchseOrdRqst">
  <timeToComplete="P6D" />
  <ConditionExpression language="XML">NegotiationIsRequired=True</ConditionExpression>
  <success fromBusinessState="CA:BPO Negotiation" />
</BusinessTransactionActivity>
<BinaryCollaborationActivity name="CA:BPO Negotiation">
  <ConditionExpression language="XML">NegotiationIsRequired=True</ConditionExpression>
</BinaryCollaborationActivity>

......
```

Figure 4. BPSS specification generated by the UP-ColBPIP-to-ebXML transformation method.

Afterward, the Business Transactions are generated from the Speech Acts by the third rule. As example, Lines 10 to 15 defines the Business Transaction generated from the Speech Act propose with the Blanket Purchases Order document associated to it. This is used in the first Business Message of the Interaction Protocol.
Then, the InteractionProtocol-to-BinaryCollaboration rule is executed \( n \) times for each Interaction Protocol found. Each Interaction Protocol realizing a collaborative process is transformed into a Binary Collaboration. As example, the \textit{Blanket Purchase Order} Interaction Protocol is mapped to the Binary Collaboration with the same name (Lines 38 to 53). In each execution of this rule, the subrules are executed.

The \textit{Customer} and \textit{Supplier} Partner Roles are transformed into Roles of the Binary Collaboration (Lines 40 and 41) through the execution of the PartnerRole-to-Role rule for each Partner Role found. This rule also sets the reference \textit{CustomerID} in the \textit{initiatingRole} attribute of the Binary Collaboration because the \textit{Customer} is who initiates the process (Line 39).

Then the Fragment-to-BusinessState rule is executed for each Interaction Protocol’s Fragment. In each execution of this rule one subrule will be executed. As example, the first two Fragments of the Interaction Protocol are the EventOccurrences referencing the Business Message \textit{propose(Blanket Purchase Order)}. They indicates that the protocol start with this Business Message. Therefore the BusinessMessage-to-BusinessTransactionActivity rule is executed. As result of this rule the Lines 42 and 43 define the Business Transaction Activity generated and the reference to the corresponding Business Transaction before generated.

As example of the mapping of CFSs, we show the \textit{CA:BPO Negotiation} Collaboration Activity and the \textit{BC:BPO Negotiation} Binary Collaboration (Lines 46-47 and 55-79, respectively), which are generated from the \textit{BPO Negotiation} CFS through the execution of the ControlFlowSegment-to-BinaryCollaboration rule.

5 Related Work

Through the modeling of interaction protocols that define the behavior of the collaborative processes we focus on a whole view of the choreography of the messages exchanged between partners. This is the main difference with other works that focus on the modeling and automatic code generation of conversation protocols [BBC04, GAR03], i.e. the message sequence between a client and a service as part of the invocation of a Web Service. These approaches can be used to define the behavior of collaborative processes but only from the point of view of one partner. Instead, collaborative processes based on interaction protocols show the peer-to-peer interactions between the partners as a whole. In interaction protocol models, from the lifeline of each role it is possible to derive the behavior of the collaborative process from the point of view of each partner. Therefore, conversation protocols can be derived from the interaction protocols.

The generation of ebXML BPSS specifications from conceptual models of collaborative processes also has been described in [HH04]. The modeling methodology used to supports the analysis and design of collaborative processes is UN/CEFACT Modeling Methodology (UMM). Although UMM claims independence of the technology, the main conceptual elements are those used in BPSS because the BPSS metamodel is a subset of the UMM metamodel. Hence, the transformation of UMM to BPSS is simple and almost direct. However, this methodology influences the adoption of one standard. Moreover, UMM requires carrying out a lot of steps to define collaborative processes, because they are modeled in a hierarchical way, first defining business collaborations and then defining business transactions. This hierarchical approach does not enable representing in a high abstraction level, the interactions and the partners’ responsibilities within a collaborative process. They are defined within business transactions, but not when the complete collaborative process is defined as a business collaboration.

6 Conclusions and Future Work

In this work, we have proposed a model-driven development method based on the principles of MDA for the design of collaborative business processes. Two techniques of this method are the UP-ColBPIP modeling language and the transformation method to generate ebXML BPSS specifications from UP-ColBPIP models.

With respect to UP-ColBPIP, this provides a technology-independent modeling language to design collaborative business processes, enabling partners to define and agree on a common collaborative processes at a high abstraction level. The use of a UML Profile allows us: provide a vocabulary more suitable to model collaborative processes; add semantics and constraints to the UML metamodel from the modeling domain of collaborative processes; and reuse existent UML case tools to model collaborative processes. Currently, although some UML case tools are implementing UML 2, most of them are implementing only some parts of
the UML 2 metamodel. So, to specify UP-ColBPIP models in a XMI format we use Eclipse UML2, which is an implementation of the UML 2 metamodel based on the Eclipse Modeling Framework (EMF).

With respect to the method to transform UP-ColBPIP collaborative processes models to ebXML BPSS specifications, most of the BPSS elements can be derived from UP-ColBPIP. This indicates that UP-ColBPIP provides the main conceptual modeling elements required not only to design collaborative processes at a high abstraction level but also to derive executable specifications in a specific B2B standard.

Our ongoing work is about the transformation of UP-ColBPIP models to executable Web Services Composition languages, such as BPEL and WSCI, to implement collaborative processes based on interaction protocols. The most important aspect to consider in this transformation is that these languages support the definition of collaborative processes using abstract conversation protocols. This means the choreography is defined considering the view of only one role. Hence, due to interaction protocols show the whole view of the message choreography between the partners, from them we have to generate the conversation protocol required by each partner in order to implement collaborative processes.

References


